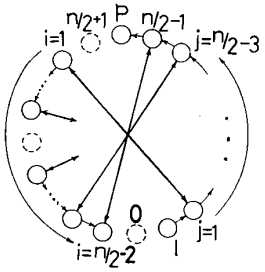


Fig. 4. A double-loop network in triple-node failure in Case II) where $n = 2p$.

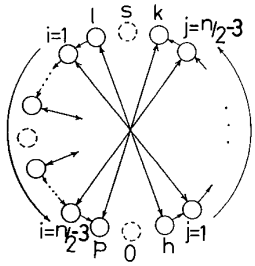


Fig. 5. A double-loop network in triple-node failure in Case III) where $n = 2p$ and the distance between a faulty node pair is s.

nodes 0 and $p + 1$ are assumed to be faulty. Then we have

$$E_1 = \sum_{i=1}^{p-2} (n-2(i+1)+2i) + \sum_{j=1}^{p-3} (n-2(j+1)+2j) + 2(n-3)$$

$$= n^2 - 5n + 4.$$

Let E_2 denote the number of communicable node pairs in a double-node failure with $D_f = s - 1$ or $n - s + 1$. Then we get $E_2 = n^2/2 - 2n + 1$ from Table I. Let E denote the total number of communicable node pairs for all triple-node failures where two of three are fixed at nodes 0 and $p + 1$. Then we have

$$E = (n-3)E_2 - E_1$$

$$= (n^3 - 9n^2 + 24n - 14)/2.$$

There exist two types of networks in Case III): there exists some pair with $D_f = s$ or none. We first investigate the case of $D_f = s$. Fig. 5 shows an example of these networks where faulty nodes are 0, s, and i (or j). In order to make i (or j) such that the distance between any two faulty nodes is not $s + 1$ or $n - s - 1$, we set i (or j) $\neq h, k, l, p$. Therefore, the number of triple-node failures that two faulty nodes are fixed at nodes 0 and s is $n - 6$. When only two nodes 0 and s are assumed to be faulty, the number of communicable node pairs $(i, \cdot)$ or $(j, \cdot)$ is, respectively, $n - (2i + 3)$ or $n - (2j + 3)$. And then the number of communicable node pairs $(\cdot, i)$ or $(\cdot, j)$ is, respectively, $2i + 1$ or $2j + 1$. Let F_1 denote the total number of communicable node pairs for all i and j. Then we have

$$F_1 = 2 \sum_{i=1}^{p-3} (n-(2i+3)+(2i+1))$$

$$= n^2 - 8n + 12.$$

We can see that F_2 (the number of communicable node pairs in a double-node failure with $D_f = s$) is $n^2/2 - 2n + 2$ from Table I. Let F denote the total number of communicable node pairs in a triple-node failure when two faulty nodes are fixed at nodes 0 and s. Then we get the following expression:

F = (the number of communicable node pairs when only 0 and s are faulty)
− (the number of communicable node pairs whose counterpart is node i or j where i or j is the third faulty node candidate)
$= (n - 6)F_2 - F_1$
$= n^3/2 - 6n^2 + 22n - 24.$

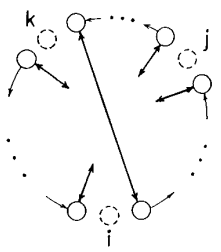


Fig. 6. A double-loop network in triple-node failure where the distance between any faulty node pair is not s.

The network shown in Fig. 6 is such that the distance between any faulty node pair is not s. In this case, we can easily obtain the number of communicable node pairs in a double-node failure, i.e., ${}_{2n-3}C_2$.

We can calculate the numbers of communicable node pairs in other different cases in the similar manner as mentioned above.

References

[1] B. W. Arden and H. Lee, "Analysis of chordal ring," *IEEE Trans. Comput.*, vol. C-30, pp. 291–295, Apr. 1981.

[2] D. K. Pradhan and S. M. Reddy, "A fault-tolerant communication architecture for distributed systems," *IEEE Trans. Comput.*, vol. C-31, pp. 863–870, Oct. 1982.

[3] M. T. Liu, "Distributed loop computer networks," in *Advances in Computers, Vol. 17.* New York: Academic, 1978, pp. 163–221.

[4] A. Grnarov *et al.*, "A highly reliable distributed loop network architecture," in *Proc. Int. Symp. Fault-Tolerant Comput.*, Kyoto, Japan, Oct. 1980, pp. 319–324.

