A Performance-Driven MCM Router with Special Consideration of Crosstalk Reduction

Dongsheng Wang Ernest S. Kuh
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720

Abstract

This paper presents a new performance-driven MCM router, named MRC, with special consideration of crosstalk reduction. Router MRC completes an initial routing with an adequate performance trade-off including wire length, vias, number of layers, timing and crosstalk. Then a crosstalk reduction algorithm is used to make the routing solution crosstalk-free without big influence on other routing performances. Thus, efficiently handling timing and crosstalk problems becomes the unique feature of MRC. Router MRC has been implemented and tested on MCM benchmarks and the experimental results are very promising.

1 Introduction

Performance-driven MCM routing is very difficult because of its high complexity. Some MCM routing approaches have been proposed [1]-[8]. The four-via routing algorithm V4R [1] routes two adjacent layers (i.e., a layer-pair) at one time. For each layer-pair, V4R processes columns one by one from left to right. But V4R's column-by-column method may introduce more vias in the large-size grid routing. The MCG algorithm in [2] considers a small number of possible routes for each net and constructs a compatibility graph. Then this compatibility graph is reduced to yield a subset of routes which are fully compatible. Finally, a three-phase routing strategy is used to route nets with as few vias as possible. The V4C algorithm in [3] completes the MCM routing with consideration of crosstalk constraint. But its crosstalk estimation method is not appropriate for high performance MCM routing. The MCM router M^2R in [4] shows a lower bound of the time delay based on the lower bound of wire length and the number of vias. However this estimation is not accurate because the time delay of interconnection depends not only on the wire length and vias, but also on the net topology. Recently, a timing-driven MCM router MLR is proposed in [5]. MLR produces a Steiner tree for each multi-pin net to maintain the completed net topology instead of splitting it into several two-pin nets. Thus, the accurate timing estimation can be obtained. However, the crosstalk issue is still not addressed.

In this paper, we propose a new performance-driven MCM router, named MRC, which extends router MLR in [5] by introducing crosstalk constraint. The goal of router MRC is to pursue the final crosstalk-free detailed routing with an adequate multi-performance trade-off. The performance issues considered by MRC include the number of routing layers, vias, the total wire length, timing and crosstalk. The key feature of MRC is the ability of efficiently handling timing and crosstalk problems for high performance MCM routing.

2 Crosstalk Estimation

Crosstalk is a kind of parasitic coupling, i.e., mutual capacitances and inductances, between neighboring signal nets. In deep sub-micron technologies, the wire capacitances and inductances become critical in timing and crosstalk estimation. So a simple crosstalk model can be defined as the coupling length between the parallel routed adjacent nets (wires). Let cpl_{ij} be the parallel coupling length between adjacent net n_i and net n_j. We define the crosstalk CT_i of net n_i as the sum of the coupling length between n_i and all the adjacent nets $n_j \in N$, i.e.,

$$CT_i = \sum_{n_j \in N, j \neq i} cpl_{ij} \tag{1}$$

Where N denotes the net set. During the routing process, the crosstalk is measured by the coupling length between the net being routed and nets already routed during the routing process. When a net has been routed, the coupling length, i.e., the crosstalk, can be accurately computed. Fig. 1 illustrates an example for crosstalk estimation. In Fig. 1, the net indicated in thick solid line is the net being routed, and all the other nets indicated in thin solid lines are already routed. The crosstalk of the net being routed is $1+4+1+3+6+1+3+5=24$.

In MCM routing, the crosstalks of all nets should be reduced to tolerable values. Usually, an upper bound of crosstalk is imposed to each net as a routing constraint. When all the nets' crosstalks are lower than their upper bounds, the routing is considered "crosstalk-free". Let $N_{CPL} \subseteq N$ denote a subset of the nets having crosstalk constraint, and CPL_i the upper bound of the crosstalk of net n_i. The routing solution is considered crosstalk-free when the following condition is satisfied:

