

## Short Papers

### A Layout Modification Approach to Via Minimization

Khe-Sing The, D. F. Wong, and Jingsheng Cong

**Abstract**—We present in this paper a new approach to the via minimization problem. Our approach is to systematically eliminate vias by modifying the routing layout. We have implemented our algorithm and applied it to benchmark routing layouts published in the literature, and obtained significant reduction in the number of vias without increasing the routing area. The experimental results show that our algorithm is more effective in via reduction and more efficient in running time compared to conventional via minimization algorithms. In particular, for Burstein's 19-track two-layer routing solution to Deutsch's difficult problem, our algorithm obtains 34% reduction in the number of vias, which is more than 11% improvement over the conventional constrained via minimization (CVM) approach. The application of our algorithm to various solutions to the Deutsch's difficult problem produces the fewest numbers of vias ever reported in the literature.

#### I. INTRODUCTION

In a routing layout where each signal net is formed by the wire segments interconnecting a set of electrically connected terminals, vias (contact holes) are used to connect wire segments on different layers. When two conducting layers are available for interconnection, the easiest way to route the signal nets is assigning all horizontal wire segments to one layer and all vertical ones to the other layer. However, this method uses a large number of unnecessary vias. Vias introduce some drawbacks. Besides increasing the manufacturing cost and complexity, vias in a circuit degrade its performance and reliability. The via minimization problem is to minimize the number of vias used in a layout.

The existing solutions to via minimization problem can be classified in two general categories: unconstrained via minimization (UVM), and constrained via minimization (CVM). The UVM approach [8], [11] first decides the topological information between the signal nets such that the number of required vias is minimum, and then performs a geometrical mapping of the topology to the layout. An example is shown in Fig. 1. The topology for the nets interconnecting the given terminals is shown in Fig. 1(a), and the layout after geometrical mapping is shown in Fig. 1(b). Although UVM in most cases can produce solutions with fewer vias, the layout area in general is not compact. Notice in Fig. 1(a) that absolutely minimizing the number of vias requires many nets to meander around a via. In the geometrical mapping, meandering nets use two or more tracks, which results in a larger routing area. Compared to the seven tracks needed in Fig. 1(b), the routing can

Manuscript received October 26, 1989. The work of K.-S. The and D. F. Wong was supported in part by the Texas Advanced Research Program under Grant 4096, the National Science Foundation under Grant MIP-8909586, and by an IBM Faculty Development Award. The work of J. Cong was supported in part by grants from the Digital Equipment Corporation and the General Electric Company. This paper was recommended by Associate Editor A. E. Dunlop.

K.-S. The and D. F. Wong are with the Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712.

J. Cong was with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL. He is now with the Department of Computer Science, University of California, Los Angeles, CA 90024.

IEEE Log Number 9040962.

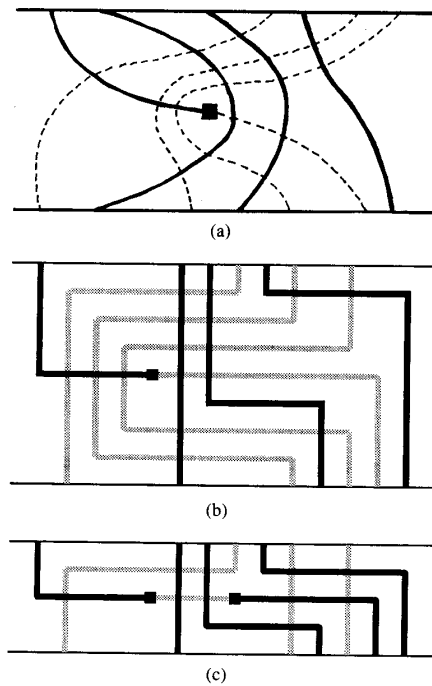


Fig. 1. UVM. (a) Topology of the nets with one via. (b) Geometrical mapping. (c) A more compact layout with two vias.

be achieved with only three tracks by allowing two vias, as shown in Fig. 1(c).

On the other hand, the CVM approach [2], [9], [12], [13] first starts with a layout that has a small area, and then assigns layers to the wire segments such that the number of vias is minimized. In this approach, the constraint is that the layout does not change, only the layers of wire segments and the positions of vias can change in the via minimization process. Fig. 2(a) shows a layout before via minimization, and Fig. 2(b) shows the same layout after CVM has been applied. The CVM approach has been well studied, and there are several polynomial time optimal CVM algorithms available in the literature.