[5] C. S. Raghavendra *et al.*, "Reliable loop topologies for large local computer networks," *IEEE Trans. Comput.*, vol. C-34, pp. 46–55, Jan. 1985.

[6] H. Masuyama, "Fault-tolerant ability to double-node failures in double-loop computer networks," *Trans. IECE Japan*, vol. E69, pp. 597–600, May 1986.

A Fast and Flexible Thinning Algorithm

P. S. P. WANG AND Y. Y. ZHANG

Abstract—**A fast serial and parallel algorithm for thinning digital patterns is presented. The processing speed is faster than the algorithms in the literature [1]–[3] in that it reads pixels along the edge of the input pattern rather than all pixels in each iteration. Using this algorithm, an experiment is conducted and the patterns such as "X," "H," "A," "moving body," and "leaf" are tested. The results show that this**

Manuscript received July 25, 1986; revised March 10, 1987 and October 10, 1987.

The authors are with the College of Computer Science, Northeastern University, Boston, MA 02115.

IEEE Log Number 8824551.

p[7]:q[i-1,j-1]	p[0]:q[i-1,j]	p[1]:q[i-1,j+1]
p[6]: q[i,j-1]	p : q[i,j]	p[2]: q[i,j+1]
p[5]:q[i+1,j-1]	p[4]:q[i+1,j]	p[3]:q[i+1,j+1]

Fig. 1. Pixel and its neighbors.

algorithm is faster, structure-preserving, and more flexible in that it can be done either sequentially or in parallel.

Index Terms—**Image processing, parallel thinning, pattern recognition, skeleton.**

I. Introduction

In dealing with image processing and pattern recognition problems, a digitized binary pattern is normally defined by a matrix, where each element, called a pixel, is either 1 (dark point) or 0 (white point). "Thinning" is a process that deletes the dark points and transforms the pattern into a "thin" line drawing known as a skeleton. The thinned pattern must preserve the basic structure of the original pattern and the connectedness. "Thinning" plays an important role in digital image processing and pattern recognition, since the outcome of thinning can largely determine the effectiveness and efficiency of extracting the distinctive features from the patterns [12], [13].

A variety of thinning algorithms [1]–[11], [14], [15], such as serial, parallel, and maximum methods, have been proposed for thinning digital patterns. In a serial method, the value of a pixel at the nth iteration depends on a set of pixels for some of which the result of nth iteration is already known. In parallel processing, the value of a pixel at the nth iteration depends on the values of the pixel and its neighbors at the $(n-1)$th iteration. Thus, all the pixels of the digital pattern can be thinned simultaneously.

In the literature [4], a fast parallel algorithm for thinning digital patterns was proposed. This algorithm uses a "parallel" approach so that the thinning process is very good with respect to both parallelism and connectivity. However, there are some disadvantages, such as erosion of diagonal patterns. In the literature [2], the pixel is flagged when it should be deleted at an iteration. So it is not a parallel thinning algorithm. In this paper, we propose a new serial and parallel algorithm which is faster and structure-preserving.

II. Basic Definitions and Notations

In order to describe the algorithm in this paper, we give some definitions as follows.

Definition 1: The neighbors of a pixel, $p{:}q[i,j]$, are identified by the eight directions, $q[i-1,j]$, $q[i-1,j+1]$, $q[i,j+1]$, $q[i+1,j+1]$, $q[i+1,j]$, $q[i+1,j-1]$ $q[i,j-1]$, $q[i-1,j-1]$, i.e. $p[k]$, $(k = 0 \cdots 8, p[8] = p[0])$, shown in Fig. 1.

Definition 2: The contour points of a digital pattern are defined as those pixels for which at least one neighbor is white. In Fig. 2, "a," "b," $\cdots$, "s" are contour points, "t" is not contour point.

Definition 3: The contour loop is a set of contour points which are connected into a loop. We use $L(1), \cdots, L(m)$ to label the different contour loops of a pattern, where m is the number of contour loops. For example, the letter "Y" (Fig. 2) is one contour loop pattern ($m = 1$). The letter "A" (Appendix c) is a two-contour loop pattern ($m = 2$).