$$CT_i \leq CPL_i, n_i \in N_{CPL} \tag{2}$$

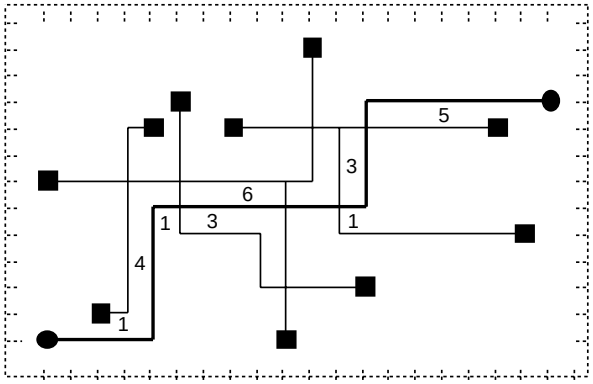

Fig. 1. Crosstalk Estaimation

Because the definition of our crosstalk model is based on the coupling length of neighboring nets, the crosstalk upper bound of a net should be related to the wire length of the net. According to our experience, the simplest definition of the crosstalk upper bound can be computed by the following formula:

$$CPL_i = c \times l_i, 0 < c \leq 1 \tag{3}$$

Where l_i is the wire length of net n_i. c $(0 < c \leq 1)$ is a constant. Fig. 2 shows an example of the crosstalk-free routing. The routing example is same with Fig. 1, but the routing solution is different. The digitals beside the routes are the coupling length of the corresponding nets. The routing in Fig. 2 satisfies the crosstalk constraint (2), i.e., all the nets in Fig. 2 are crosstalk-free in terms of equations (2) and (3) with $c = 0.5$.

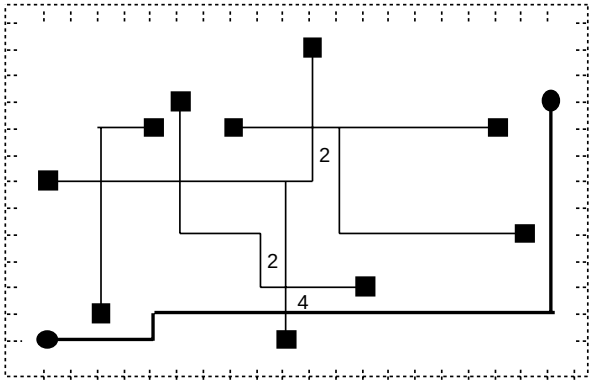

Fig. 2. Crosstalk-Free Routing

3 Problem Formulation

Assume that there is a Manhattan routing grid imposed on each routing layer where the space between grid lines is determined by the routing pitch for the given technology. Given a net set N whose pins locate on the routing grid, and each net may have two or more pins. MRC routes all the nets layer-pair by layer-pair based on the routing grid. Each net is routed by some horizontal and vertical segments. A layer-pair uses one layer for horizontal segments and the other for vertical segments. So no parallel wires are routed on the adjacent layers. Thus, the couplings among wires belonging to different layers can be ignored.

The timing computation is based on Elmore delay model. For a given net $n_i \in N$, if $d_i(j)$ is the Elmore delay at sink $s_i(j)$ [10], then the maximum sink delay del_i of net n_i becomes:

$$del_i = max(d_i(j), s_i(j) \in n_i) \tag{4}$$

The input of our routing problem includes 1) a given routing grid, 2) a net list, and 3) crosstalk constraint $N_{CPL} \subseteq N$. The output is the Steiner tree set of all the nets on the routing grid. The routing objectives includes 1) using as few layers as possible to complete the routing, 2) minimizing time delay $del_i, \forall i, n_i \in N$, and 3) minimizing

$$\sum_{n_i \in N} \alpha \cdot l_i + \beta \cdot via_i + \gamma \cdot CT_i \tag{5}$$

subject to equations (2). Here via_i is the number of vias used for routing net n_i. In our approach, the different routing objectives are achieved by different routing phases which are described in section 4. Parameters α, β, γ are experimentally determined, as shown in section 4.2.