In addition to the intricacies of its implementation, the CVM approach has another disadvantage. The result obtained by the CVM approach is minimized only with respect to the given layout. In fact, there may exist many other layouts with the same area that will achieve the same connection for the nets, but require a fewer number of vias. Notice that the layout shown in Fig. 2(a) requires five vias when it is optimized in Fig. 2(b). In Fig. 3 we show a different layout that has the same set of terminals, and the terminals have the same interconnection as in Fig. 2(a). However, it requires only three vias.

We propose a new approach to the via minimization problem, which combines the advantages of both UVM and CVM. Our method is based on the observation that the same routing problem can have different layout solutions, each layout requires different minimum number of vias. In our approach, we eliminate vias by systematic modification of the layout. We do not insert any new

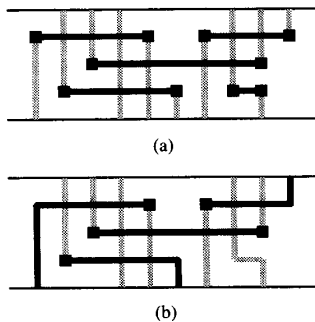


Fig. 2. CVM. (a) Before via minimization. (b) After via minimization.

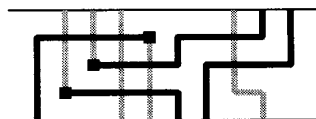


Fig. 3. Another layout with fewer vias.

tracks during the modification. Our algorithm can produce better results than CVM, and it does not increase the routing area. In fact, for many benchmark routing problems, our method produces significantly fewer number of vias compared to the result of optimal CVM algorithms.

In the next section, we describe the layout modification technique for two-layer via minimization. An algorithm that incorporates the technique is presented in Section III. In Section IV we discuss the extension of the method to multilayer via minimization. Experimental results are compared in Section V, and concluding remarks are given in Section VI.

## II. LAYOUT MODIFICATION

For via minimization, our approach uses two procedures to modify the layout. The first one eliminates a via, provided that it can find a new path to maintain the connection of the net without using the via. The new path should be accomplished without any via. In the worst case, applying this procedure does not reduce the number of vias, but it never adds any new via. To enhance the reconnecting process, we introduce another procedure to change the layer of wire segments by shifting vias along wire segments. Via shifting helps in finding a path with shorter wirelength, or in creating a new path that does not exist otherwise. Sometimes it is also useful for removing vias.

### 2.1. Reconnecting Net Components

The algorithm investigates the possibility of eliminating each via. It searches for an alternate route to connect the net components which become disjoint if the via is removed. The alternate route should be accomplished without a via. The following are the details of the algorithm in its attempt to eliminate a via.

Without loss of generality, we can assume that the wire segments do not form a loop in any net. Loops in the layout are redundant and can be removed by deleting a wire segment. Fig. 4 shows a loop in a net. The original connections among the terminals will still be preserved when a wire segment in the loop is deleted.

Let  $v$  be a via that belongs to net  $N$ . Since there is no loop in the net, via  $v$  defines two components of net  $N$  whose connection to each other depends on  $v$ . One of the components contains all the wire segments that attach to layer 1 at the location of via  $v$ , and the other component contains all the wire segments that attach to layer 2 at the location of via  $v$ . If via  $v$  is eliminated, then net  $N$

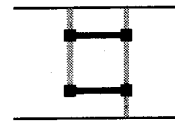


Fig. 4. A net with a loop.

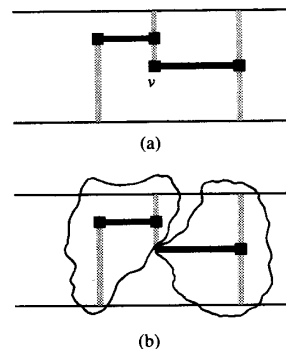


Fig. 5. The disjoint components of a net. (a) A net in the layout. (b) Components defined by via  $v$ .

is partitioned into the two disjoint components. Fig. 5 illustrates how a via defines the two components of a net. If there is a path enabling wire segments to connect both components of  $N$  without requiring a via, then the two components can be connected with wire segments along the new path. In the example shown in Fig. 6(a), a reconnecting path to replace via  $v$  is shown in Fig. 6(b). Via  $v$  then can be eliminated without altering the connectivity of the nets.