Definition 4: When a contour point u at a time is processed, the next processing contour point v is called a successor contour point of u and u is called a previous contour point of v. In Fig. 2, if the pixel "b" is processed, then the point "a" is the previous contour point of "b" and "c" is the successor contour point of "b."

Definition 5: Let p be the processing contour point. Let x be the previous contour point of p, and let z be the successor contour point of p. Then s will be called successor function if $z = s(x, p)$ is a function from x according to clockwise order around the neighbors of p to meet the first dark point z. In Fig. 2, if the current processing pixel is "7" and the previous contour point of "7" is "6," then $z = s$("7," "6") is "j." If the previous contour point of "7" is "r," then z is "3."

```
a b       e f
s 1 c     4 g
  2 d   5 h
    3 6 i
      7
    r 8 j
    q 9 k
    p t l
    o n m
```

Fig. 2. The digit pattern "Y."

III. Serial Thinning Algorithm

In serial processing, the result of thinning a point at the nth iteration depends on a set of points for some of which the result of the nth iteration is already known. Naccache and Shinghal gave in 1984 a serial thinning algorithm [2]. In this paper, we also give a serial thinning algorithm which is faster than algorithm [2]. Our algorithm stores the original picture in a matrix Q. All processes in the following algorithm are operated on matrix Q. To describe this algorithm, several specific functions need to be provided.

1) *Initial*: a function that computes: a) the contour loop number m, b) the first contour point FIRST$[k]$, and its previous point, PREV$[k]$, for each contour loop $L[k]$, $(k = 1, 2, \cdots, m)$, c) set the initial value "1" to loop decision variable $h[k]$ ($h[k] = 1$ means that the contour loop points need to be thinned).

2) *loop-end-test* (k): a function that tests whether the kth contour loop is terminated.

3) *successor* (x, p): a function that computes the successor point from previous point x according to clockwise order around the neighbors of p to meet the first dark point.

4) *deletion* (Q, p): a function that deletes a point p in Q if it satisfies following two conditions: a) $1 < B(p) < 7$, b) $A(p) = 1$ or $c(p) = 1$.

Where $B(p)$ is the number of neighbors of p which are not equal to zero. $A(p)$ is the number of white-to-dark transitions when the neighbors are taking a clockwise walk around p (i.e., along the neighbors of pixel p).

$$c(p) = \begin{cases} 1 & p[k-1]+p[k]+p[k+1]+p[k+4]=0 \\ & \text{and } p[k+3]=1 \text{ and } p[k+5]=1 \ (k=1 \text{ or } 3) \\ 0 & \text{otherwise.} \end{cases}$$

Function: serial thinning

Arguments:

initial—a function that counts m FIRST$[k]$, PREV$[k]$, and sets 1 to $h[k]$ and $g(k = 1, 2, \cdots, m)$.

loop-end-test (k)—a function that returns t only endpoint of kth contour loop.

successor (x, p)—a function that returns the successor point.

deletion (Q, p)—a function that deletes p in Q.

Algorithm:

initial;

repeat

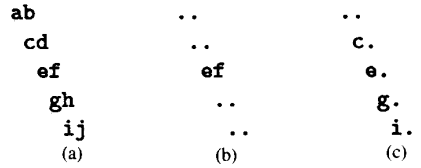


Fig. 3. A slanting pattern and its thinning.

TABLE I
THE COMPARISON OF RUN TIME (IN SECONDS) CONSUMED BY DIFFERENT SERIAL THINNING ALGORITHMS

Algorithm	Pattern			
	"H"	"A"	"body"	"leaf"
Our Algorithm	7.80	8.57	10.60	13.68
Naccache & Shinghal	12.91	11.25	17.03	32.57

```
for k := 1 to m do if h[k] = 1 then
begin
  p := FIRST[k]; x := PREV[k]; h[k] := 0;
  repeat
    z := successor (x, p); x := p;
    if 1 < B(p) < 7 and (A(p) = 1 or c(p) = 1)
      then begin deletion (Q, p); h[k] := 1 end;
    p := z
  until loop-end-test (k);
end;
until h[1] + ··· + h[m] = 0.
```

IV. Parallel Thinning Algorithm

In parallel processing, the value of a pixel at the nth iteration depends on the values of the pixel and its neighbors at the $(n-1)$th iteration.