4 Performance-Driven MCM Routing Algorithm

4.1 Overview of the Algorithm

Router MRC uses two phases to complete the routing. The first phase complete an initial routing based on layer-pair based routing approach, which uses as few routing layers as possible. For each layer-pair, a timing-driven Steiner optimal area routing (SOAR) algorithm generates a Steiner tree topology for each net. SOAR algorithm is a variant of SERT (Steiner Elmore Routing Tree) algorithm. Starting with source node, SOAR algorithm constructs a Steiner tree by extending the sinks one by one into the tree. For each edge of the Steiner tree, a crosstalk-driven shortest path algorithm, called the (α, β, γ) algorithm, is used to complete the physical connection of the edge. Therefore, SOAR algorithm with (α, β, γ) algorithm together can minimize the maximum sink delay while satisfying crosstalk constraint for each individual net. It guarantees the net being routed to be crosstalk-free in the current routing status. However, as the routing process continues, the nets being routed may cause some new crosstalk violations to the nets routed before. Thus, when the entire routing completed at the end of phase one, the routing may not be crosstalk-free. Therefore, in phase two of router MRC, a crosstalk reduction process is used to adjust some nets' routes to obtain the final crosstalk-free routing. An efficient crosstalk reduction algorithm, called the CR algorithm, is applied in this phase.

In the following sections, we will focus on (α, β, γ) algorithm and CR algorithm. The details about SOAR algorithm can be found in [5].

4.2 (α, β, γ) Algorithm

In phase one of MRC, the crosstalk constraint is introduced to initial routing by the crosstalk-driven shortest path algorithm (α, β, γ). It extends from

$(\alpha,\beta)^*$ algorithm in [5] which uses the basic ideas of (α,β) algorithm in [9]. Algorithm (α,β,γ) finds a path P to connect two given nodes, the source p_s and the target p_t, on the Manhattan plane. Parameters α,β and γ are used to build a weighted routing cost function in order to obtain an adequate trade-off among three routing performances, total wire length, the number of vias, and crosstalk. Corresponding to equation (5), the path cost function is factorized by parameters α,β and γ, and defined as follows:

$$pathCost(\alpha,\beta,\gamma) = m_P + \alpha\cdot(l_P - m_P)/2 + \beta\cdot via_P + \gamma\cdot cpl_P \quad (6)$$

Here, l_P is the real path length from p_s to p_t, m_P is the Manhattan distance between p_s to p_t, via_P is the number of vias of the path P, and cpl_P is the coupling length along the path. Obviously, the cost function *pathCost()* reaches its minimal value m_P when $l_P = m_P$, $via_P = 0$ and $cpl_P = 0$. $pathCost() > m_P$ means a penalty on the path. The only difference between equation (5) and equation (6) is the penalty on path length. In fact, Equation (5) penalizes path so long as path length satisfies $l_P > 0$, while equation (6) penalizes path when path length satisfies $l_P > m_P$. $m_P > 0$ is a constant for the given p_s and p_t. So equation (6) is consistent with equation (5).

Minimizing the cost function (6) means a "cheapest" path with shorter path length, fewer vias and less crosstalk. The basic idea of the (α,β,γ) algorithm is to find such a cheapest path P from source p_s to target p_t on the routing grid.

(α,β,γ) Algorithm

Input: Two nodes p_s and p_t

Output: Optimal path P from p_s to p_t

```
1.  initialization()
2.  Nodes = Ψ, p = p_s, P = {p_s}
3.  while p ≠ p_t and Nodes ≠ Φ do
4.      Nodes = extendNode(p)
5.      Labels = updateLabel(Nodes)
6.      p = chooseExtendNode(Labels)
7.      P ← P ∪ p
8   if p ≠ p_t then P = Φ
9.  detour if P = Φ
10. output P
```

Fig. 3. (α,β,γ) Algorithm

Fig. 3 shows the outline of the algorithm. The procedure *initialization()* in line 1 generates the minimum bounding box including nodes p_s and p_t. Each node in the bounding box is assigned a label representing the cost of the path from source to the node. The path cost is estimated by equation (6). Those nodes that have been occupied by other nets are marked as routing blocks and can be no longer used by the path. Initially, all the node labels in the minimum bounding box are set to infinite, the path P only contains the source node p_s, and a extending node set *Nodes* is set to non-empty set Ψ. Then, an iterative procedure (line 3 to line 7) is used to extend the path within the bounding box. At each iteration, set *Nodes* is used to hold all the nodes extended by procedure *extendNode()*. Each node in *Nodes* is assigned a definite label according to the cost function (6) and added into a label set *Labels* by the procedure *updateLabel()*. Procedure *chooseExtendNode()* chooses a new extending node p with minimum label in set *Labels*. The node p becomes a new extending node on the path. The iteration terminates when the node p becomes p_t or no adequate path is found ($Nodes = \Phi$). If no adequate path is found, line 9 tries a detour process for the path from p_s to p_t by extending the bounding box and using (α,β,γ) algorithm again.

