The Design of Two Easily-Testable VLSI Array Multipliers

Joel Ferguson and John Paul Shen

Department of Electrical Engineering
Carnegie-Mellon University
Schenley Park, Pittsburgh PA 15213 U.S.A.

ABSTRACT

Array multipliers are well-suited for VLSI implementation because of the regularity in their iterative structure. However, most VLSI circuits are very difficult to test. This paper shows that, with appropriate cell design, array multipliers can be designed to be very easily-testable. An array multiplier is called C-testable if all its adder cells can be exhaustively tested while requiring only a constant number of test patterns. The testability of two well-known array multiplier structures are studied. The conventional design of the carry-save array multiplier is shown to be not C-testable. However, a modified design, using a modified adder cell, is generated and shown to be C-testable and requires only 16 test patterns. Similar results are obtained for the Baugh-Wooley two's complement array multiplier. A modified design of the Baugh-Wooley array multiplier is shown to be C-testable and requires 55 test patterns. The implementation of a practical C-testable 16 x 16 array multiplier is also presented.

1. INTRODUCTION

In recent years, it has become feasible to implement an array multiplier of reasonable size on one very large scale integrated (VLSI) circuit chip [1,2]. In general, VLSI circuits are very difficult to test for several reasons. The high device to pin ratio severely limits the controllability and observability of a VLSI circuit chip. There exists a large number and types of faults, many of which cannot be modeled by the traditional stuck-at-0/1 fault model. Test pattern generation and verification procedures are becoming very costly or even computationally infeasible to implement [3]. However, this paper shows that, because of their regular iterative structure, VLSI array multipliers can be designed to be very easily-testable.

Previous works have shown that some regularly-structured circuits are very easily testable. An iterative logic array (ILA) is a logic circuit consisting of a regular array of identical cells. Kautz was the first researcher to investigate the testing of one and two dimensional ILAs [4]. He characterized the necessary and sufficient conditions for exhaustively testing all cells in a general ILA. Friedman has shown that certain one-dimensional ILAs can be fully tested using a constant number of test patterns, i.e. the number of test patterns needed is independent of the size of the array [5]. He called these C-testable ILAs. An example of a C-testable ILA is the n-bit parallel adder consists of a linear array of n full adders [5]. The n-bit parallel adder can be fully tested using only eight test patterns regardless of the size of the adder. Others have also investigated the testing of ILAs. Parthasarathy and Reddy [6] developed the concepts of one-step testability and one-step C-testability. Sridhar and Hayes [7] have applied the concept of C-testability to the testing of bit-sliced microprocessors. Other than the original work by Kautz, most studies on easily-testable ILAs only focus on one-dimensional arrays. In this paper, the concept of C-testability is applied to the analysis of the testability of practical two-dimensional multiplier arrays.

Section 2 outlines the fault model and testing approach used in this paper and characterizes the faulty and fault propagation behavior of the basic cell. Section 3 presents a convenient notation for describing the structures of array multipliers and introduces the carry-save array multiplier. It is shown that the conventional design of the carry-save array multiplier is not C-testable. However, a modified design of the carry-save array multiplier can be obtained and is shown to be C-testable. Section 4 documents similar results on the Baugh-Wooley two's complement array multiplier. A C-testable 16 x 16 carry-save array multiplier has been designed. Its layout has been generated and can be implemented on a NMOS chip. Details of the design and possible extensions to this research are discussed in Section 5.

2. TESTING APPROACH

The basic building block, or cell, used in most array multipliers is a full adder with a 2-input AND gate (for product bit generation) connected to one of its three inputs. The diagram and the logic equations of the basic cell is illustrated in Figure 1. Each cell has four inputs, labeled a, b, c and d, and two outputs labeled x and y corresponding to the sum and the carry outputs, respectively, of the full adder.

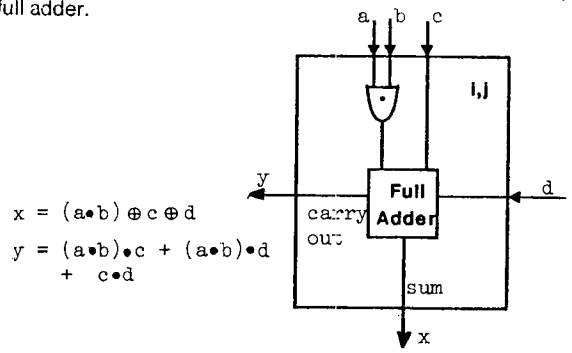

Fig. 1. The basic full adder cell used in array multipliers.