The search for an alternate path to connect the two components of net  $N$  (as defined by via  $v$ ) is as follows: starting from layer 1 at the location of the via  $v$ , which belongs to one of the components, use maze router [10] to search for any point on layer 1 that belongs to the other component. Note that wire segments of other signal nets which are on layer 1 become obstacles for the search. To make the search more effective, when it gets to a terminal belonging to net  $N$  at the border of the routing layout, we allow the search on layer 1 folding over to layer 2, and vice versa. We can also try the symmetry of the above operation by starting the search from layer 2. If the search succeeds in finding a path to connect the two disjoint components without requiring any other via, then wire segments are added along the new path to reconnect the two components. Otherwise if there is no such path, then via  $v$  cannot be eliminated by this technique.

If the two components are reconnected with wire segments along the new path, then some of the wire segments (especially the wire segments originally attached to via  $v$ ) may become redundant. Fig. 6(b) illustrates the redundant wire segments after eliminating via  $v$ . These redundant wire segments do not affect the connection among signal nets, but when we try to eliminate other vias later, they become obstacles in the search process. Thus the redundant wire segments should be deleted. The redundant wire segments are deleted unit per unit starting from the location of via  $v$ , and deletion stops when it gets to a terminal or a location where there is no more redundant wire segments.

The redundancy of wire segments can be determined by examining the existence of points with degree one on a layer. The *degree* of a point is the number of wire segments and via incident upon that point on a particular layer. Since there are at most four wire segments (each in one of the four directions) and at most one via incident upon a point on a layer, the maximum degree of a point is five. A point has degree one if and only if there is incident upon it: either only a single wire segment or only a via. Except for the

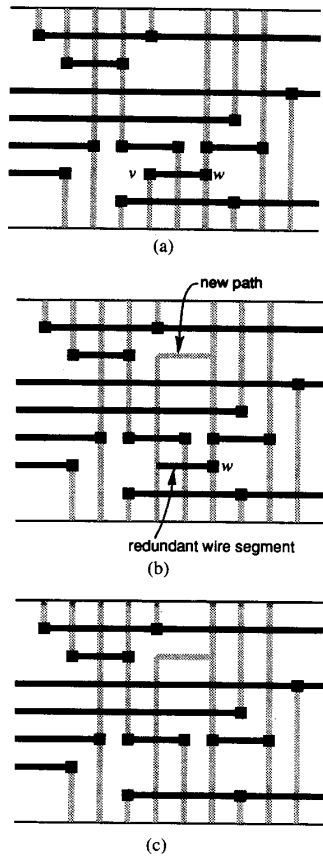


Fig. 6. Reconnecting the two components. (a) Original layout. (b) The new path and the redundant wire segment. (c) Final layout.

points on the boundary of a layout, a point with degree one indicates that the wire segment or the via incident upon the point is redundant. After a redundant wire segment or a redundant via has been deleted, we may introduce new points with degree one, which are, again, redundant. We eliminate all the redundant wires and vias by recursively deleting all the points with degree one. Fig. 6(c) shows the layout after the deletion of redundant wire segments. Note that via  $w$  disappears after it becomes attached to a point with degree one.

## 2.2. Enhancement by Shifting Vias

We introduce the technique of shifting vias to improve the results of the technique discussed in the previous subsection. Let  $v$  be a via that belongs to net  $N$ . In general, a portion of the wire segments that constitute net  $N$  is on layer 1, and the other portion is on layer 2. By shifting via  $v$  along a path in net  $N$ , the wire segment along the path where via  $v$  is shifted switches from one layer to the other as shown in Fig. 7. Via shifting is blocked when the wire segment crosses or overlaps another wire segment. Because it can switch the layers of wire segments, via shifting can be used to change the proportion of layers used by wire segments in a net.