The values of parameters α, β, and γ are experimentally determined. Parameter α controls the path length, i.e., the routing will be penalized when it detours. The penalty strength is associated with α. Parameter β controls the number of vias. Wherever a via is used, the penalty is applied by β. Practically, $\alpha < \beta$. Parameter γ controls the crosstalk (parallel coupling length). Because any coupling is penalized by γ, and the coupling exists in most cases, the γ value should not be too large compared to α and β. Usually, let $\gamma < \alpha$, $\gamma < \beta$. For all benchmarks tested by router MRC, $\alpha = 2$, $\beta = 3$, $\gamma = 1$. Experiments show that these parameter values are reasonable.

4.3 Crosstalk Reduction

Algorithm CR uses the global/local net adjusting approach. For the sake of convenience, we use the variable "*xtalk*" to present the ratio of the number of nets that have crosstalk violations over the total number of nets. We assume that there exists crosstalk violation with $xtalk = X_i$ before crosstalk reduction, then *xtalk* becomes X_g after global net adjusting, and finally it goes down to X_l when crosstalk reduction algorithm terminates. Obviously, $X_i \geq X_g \geq X_l$, and the final crosstalk-free routing requires $X_l = 0$.

Algorithm CR contains three steps, i.e., global, local and alternative global/local net adjusting, as shown in Fig. 4. Step 1 operates a global adjusting for those nets with crosstalk violation using an iterative rip-up and re-routing approach. For each iteration, all the nets with crosstalk violation are completely ripped up and re-routed by using SOAR algorithm [5]. Then procedure *estimateCrosstalk()* is applied to compute *xtalk* X for the current routing solution. The iteration terminates when *xtalk* is reduced to a given value X_g. Because the rip-up and re-routing may change the topologies of some nets, and cause the different crosstalk distribution over the entire routing area, step 1 does not guarantee the crosstalk-free routing solution.

In the step of local adjusting, only some segments of some nets are ripped up and re-routed to further reduce the crosstalk without causing any new crosstalk violation, which requires us to restrict each net to be adjusted within a very limited local region. This is implemented by net adjusting routines *changeLayer()* and *detour()* in Fig. 4. Routine *changeLayer()* moves a segment from current routing layer into another one. A successful movement means that a feasible route for the segment is available on that layer and no new crosstalk violation is caused by the movement. Routine *detour()* re-routes a segment in a very limited

nearby region on current or another routing layer. Step 2 enables most of the adjusted nets to evade those locally congested routing regions without changing their topologies. If there still exists a few nets with crosstalk violation ($X > X_l = 0$) after step 1 and 2 are carried out, CR algorithm goes to step 3 where the global/local net adjustings are alternatively applied. The algorithm terminates when the final routing is crosstalk-free or a CPU-limit exceeds (detected by routine $timeout()$). Usually, when the step 1 and 2 are finished, *xtalk* X is equal to zero or a small number, so that the step 3 can complete the final routing very fast.

CR Algorithm

Input: Initial routing solution with *xtalk* X_i
Output: Final routing solution with *xtalk* $X_l = 0$

$X = X_i$
Step 1. /* global net adjusting */
  while $X > X_g$ and $timeout()$ is not true:
    rip up all the nets with crosstalk violation
    re-route the ripped-up nets using SOAR
    $X = estimateCrosstalk()$
Step 2. /* local net adjusting */
  for each net with crosstalk violation :
    if $changeLayer()$ fails then $detour()$
  $X = estimateCrosstalk()$
Step 3. /* alternative net adjusting */
  While $X > X_l$ and $timeout()$ is not true:
    Repeat Step 1 & 2

Fig. 4. Crosstalk Reduction Algorithm

5 Experiments

The performance-driven MCM router MRC has been implemented in C programming language, and all experiments are performed on a DEC Station 5000/125.

Four MCM benchmarks, spert, MCC1-75, MCC2-75, and MCC2-45, are used for the experiments. Benchmark spert is an MCM consisting of a vector processor (ASIC), 16 SRAMs and 3 buffer components. All other MCM benchmarks are industrial routing examples provided by MCC. MCC2-75 and MCC2-45 are generated from same MCC benchmark by using different routing pitch, 75 μm and 45 μm, respectively. Table 1 lists the main characteristics of the benchmarks. Where #C, G_S and S_S are the number of chips, grid size and substrate size (in mm^2) of each MCM, respectively.