The principal aim of the testing approach proposed in this paper is to exhaustively test each cell in the array without exhaustively testing the entire array. The exhaustive testing of the entire array, as has been the practice to date, is becoming impractical. It is believed that by taking advantage of the iterative structure in an array multiplier, all the cells can be simultaneously tested. The fault model used in this paper is similar to that used by previous researchers [4,5,7]. The fault model assumes: (1) In an array multiplier, at most one basic cell is faulty at a time. (2) The fault is a

permanent fault. (3) The fault may alter the cell's output functions in any arbitrary way, as long as the faulty cell remains a combinational circuit. With the above assumptions, the testing of an array multiplier must involve the exhaustive testing of every cell by applying all possible input patterns and observing all output patterns. In effect, the truth table of every cell must be verified.

Definition 1: An array multiplier is testable if the following two conditions are met:

1. For each cell in the array, all possible (2^4) cell input vectors can be applied to the cell.

2. For each cell input vector applied to a cell under test any faulty signal produced at the outputs of the cell can be propagated to a primary output of the array.

Condition 1 is necessary for the exhaustive testing of every cell. Condition 2 ensures that as each cell is being exhaustively tested any faulty signal at the cell output(s) is observable at the array outputs. The concept of C-testability can be extended to the testing of array multipliers.

Definition 2: An array multiplier is C-testable if it is testable and the number of array input vectors, or test patterns, required is a constant and independent of the size of the array multiplier.

Since each basic cell has four inputs, a cell input vector can be represented by a binary 4-tuple $\langle a,b,c,d\rangle$. v_k can be used to denote the input vector whose binary 4-tuple is the binary representation of k. For example the cell input vector $\langle a,b,c,d\rangle = \langle 0,1,0,1\rangle$ is denoted as v_5. In order to satisfy condition 1 of Definition 1, $v_0, v_1, \ldots, v_{15}$ must be applied to every cell in the array.

Based on the fault model defined earlier the effects of a fault in a cell on its output signals can be easily characterized. Let $\langle a,b,c,d\rangle$ be an input vector to a cell that produces the output vector $\langle x,y\rangle$. If the cell is faulty, then the possible faulty output vectors are: $\langle x',y\rangle$, $\langle x,y'\rangle$ and $\langle x',y'\rangle$, where x' and y' are the complements of x and y respectively. Hence, given a faulty cell and a cell input vector that produces faulty outputs, then either one of the two output signals is inverted or both are inverted.

Lemma 1: Given a cell input vector $\langle a,b,c,d\rangle$ that produces the output vector $\langle x,y\rangle$. The cell input vectors $\langle a,b,c',d\rangle$ and $\langle a,b,c,d'\rangle$ will both produce output vectors of the form $\langle x',y^*\rangle$, where x' is the complement of x and y* is either y or y'.

Proof: The x output of a cell is defined as the exclusive-OR of three terms, namely a b, c and d, i.e. $x = (a \bullet b) \oplus c \oplus d$. Clearly, x is the parity of the three terms and if either the c or the d input is inverted then the output x will also be inverted. Δ

The above lemma implies that if a faulty signal appears at only the c or the d input of a fault-free cell then the faulty behavior will be propagated to the x output of the cell. This lemma will be used in the following sections to prove the C-testability of two array multipliers.

3. THE CARRY-SAVE ARRAY MULTIPLIER

The 3 x 3 carry-save (CS) array multiplier is illustrated in Figure 2. The array receives two 3-bit operands and produces a 6-bit product. The c and d inputs in the top row and the c inputs of the cells in the left most diagonal are connected to logical 0 during multiplication. However, it is assumed that during testing these inputs are available as primary inputs to the array.

The structure of the CS array multiplier is now formally defined. The n x n CS array multiplier consists of n^2 cells arranged as a shifted cascade of n rows of n cells each. There are n diagonals and 2n-1 columns. The rows are numbered 0, 1, . . . , n-1 from top to bottom, while the diagonals are numbered 0, 1, . . . , n-1 from right to left. A cell is denoted (i,j) if it is in the i^{th} row and j^{th} diagonal. The sum bit from cell (i,j) has positional weight of 2^{i+j}

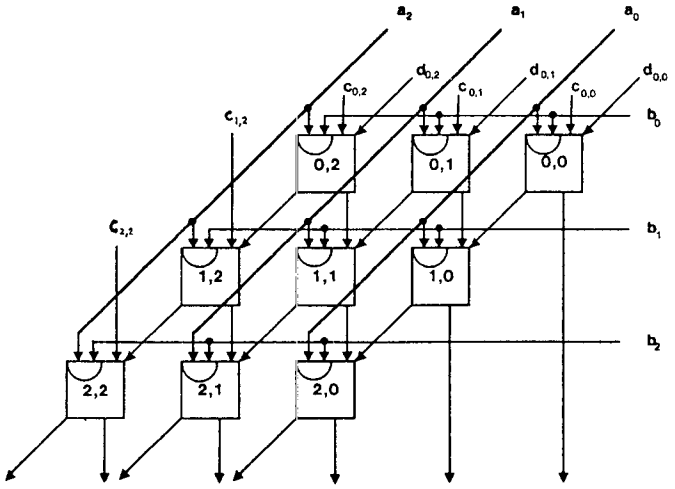

Fig. 2. The 3 x 3 carry-save array multiplier.