Suppose we are searching for a reconnecting path of via  $v$  in net  $N$  on layer 1. Wire segments of other nets on layer 1 become obstacles in the search process. Hence, it is advantageous to have minimum length of all other nets on layer 1 (to reduce obstacles in the search). By shifting the vias along wire segments, we can minimize the wire segments of other nets on layer 1. Such shifting can create a path during the search on layer 1. For example, in Fig. 8(a) there is no reconnecting path to eliminate via  $v$ . However,

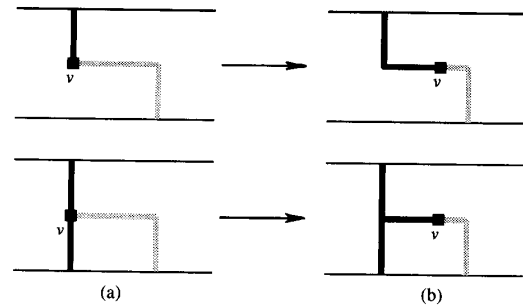


Fig. 7. Shifting vias along a wire segment. (a) Net  $N$  before shifting via  $v$ . (b) Net  $N$  after shifting via  $v$ .

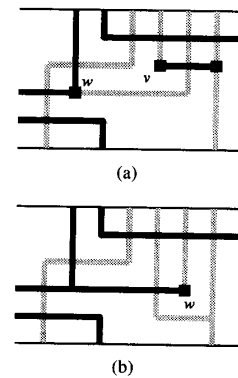


Fig. 8. Shifting a via of a different net. (a) Before shifting via  $w$ . (b) Via  $v$  is eliminated.

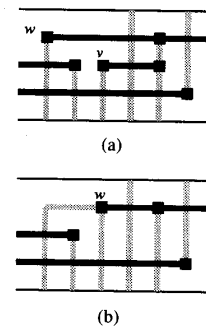


Fig. 9. Shifting a via of the same net. (a) Before shifting via  $w$ . (b) Shorter reconnection path.

after shifting via  $w$  as shown in Fig. 8(b), via  $v$  can be eliminated because now there is a path to reconnect the net components defined by via  $v$ .

We also consider shifting vias along net  $N$ . Maximizing wire segments of net  $N$  on the search layer does not help to create a new path that does not exist otherwise, but in general it helps to shorten the wirelength and to avoid unnecessary overlaps of the reconnecting path. As shown in Fig. 9, the path to reconnect the components defined by via  $v$  becomes shorter after via  $w$  has been shifted.

In addition to its role to facilitate reconnection, shifting vias itself can reduce the number of vias. This happens in two situations. If a via can be shifted to the margin of the layout, then the via disappears. If a via is shifted and it "bumps" into another via, then the two vias can either merge into one or disappear altogether.

### III. THE VIA MINIMIZATION ALGORITHM

The algorithm considers the possibility of eliminating each via in the layout. For each via  $v$ , the algorithm attempts to reconnect the two components defined by via  $v$  as described in Section II-2.1. The search for the reconnecting path is conducted twice, each one starts from a different component. Before the search begins, all other vias are shifted to facilitate the search. Of the two search results, the better one (in terms of the number of eliminated vias and the wirelength of the reconnection path) will be adopted.

If the search is successful, then the layout is actually modified by 1) reconnecting the two net components with wire segments along the new path, 2) eliminating via  $v$ , and 3) deleting wire segments and other vias that become redundant after via  $v$  has been eliminated.

The following is the high level description of the steps in the algorithm.

FOR each via  $v$  in the layout DO

1. Assume that via  $v$  belongs to net  $N$ . Find the components of net  $N$  as defined by via  $v$ .
2. To find an alternate path reconnecting the two components of net  $N$  obtained in step 1, explore the results achieved by starting the search at layer 1 and starting at layer 2:
  - 2.1. Shift all other vias to maximize wire segments of net  $N$  on layer 1 and minimize wire segments of other nets on layer 1. Starting from layer 1, use the maze router to find a path to reconnect the two components.
  - 2.2. Shift all other vias to maximize wire segments of net  $N$  on layer 2 and minimize wire segments of other nets on layer 2. Starting from layer 2, use the maze router to find a path to reconnect the two components.
3. Compare the two paths found in step 2.1 and in step 2.2, in terms of number of vias eliminated and wirelength of the reconnection path.
4. Based on the comparison in step 3, the better rerouting path is selected to be used in the next iteration. If the components are reconnected, then eliminate via  $v$  and delete redundant wire segments.