The experimental results of MRC are shown in Table 2. As shown, we have obtained the reasonable maximum sink delays for all benchmarks tested. The *xtalk* obtained at different routing steps for each test are used to demonstrate MRC's ability of handling crosstalk constraint. "X_n" is the crosstalk violation generated by routing without consideration of crosstalk constraint. X_i, X_g, and X_l have the same meanings as those in section 4.3. *L, Vis, WL*, and D denote the number of layers used to complete the routing, the number of vias, the wire length, and the maximum sink delay, respectively. In Table 2, for all cases, $X_l = 0$, which means the final routing solutions are crosstalk-free.

Table 1. MCM Benchmark Specifications

	spert	MCC1-75	MCC2-75	MCC2-45
#C	20	6	37	37
Net	248	802	7118	7118
Pin	1168	2496	14659	14659
G_S	2138x2119	599x599	2032x2032	3386x3386
S_S	85.5x84.8	45x45	152x152	152x152

Table 2. Exprimental Results of MRC

	spert	MCC1-75	MCC2-75	MCC2-45
L	4	6	8	6
Via	1466	3151	22939	18243
WL	540828	428822	6169959	9522002
D(ns)	19.196	0.190	0.157	0.215
X_n	20.56%	71.84%	72.44%	50.51%
X_i	8.47%	60.45%	46.22%	27.10%
X_g	2.82%	32.54%	25.19%	8.92%
X_l	0.0	0.0	0.0	0.0
cpu(s)	15	551	3422	1868

Results have previously been reported by V4R [1], MCG [2], and V4C [3] for all the MCC benchmarks. The performance comparisons of MRC with V4C, MCG and V4R are shown in Table 3 and 4. As shown, MRC uses fewest vias for all cases. There may be two reasons for this. First, MRC routes each multi-pin net into a complete Steiner tree instead of splitting it into a set of two-pin nets. This, general speaking, leads fewer vias and shorter wire length because the net splitting may result in some detour from the point of view of the complete multi-pin net routing. Second, by assigning a reasonably large β value to the (α, β, γ) routing process, router MRC may use fewer vias to complete the routing. On the other hand, MRC routes the MCC benchmarks with shortest wire length and use the same number of layers with V4R for all cases without considering crosstalk constraint (see those rows indicated by "w/o").

Comparing to V4C, MRC has more powerful ability of handling crosstalk problem while the other performances are almost same except for using fewer vias of MRC. As shown in those rows indicated by "w" of Table 4, both MRC and V4C have achieved the final crosstalk-free routing for all cases. However, the crosstalk estimation methods used by MRC and V4C are different. MRC estimates the crosstalk of a net based on the *sum of couplings* between the net and all the other neighboring nets as shown in equation (1) in section 2, while V4C is based on the *maximum coupling* [3]. General speaking, the former estimation method is more adequate than the latter one. The reason is that both crosstalk models are simply defined based on the coupling length of the neighboring nets. In this case, any coupling should have the contribution to crosstalk noise rather than only "maximum coupling" unless the length of "maximum coupling" is competitive with that of "sum of coupling".

For all the benchmarks tested in this paper, crosstalk reduction algorithm has successfully reduced

crosstalk X from the large value X_i to $X_l = 0$. Of course, crosstalk reduction algorithm may result in longer wire length, more vias. In our experiments, the increase in wire length and vias is not larger than 27.0% and 15.6%, respectively (see each row of "MRC" in Table 3).