In paper [3], Zhang and Suen proposed a parallel thinning algorithm ZS, which yields good results with respect to both connectivity and contour noise immunity, but algorithm ZS sometimes no longer preserves the structure of the original pattern. In Fig. 3(a) is the original pattern. Fig. 3(b) is the skeleton using algorithm ZS and Fig. 3(c) is the skeleton using our parallel thinning algorithm.

Our parallel thinning algorithm can preserve the structure of the original picture and the speed is faster than that of Algorithm ZS. We store the picture in matrix Q and $Q1$. The algorithm is shown as follows.

Function: parallel thinning

Arguments:
initial—a function that computes m, FIRST[k], PREV[k], and sets 1 to $h[k]$ ($k = 1, 2, \cdots, m$).
loop-end-test (k)—a function that returns t only endpoint of kth contour loop.
successor (x, p)—a function that returns the successor point.
deletion (Q, p)—a function that deletes p in matrix Q if p satisfies the following condition.

$1 < B(p) < 7$ and $(A(p) = 1$ or $c(p) = 1)$ and $(p[2] + p[4] * p[0] * p[6] = 0)$

or

$1 < B(p) < 7$ and $(A(p) = 1$ or $c(p) = 1)$ and $(p[0] + p[6] * p[2] * p[4] = 0)$

Algorithm:

```
initial; g := 1;
repeat
  Q := Q1; if g = 1 then g := 0 else g := 1;
  for k := 1 to m do if h[k] = 1 then
  begin p := FIRST[k]; x := PREV[k]; h[k] := 0;
    repeat
      z := successor (x, p); x := p;
      case g of
        0: if 1 < B(p) < 7 and (A(p) = 1 or c(p) = 1)
                           and (p[2] + p[4]) * p[0] * p[6]
                           = 0
             then begin deletion (Q1, p); h[k] := 1 end;
        1: if 1 < B(p) < 7 and (A(p) = 1 or c(p) = 1)
                           and (p[0] + p[6]) * p[2] * p[4]
                           = 0
             then begin deletion (Q1, p); h[k] := 1 end;
      end case;
      p := z;
    until loop-end-test (k)
  end
until h[1] + ··· + h[m] = 0.
```

V. The Result of Experiment

To compare the performance of our algorithms to Algorithms NS[2], ZS[3], and SV[4], all algorithms were coded in Pascal and run on an IBM personal computer XT. These algorithms were tested on digital patterns "H," "A," "body," and "leaf." The units of time taken by these algorithms are shown in Table I (serial thinning algorithms) and in Table II (parallel thinning algorithms).

VI. Discussion and Conclusions

We have just proposed the thinning algorithms which work along the contours of patterns. Comparing our algorithms to a couple of algorithms from the literature [2]–[4], we find the following.

1) They are faster, in that they do not require each pixel of the pattern to be tested except contours. As we know, "thinning" is a process that tests a dark pixel to see if it satisfies thinning conditions. If it does, delete this pixel, otherwise leave it as it is. Therefore, thinning spends most of the time on testing thinning conditions. It is faster since our proposed algorithms only need to test contours.

For example, a 556-pixel digitized character "H" shown in Fig. 4 adopted from [3]. The points marked by "." have been removed. Fig. 4(a) shows the result after the first iteration. The final result is shown in Fig. 4(b). In the first iteration, Algorithm ZS needs to test 556 pixels. In order to obtain the final result, Fig. 4(b), Algorithm ZS needs to test total 2552 pixels (556 + 481 + 409 + 339 + 271 + 205 + 141 + 96 + 54). However, our new algorithms only need to test 146 pixels at the first iteration and the total tested pixels is 1170 (146 + 142 + 138 + 134 + 130 + 126 + 122 + 118 + 114).

TABLE II
THE COMPARISON OF RUN TIME (IN SECONDS) CONSUMED BY DIFFERENT PARALLEL THINNING ALGORITHMS

Algorithm	Pattern			
	"H"	"A"	"body"	"leaf"
Our Algorithm	9.56	10.43	14.67	22.35
Zhang & Suen	11.59	10.65	14.72	26.64
Rutovitz	16.53	14.11	21.59	42.08