An array multiplier receives two n-bit operands, namely the multiplicand and the multiplier. The fanout of these input bits in the array results in the following constraints on the a and b inputs to all the cells.

$$a_{0,j} = a_{1,j} = \ldots = a_{n-1,j} \quad \text{for } j = 0, 1, \ldots, n-1$$
$$b_{i,0} = b_{i,1} = \ldots = b_{i,n-1} \quad \text{for } i = 0, 1, \ldots, n-1$$

Hence, the notations for the a and b cell inputs can be simplified by using a_j to denote all $a_{i,j}$ for i = 0, 1, . . . , n-1, and using b_i to denote all $b_{i,j}$ for j = 0, 1, . . . , n-1. The primary inputs to the CS array consist of the following five n-bit vectors.

$$\boldsymbol{a} = \langle a_{n-1}, a_{n-2}, \ldots, a_0\rangle$$
$$\boldsymbol{b} = \langle b_{n-1}, b_{n-2}, \ldots, b_0\rangle$$
$$\boldsymbol{c_0} = \langle c_{0,n-1}, c_{0,n-2}, \ldots, c_{0,0}\rangle$$
$$\boldsymbol{d_0} = \langle d_{0,n-1}, d_{0,n-2}, \ldots, d_{0,0}\rangle$$
$$\boldsymbol{c_{n-1}} = \langle c_{n-1,n-1}, c_{n-2,n-1}, \ldots, c_{0,n-1}\rangle$$

Consequently, each input pattern to the CS array can be denoted: $T^{CS} = [\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c_0}, \boldsymbol{d_0}, \boldsymbol{c_{n-1}}]$. The intercell connections in the array can be defined by equations of the form $p_{i,j} := q_{k,l}$ denoting the connection from the q output of cell (k,l) to the p input of cell (i,j). The intercell connections in the CS array multiplier can be divided into the following two groups:

Diagonal: $d_{i,j} := y_{i-1,j}$ for i = 1, 2, . . . , n-1 and j = 0, 1, . . . , n-1
Vertical: $c_{i,j} := x_{i-1,j+1}$ for i = 1, 2, . . . , n-1 and j = 0, 1, . . . , n-2

In order for an array multiplier to be C-testable, it must be testable and the number of test patterns required must be a constant. It can be shown that the carry-save array multiplier is

testable but the number of test patterns required is a function of the size of the array.

Assume that the cell input vector $\langle a,b,c,d\rangle = \langle 0001\rangle = v_1$ is to be applied to every cell in the array. Let v_1 be the input vector to cell (i,j), i.e. $\langle a,b,c,d\rangle_{i,j} = \langle 0001\rangle$. We now examine the implications on the input vectors of the other cells in the same j^{th} diagonal of the array. Since the a_j input is fanned out to all the cells in the j^{th} diagonal, the a input to every cell in the diagonal must be 0; see Figure 3. Since the carry output of cell (i,j) is $y_{i,j} = 0$ and $d_{i+1,j} := y_{i,j}$, the input vector to cell (i+1,j) must be $\langle a,b,c,d\rangle_{i+1,j} = \langle 0\text{-}0\rangle$, where - denotes either 0 or 1. With an input vector of the form $\langle 0\text{-}0\rangle$, the carry output of cell (i+1,j) must be 0. Hence the input vector to cell (i+2,j) must also be $\langle 0\text{-}0\rangle$. Continuing the same argument, it can be concluded that all cells below the cell (i,j) in the j^{th} diagonal must have input vectors of the form $\langle 0\text{-}0\rangle$.

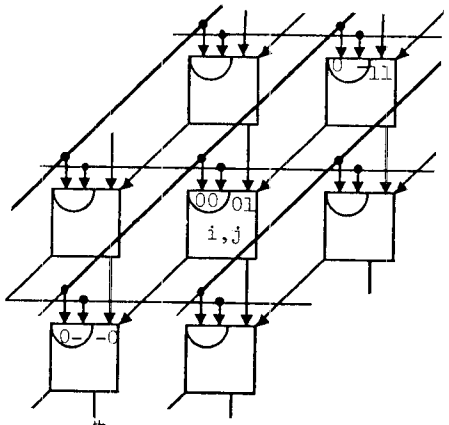

Fig. 3. The j^{th} diagonal with $\langle a,b,c,d\rangle_{i,j} = \langle 0001\rangle$.