Table 3. Performance Comparisons I

MCCs	Methods		L	Via	WL
MCC1-75	MRC	w/o	4	2481	370886
		w	6	3151	428822
	V4C	w/o	4	6757	381592
		w	6	6957	381270
	V4R	w/o	4	6993	394272
	MCG	w/o	4	5747	378707
MCC2-75	MRC	w/o	6	19041	5434400
		w	8	22939	6169959
	V4C	w/o	6	33974	5429010
		w	8	34621	5433718
	V4R	w/o	6	36438	5559479
	MCG	w/o	4	34311	5695039
MCC2-45	MRC	w/o	4	21665	9050509
		w	6	18234	9522002
	V4C	w/o	4	34026	9039996
		w	6	34600	9045298
	V4R	w/o	4	36473	9130705

Table 4. Performance Comparisons II

MCCs	Methods		*xtalk*	CPU(s)
MCC1-75	MRC	w/o	71.84%	376
		w	0.0	551
	V4C	w/o	34.3%	38
		w	0.0	63
	V4R	w/o	-	180
	MCG	w/o	-	540
MCC2-75	MRC	w/o	72.44%	3079
		w	0.0	3422
	V4C	w/o	49.8%	714
		w	0.0	1714
	V4R	w/o	-	3960
	MCG	w/o	-	13020
MCC2-45	MRC	w/o	50.51%	2697
		w	0.0	1868
	V4C	w/o	47.0%	708
		w	0.0	1782
	V4R	w/o	-	5820

6 Conclusions

A new performance-driven MCM router MRC with special consideration of crosstalk reduction has been proposed in this paper. Router MRC has been tested on several industrial benchmarks, and shown to have some unique features. First, the MRC exactly obtains the time delay of all benchmarks tested based on Elmore delay model. Second, comparing with the best known MCM routing approaches, MRC uses fewer vias for all the benchmarks tested. Third, MRC has successfully obtained the final crosstalk-free routing solutions with an adequate performance trade-off among multiple routing objectives. Finally, we claim that, although only tested on MCM benchmarks, the idea and algorithms proposed in this paper can also be used in IC layout.

As the future work, the crosstalk model may be modified at two points. First, the coupling length and the sink delay are simultaneously taken into consideration of the model because the allowable coupling length also depends on timing slacks between the required arrival time and the real arrival time of each sink. Second, each net's coupling to net n_i is given a priority according to its coupling length. The nets with longer coupling length with n_i is assigned a higher priority. The net with highest priority is first ripped up and re-roouted. This is helpful to reduce the worse couplings that have higher single coupling.

Acknowledgements

The authors wish to acknowledge the support of SRC and NSF under the grants 95-D C-324 and MIP-9117328.

References

[1] Kei-Yong Khoo, Jason Cong, "An Efficient Multilayer MCM Router Based on Four-Via Routing", *30th DAC*, pp. 590-595, 1993.

[2] J.D. Carothers, D. Li, "A Multilayer MCM Auto-router Based on the Correct-by-Design Approach", *Proc. 8th Annual IEEE Intl. ASIC Conf. and Exhibit*, pp.139-142, Sept. 1995.

[3] T. Miyoshi, S. Wakabayashi, T. Koide, N. Yoshida, "An MCM Routing Algorithm Considering Crosstalk", *IEEE ISCAS'95*, pp.211-214, 1995.

[4] J.D. Cho, K.F. Liao, S. Raje, M. Sarrafzadeh, "M^2R: Multilayer Routing Algorithm for High-Performance MCMs", *IEEE Trans. on CAS-I*, Vol.41, N0.4, pp.253-265, 1994.

[5] D.S. Wang, E.S. Kuh, "A New Timing-Driven Multilayer MCM/IC Routing Algorithm", *Proc. MCMC'97*, pp.89-94, 1997.

[6] Kei-Yong Khoo, Jason Cong, "A Fast Multilayer General Area Router for MCM Design", *IEEE Trans. on CAS-II*, Vol. 39, No. 11, Nov. 1992.

[7] Q. Yu, S. Badida, N. Sherwani, "Algorithmic Aspects of Three Dimensional MCM Routing", *31st DAC*, pp. 397-401, 1994.

[8] D. Staepelaere, J. Jue, T. Dayan, W.W.W. Dai, "Surf: A Rubber-Band Routing System for Multichip Modules", *IEEE Design and Test of Computers*, Vol.10, pp.18-26, Dec. 1993.

[9] T.C. Hu, M.T. Shing, "The $\alpha - \beta$ Routing", *in VLSI Circuit Layout: Theory and Design, Ed. T.C. Hu and Ernest S. Kuh, New York*, 1985, IEEE Press, pp. 139-143.

[10] K.D.Boese, A.B.Kahng, B.A.McCoy, G.Robins, "Rectilinear Steiner Trees with Minimum Elmore Delay", *31st DAC*, pp. 381-386, 1994.