END FOR.

The time complexity for the algorithm in the worst case is  $O(vn)$ , where  $v$  is the number of vias, and  $n$  is the number of grid points in the layout. The algorithm's complexity is derived from the fact that for each via we attempt to eliminate, both maze routing and via shifting require  $O(n)$  time. However, in most cases the initial layouts are dense. Since the maze routing and the via shifting procedures usually take only  $O(1)$  time for dense layouts, the algorithm practically runs in  $O(v)$  time.

### IV. EXTENSION TO MULTILAYER LAYOUT

Our layout modification approach to via minimization can be extended to handle multilayer layouts. In a multilayer layout, two vias  $v$  and  $w$  are connected if i) via  $v$  connects the wire segments on layers  $i$  and  $i + 1$ , ii) via  $w$  connects the wire segments on layers  $i + 1$  and  $i + 2$ , and iii)  $v$  and  $w$  are attached to each other on layer  $i + 1$ . A *via-pile* is a maximal set of connected vias. Note that in a layout with  $k$  layers, the maximum size of a via-pile is  $k - 1$ . The main difference between the two-layer case and the multilayer case is the existence of via-piles. We consider two different ways to count the number of vias in a layout, depending on how we count the number of vias in a via-pile. The first way, which is the more reasonable way, is to count a via-pile of  $j$  vias as  $j$  vias. The other way is to treat a via-pile as a special via and count it as one via.

In the procedure for reconnecting net components, a via in a multilayer layout is eliminated in the same way as a via in a two-layer layout. A via can be eliminated if the two net components defined by the via can be reconnected by an alternate path, and the new path does not require another via. The search for an alternate

path is done by a maze router. If the search in the starting layer (where one layer of the via is located) fails, then the search is continued to the other layers until the other component is found, or until there is no possible reconnection. The removal of redundant wire segments is similar to the two-layer case. In the case when a via-pile is considered as one special via, we are not allowed to eliminate a via in the middle of a via-pile because it would break the via-pile into two smaller via-piles and increase the via count.

Shifting of vias in a multilayer layout is done as in the two-layer case. However, when we try to maximize the wire segments of a particular net and minimize all other wire segments on the search layer, wire segments of two other layers (the layer immediately above and below the search layer) might be affected. Only when the search layer is either the top or the bottom layer, will there be only one other layer affected, exactly as in the case of two-layer.

### V. EXPERIMENTAL RESULTS

The algorithm was implemented in Pascal language on a Sun 3/50 workstation running Unix 4.2BSD. We also implemented the optimal CVM algorithm described in [12] for comparison. The performance was tested with many published channel routing solutions, including different solutions to Deutsch's difficult problem. The result of comparison is shown in Table I.

In the first column of Table I, the notations YK3a, YK3b, YK3c, and YK5 refer to Yoshimura and Kuh [18, examples 3a, 3b, 3c, and 5]. The notations Deutsch19a, Deutsch19b, Deutsch20, and Deutsch28 refer to different solutions to the Deutsch's difficult problem. Specifically, Deutsch19a refers to the 19-track 9-dogleg solution by Deutsch mentioned in [5], Deutsch19b refers to Burstein and Pelavin's 19-track solution [1], Deutsch20 refers to Yoshimura and Kuh's 20-track solution with restricted doglegs [18], and Deutsch28 refers to another Yoshimura and Kuh's solution with 28 tracks without doglegs [18]. The second column is the number of vias when all horizontal wire segments are assigned to one layer and all vertical segments to the other layer. The third column is the number of vias produced by the optimal CVM algorithm, and the fourth column is the number of vias produced by our algorithm. The layout obtained by applying our algorithm to example Deutsch19a is shown in Fig. 10.

The results obtained by our algorithm are significant improvements over those of the optimal CVM algorithm. For example, in Burstein and Pelavin's 19-track routing solution to the Deutsch's difficult problem, our result shows more than 34% reduction in the number of vias or more than 11% improvement over the optimal CVM algorithm. We would also like to point out that our method produces the smallest number of vias for the Deutsch's difficult problem ever reported in the literature. The comparison is displayed in Table II.