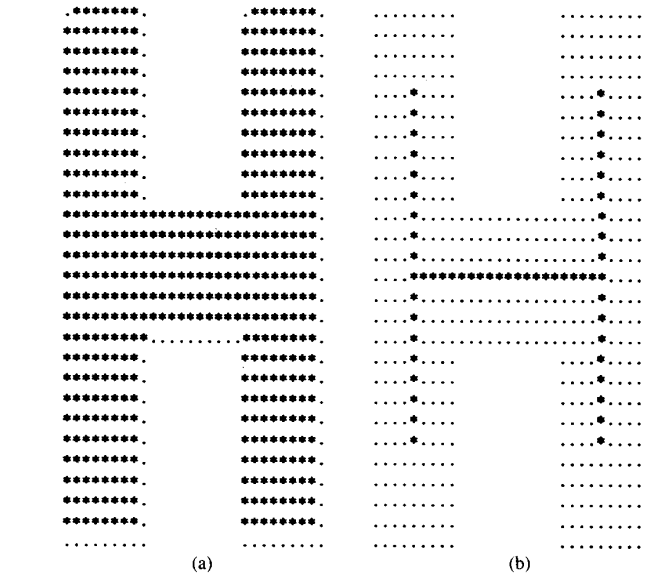


Fig. 4. Character "H" and its skeleton.

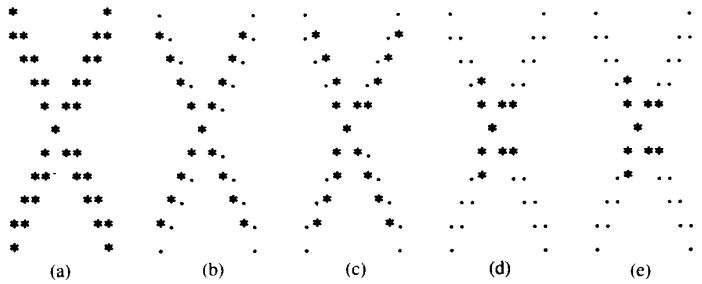


Fig. 5. Comparing different algorithms for thinning pattern "X."

2) It is structure-preserving. The simple version algorithm (SV) presented by Rutovitz [4] is a parallel algorithm. Zhang and Suen's (ZS) algorithm is also parallel. However, both of them have to overcome the disadvantage of not preserving structures. For example, using Algorithms ZS and SV to thin the digital pattern "X," the results are shown in Fig. 5(d) and (e) which lose the structures of the original pattern. (Fig. 5(a) is the original pattern, 5(b) is the result thinned by our algorithm 3 which preserves the structure). The algorithm NS can keep the shape of the original pattern as shown in Fig. 5(c), but sometimes it cannot exactly keep the basic structure as shown in Fig. 6, and it is not a parallel algorithm as we defined above. In this paper, the algorithms we presented can keep the structure of the original pattern.

For future research, it would be interesting and worthwhile to explore thinning algorithms for high-dimensional patterns. Realizing the advantages of our approach, it would also be very interesting to polish, implement, and apply this serial and parallel thinning algorithm to a variety of pattern recognition problems including: character recognition, fingerprint recognition, signature verification, medical diagnosis, business application, industrial parts inspection, computer vision, and robotics.

However, there are still some disadvantages: 1) if a spur of two dots were added to the left end of the middle row of Appendix d, the noise will propagate. This shares the same drawback as discussed in [1]–[3], [11], and [15], and 2) even worse, the same digital patterns will totally disappear as shown in literature [1].

ACKNOWLEDGMENT

The authors want to show appreciation to the reviewers for their very helpful comments and suggestions.

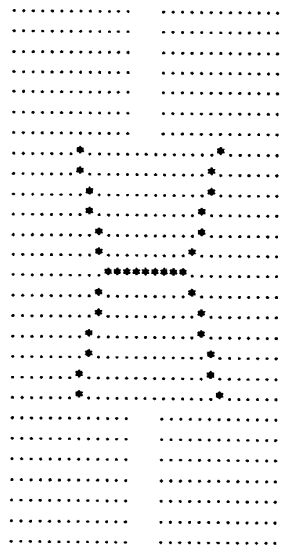
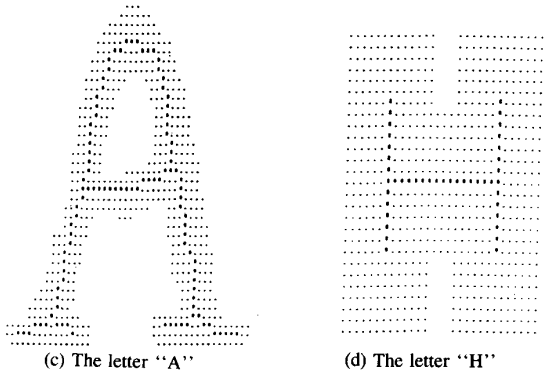


Fig. 6. Thinning of the character "H" using the algorithm NS.