We note that the usage of maze router in our approach does not introduce unnecessarily long wire segments. In some cases the total wirelength is slightly reduced, and in the worst cases it is increased only by 2.2%. The total wirelengths of the example layouts before and after via minimization are shown in Table III.

The running time of our algorithm is much shorter than the optimal CVM algorithm. For example, our algorithm takes less than 10 min to solve each of the 19-track examples in Table I, while the optimal CVM algorithm takes more than 5 h.

We also implemented our algorithm for three-layer via minimization. The experimental results are shown in Tables IV and V. In Table IV, the layouts before minimization were obtained by applying the algorithm in [4] to convert the two-layer routing solutions into three-layer routing solutions. A different set of three-layer layouts, obtained by optimal VHV algorithm in [3], is also used as inputs to our algorithm. The results are shown in Table V. Note that for the same routing problem, the solutions in Table V consistently use fewer number of vias than those in Table IV. It is because the original solutions in Table IV are essentially HVH solutions which in general require fewer number of tracks than the VHV solutions in Table V. Due to the larger number of tracks used

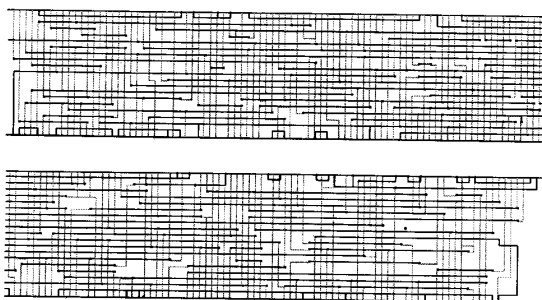


Fig. 10. A 19-track solution to Deutsch's problem with 226 vias.

TABLE I  
RESULTS OF VIA MINIMIZATION

EXAMPLE	NUMBER OF VIAS		
	Before minimization	Optimal CVM algorithm	Our method
YK3a in [18]	91	72	66
YK3b in [18]	107	91	78
YK3c in [18]	125	109	103
YK5 in [18]	150	114	105
Deutsch19a in [5]	301	252	226
Deutsch19b in [1]	355	263	233
Deutsch20 in [18]	308	250	223
Deutsch28 in [18]	290	239	212

TABLE II  
RESULTS FOR DEUTSCH'S PROBLEM

SOLUTION	Number of tracks	Number of vias
Yoshimura and Kuh [18]	20	308
Yoshimura and Kuh [18]	28	290
Burstein and Pelavin [1]	19	354
YACR [14]	19	287
MIGHTY [17]	19	303
Royle <i>et al.</i> [16]	81	271
Hamachi and Ousterhout [7]	20	412
Greedy [15]	20	347
G. C. R. [6]	—	263
Our result	19	226
Our result	19	233
Our result	20	223
Our result	28	212

TABLE III  
WIRELENGTH COMPARISON

EXAMPLE	WIRELENGTH		
	Before	After	Difference
YK3a in [18]	1365	1362	-0.22%
YK3b in [18]	1651	1653	+0.12%
YK3c in [18]	2100	2100	0%
YK5 in [18]	2785	2801	+0.58%
Deutsch19a in [5]	4989	5058	+1.38%
Deutsch19b in [1]	5049	5137	+1.74%
Deutsch20 in [18]	5078	5153	+1.48%
Deutsch28 in [18]	6231	6368	+2.20%

TABLE IV  
RESULTS OF THREE-LAYER VIA MINIMIZATION

EXAMPLE (converted into 3-layer based on algorithm in [4])	NUMBER OF VIAS	
	Before minimization	After minimization
YK3a in [18]	91	57
YK3b in [18]	114	74
YK3c in [18]	129	95
YK5 in [18]	150	90
Deutsch19a in [5]	301	183
Deutsch19b in [1]	360	207
Deutsch20 in [18]	313	181
Deutsch28 in [18]	316	185

TABLE V  
RESULTS OF THREE-LAYER VIA MINIMIZATION

EXAMPLE (obtained by optimal VHV algorithm in [3])	NUMBER OF VIAS	
	Before minimization	After minimization
YK3a	92	50
YK3b	108	54
YK3c	128	66
YK5	158	80
Deutsch	301	174

in the VHV solutions, there are more space for the maze router to eliminate vias for the solutions in Table V.

## VI. CONCLUDING REMARKS

In this paper, we introduce a new approach to via minimization by modifying the layout and systematically eliminating the vias without increasing the routing area. Our algorithm obtains better results and runs more efficiently compared to optimal CVM algorithms.

We also made experiments with iterating our program (i.e., feeding the new layout back as input into the program after all vias have been considered), and using it as a preprocessor and as a post-processor for the optimal CVM algorithm. The running time was substantially increased, but in almost all cases there were no further reduction in the number of vias.

Throughout the discussions, we assume that net terminals are accessible from both layers. This is not a limitation of our approach, because it is trivial to modify our algorithm such that each terminal can only be accessed from a certain layer.

Finally we note that, due to the greedy nature of our algorithm, sometimes in the final layout there are wires which are unnecessarily detoured. To improve the quality of the final layout, we can use a clean-up program to straighten these unnecessarily detoured wires.

## ACKNOWLEDGMENT

The authors would like to thank David Deutsch for his helpful comments.

## REFERENCES

- [1] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 223-234, Oct. 1983.
- [2] R. W. Chen, Y. Kajitani, and S. P. Chan, "A graph-theoretic via minimization algorithm for two-layer printed circuit boards," *IEEE Trans. Circuits Syst.*, vol. CAS-30, pp. 284-299, May 1983.

- [3] Y. K. Chen and M. L. Liu, "Three layer channel routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 156-163, 1984.
- [4] J. Cong, D. F. Wong, and C. L. Liu, "A new approach to three- or four-layer channel routing," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 1094-1104, Oct. 1988.
- [5] D. N. Deutsch, "Compacted channel routing," in *Proc. Int. Conf. CAD*, 1985, pp. 223-225.
- [6] P. Groeneveld, H. Cai, and P. Dewilde, "A contour-based variable-width gridless channel router," in *Proc. Int. Conf. CAD*, 1987, pp. 374-377.
- [7] G. T. Hamachi and J. K. Ousterhout, "A switch-box router with obstacle avoidance," in *Proc. 21st Design Automation Conf.*, 1984, pp. 173-179.
- [8] C.-P. Hsu, "Minimum-via topological routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 235-246, Oct. 1983.
- [9] Y. S. Kuo, T. C. Chern, and W. K. Shih, "Fast algorithm for optimal layer assignment," in *Proc. 25th Design Automation Conf.*, 1988, pp. 554-559.
- [10] C. Y. Lee, "An algorithm for path connection and its application," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 346-365, 1961.
- [11] M. Marek-Sadowska, "An unconstrained topological via minimization problem for two-layer routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 184-190, July 1984.
- [12] N. J. Naclerio, S. Masuda, and K. Nakajima, "Via minimization for gridless layouts," in *Proc. 24th Design Automation Conf.*, 1987, pp. 159-165.
- [13] R. Y. Pinter, "Optimal layer assignment for interconnect," in *Proc. Int. Conf. Circuits Comput.*, 1982, pp. 398-401.
- [14] 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.
- [15] R. Rivest and C. Fiduccia, "A 'greedy' channel router," in *Proc. 19th Design Automation Conf.*, 1982, pp. 418-424.
- [16] J. Royle, M. Palczewski, H. VerHeyen, N. Naccache, and J. Soukup, "Geometrical compaction in one dimension for channel routing," in *Proc. 24th Design Automation Conf.*, 1987, pp. 140-145.
- [17] H. Shin and A. Sangiovanni-Vincentelli, "MIGHTY: A 'rip-up and reroute' detailed router," in *Proc. Int. Conf. CAD*, 1986, pp. 2-5.
- [18] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 25-35, Jan. 1982.

## Efficient Simulation of MOS Circuits

Reinhard Erwe and Norio Tanabe

**Abstract**—This paper describes new techniques which significantly speed up classical circuit simulation of MOS LSI circuits. By taking advantage of the unilateral properties of MOS transistors, modifications of Newton's method are developed reducing the computational effort for the Gaussian elimination mainly for large circuits. Latency is another property of MOS circuits which can be exploited to enhance simulation efficiency. This paper describes a new latency exploitation technique. In contrast to methods published previously, no partitioning into subcircuits is required. These new techniques can be easily implemented in existing circuit simulators.