APPENDIX

THINNING OF DIFFERENT DIGITAL PATTERNS USING ALGORITHM 3.

(a) A moving body (b) A leaf



(c) The letter "A" (d) The letter "H"

REFERENCES

[1] H. E. Lu and P. S. P. Wang, "A comment on 'A fast parallel algorithm for thinning digitial patterns.'" *Commun. ACM*, vol. 29, pp. 239-242, Mar. 1986.

[2] N. J. Naccache and R. Shinghal, "An investigation into the skeletonization approach of Hilditch," *Pattern Recognition*, vol. 17, no. 3, pp. 274-284, 1984.

[3] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, pp. 236-239, 1984.

[4] D. Rutovitz, "Pattern recognition," *J. Royal Statist. Soc.*, vol. 129, pp. 504-530, 1966.

[5] C. Arcelli, L. P. Cordella, and S. Levialdi, "From local maxima to connected skeletons," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 134-143, 1981.

[6] C. J. Hilditch, "Linear skeletons from square cupboards," *Machine Intell.*, vol. 4, pp. 403-420, 1969.

[7] T. Pavlidis, "A flexible parallel thinning algorithm," in *Proc. IEEE Comput. Soc. Conf. Pattern Recognition, Image Processing*, 1981, pp. 162-167.

[8] M. C. Rahier and P. G. A. Jespers, "Dedicated LSI for a microprocessor controlled hand carried OCR system," *IEEE J. Solid-State Circuits*, vol. SC-15, pp. 15-24, 1980.

[9] A. Rosenfeld, "Connectivity in digital pictures," *J. ACM*, vol. 17, pp. 146-160, 1971.

[10] ——, "A characterization of parallel thinning algorithm," *Inform. Contr.*, vol. 29, pp. 286-291, 1975.

[11] R. Stefanelli and A. Rosenfeld, "Some parallel thinning algorithms for digital pictures," *J. ACM*, vol. 18, pp. 255-264, 1971.

[12] P. S. P. Wang, "An application of array grammars to clustering analysis for syntactic patterns," *Pattern Recognition*, vol. 17, pp. 441-451, 1984.

[13] P. S. P. Wang, Ed., *Intelligent System Imaging Technology and Software Engineering*. Sung Kang Computer Book Co., 1984.

[14] R. Stefanelli, "A comment on an investigation into the skeletonization approach of Hilditch," *Pattern Recognition*, vol. 19, pp. 13-14, 1986.

[15] P. S. P. Wang and Y. Y. Zhang, "A fast serial and parallel thinning algorithm," in *Proc. 8th Euro. Meet. Cybern. Syst. Res.*, Apr. 1-4, 1986, Wien, Austria.

Hamiltonian Cycles in the Shuffle-Exchange Network

WENTAI LIU, THOMAS H. HILDEBRANDT, AND RALPH CAVIN, III

Abstract—**The usefulness of the shuffle-exchange network in parallel processing applications is well established. The optimal embedding of a shuffle-exchange network of a given size depends upon the number of cycles of the shuffle permutation of that size. The cost of one method of adding fault-tolerance through reconfigurability depends upon the number of such cycles, and the manner in which they can be connected to form larger cycles.**

This paper presents an exact equation for the number of cycles of a shuffle of size 2^n. We use that result to demonstrate that it is always possible to form a Hamiltonian cycle on all processors in a shuffle-exchange connected array. From this, it is apparent that there are a large number of ways of sharing spare processors among the members of many

Manuscript received July 26, 1986; revised October 15, 1986 and February 25, 1988. This work has been supported in part by Grant FAS-5-30406 from the Microelectronics Center of North Carolina, the University of North Carolina System-Bell Northern Research Award for the promising young faculty, and Semiconductor Research Corporation SRC-86-DJ-090.

W. Liu and T. H. Hildebrandt are with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695.

R. Calvin, III is with Semiconductor Research Corporation, Research Triangle Park, NC 27709.

IEEE Log Number 8824549.