Manuscript received January 17, 1989; revised January 9, 1990. This work was supported by the NEC Corporation. This paper was recommended by Associate Editor R. K. Brayton.

R. Erwe is with the Institut für Theoretische Elektrotechnik, D-5100 Aachen, Germany.

N. Tanabe is with the Institut für Theoretische Elektrotechnik, D-5100 Aachen, Germany, on leave from the NEC Corporation, Kawasaki, Japan. IEEE Log Number 9040966.

## I. INTRODUCTION

Circuit simulation programs like ADVICE [1], ASTAP [2], MEDUSA [3], and SPICE [4] are well established and indispensable tools in today's VLSI design. They are used to optimize circuit performance and help to uncover design errors in quite an early stage of design. Large circuits, even if the analysis is restrained to a critical path, can include thousands of transistors. This computational expenditure and its heavy use make circuit simulation one of the main consumers of computing resources in many companies. Speeding it up will not only reduce computer costs, but also the time the designer has to wait for results.

Two different approaches to enhance performance of circuit simulation are reported in literature. One is to exploit vectorization or parallelization or both on respective hardware, e.g., [5]-[7]. This paper deals with the second approach, which does not require dedicated hardware, but is also suitable to the conventional computer architecture, still being the most common one. Instead, the program is restricted to a special class of circuits like MOS circuits, to utilize, e.g., its unilateral signal flow, and the tight accuracy requirements are relaxed for the benefit of speed. The most prominent representatives for this approach to efficiency are MOTIS [8]—later extended to an event-driven simulator [19], [20]—and the waveform relaxation (WR) method [9]. MOTIS pioneered the class of timing simulators which are designed to bridge the gap between the highly accurate, but time and memory consuming circuit simulation, and the very fast, but not very detailed, logic simulation. The efficiency of the original MOTIS code is based on a computationally inexpensive transistor model with constant capacitances, and hence, limited accuracy.

This paper deals with algorithms which deviate in less degree from classical circuit simulation, the high accuracy of which can only be fully exploited if highly accurate transistor models are used. In context of this paper a charge-based table model [10] is employed, which combines high accuracy with high flexibility and low computational costs compared to analytical models. It stores the steady-state drain current and three terminal charges as functions of three terminal voltages. The tables are populated using a two-dimensional device simulator.

In this paper new speedup techniques of classical circuit simulation are discussed. Neglecting feedback capacitances of MOS transistors in the Jacobian results in a significantly sparser matrix without affecting accuracy. A new latency exploitation technique gives substantial saving of computer time. Unlike previously published methods, not only subcircuits but single nodes, can become latent. Thus the circuits do not have to be partitioned into blocks.

This paper is organized as follows. In Section II we give a brief description of classical circuit simulation. In Section III we discuss the new previously mentioned speedup techniques of classical circuit simulation. Results are compared against classical circuit simulation and WR. Section IV gives a conclusion of this paper.

## II. CLASSICAL CIRCUIT SIMULATION

As a test vehicle for classical and modified circuit simulation we developed the experimental program MOGUL (MOS Circuit Simulator with Gaussian Elimination Utilizing Latency). Though this is not a prerequisite to our modified circuit simulation, MOGUL is restricted to nodal analysis. The dynamic behavior of an electronic circuit is thus described by the initial value problem

$$C(\mathbf{u}(t), \mathbf{v}(t)) \cdot \dot{\mathbf{v}} + \mathbf{f}(\mathbf{u}(t), \mathbf{v}(t)) = \mathbf{0}$$

with  $\mathbf{u}: \mathbb{R} \rightarrow \mathbb{R}^s$  being the vector of stimuli (driven by grounded voltage sources),  $\mathbf{v}: \mathbb{R} \rightarrow \mathbb{R}^n$  the vector of unknown node voltages,  $\mathbf{f}: \mathbb{R}^s \times \mathbb{R}^n \rightarrow \mathbb{R}^n$  the vector of static currents, e.g., steady-state drain currents, and  $\mathbf{C}: \mathbb{R}^s \times \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$  the capacitance matrix. Displacement currents due to capacitances connected to time-varying grounded voltage sources are included in  $\mathbf{f}$ .