The cell input vector $\langle a,b,c,d\rangle_{i,j} = \langle 0001\rangle$ implies that the carry input to cell (i,j) and the carry output from cell (i-1,j) must be equal to 1. Since $a_{i-1,j} = 0$, in order to produce a carry output of 1 the input vector to cell (i-1,j) must be $\langle a,b,c,d\rangle_{i-1,j} = \langle 0\text{-}11\rangle$. Following the same line of reasoning, the input vector to cell (i-2,j) must also be a $\langle a,b,c,d\rangle_{i-2,j} = \langle 0\text{-}11\rangle$. In fact, all the cells above the cell (i,j) in the j^{th} diagonal must have input vectors of the form $\langle 0\text{-}11\rangle$.

It can be concluded from the above arguments that no two cells in the same diagonal can simultaneously receive the input vector $\langle 0001\rangle$. Hence, n test patterns are needed in order to apply the $\langle 0001\rangle$ input vector to each of the n cells in the same diagonal. Clearly, the number of test patterns required to fully test the array will depend on the size of the array. The foregoing arguments lead to the following theorem.

Theorem 1: The carry-save array multiplier is not C-testable.

A modified design of the CS array multiplier is now presented. The modified carry-save (MCS) array multiplier has the exact same structure as the unmodified CS array multiplier, only the function of the basic cell is changed. The basic cell is modified so that for input vectors $\langle 0001\rangle$ and $\langle 0101\rangle$ the y output has the value 1. There is no other modification to the basic cell. The Karnaugh maps for the original and the modified y output functions are shown in Figure 4.

The modification to the basic cell does not affect the multiplication function of the array. When the array is performing a multiplication, the primary inputs $c_{0,j}$ and $d_{0,j}$, for $j = 0, 1, \ldots, n\text{-}1$, are set to logical 0. Consequently, the input vectors $\langle 0001\rangle$ and $\langle 0101\rangle$ can never appear at the inputs to any cell in the array. Thus by changing the y output values for the input conditions $\langle 0001\rangle$ and $\langle 0101\rangle$ the multiplication function of the array is not affected. We now show that the MCS array multiplier is C-testable.

* modified minterms

ab \ cd	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	0

standard carry-out

ab \ cd	00	01	11	10
00	0	1*	1	0
01	0	1*	1	0
11	0	1	1	1
10	0	0	1	0

modified carry-out

Fig. 4. The truth table of the modified basic cell of the MCS array.

In order to show that the MCS array multiplier is C-testable, we must show that: (1) using a constant number of test patterns, all 16 input vectors $v_0, v_1, \ldots, v_{15}$ can be applied to every cell; and (2) if a faulty cell exists the faulty output signal from this cell can be detected at the primary outputs.

Part (1) can be shown by a constructive procedure. As stated earlier, a test pattern for the CS, hence the MCS, array multiplier is denoted: $T = [a, b, c_0, d_0, c_{n-1}]$. Let T_i denote the test pattern which simultaneously applies the v_i input vector to every cell in the array. It can be shown that the following group of six test patterns applies the six input vectors, v_0, v_2, v_4, v_6, v_{10}, and v_{15}, to every cell in the array.

$T_0 = [\langle 00\ldots00\rangle \langle 00\ldots00\rangle \langle 00\ldots00\rangle \langle 00\ldots00\rangle \langle 00\ldots00\rangle]$
$T_2 = [\langle 00\ldots00\rangle \langle 00\ldots00\rangle \langle 11\ldots11\rangle \langle 00\ldots00\rangle \langle 11\ldots11\rangle]$
$T_4 = [\langle 00\ldots00\rangle \langle 11\ldots11\rangle \langle 00\ldots00\rangle \langle 00\ldots00\rangle \langle 00\ldots00\rangle]$
$T_6 = [\langle 00\ldots00\rangle \langle 11\ldots11\rangle \langle 11\ldots11\rangle \langle 00\ldots00\rangle \langle 11\ldots11\rangle]$
$T_{10} = [\langle 11\ldots11\rangle \langle 00\ldots00\rangle \langle 11\ldots11\rangle \langle 00\ldots00\rangle \langle 11\ldots11\rangle]$
$T_{15} = [\langle 11\ldots11\rangle \langle 11\ldots11\rangle \langle 11\ldots11\rangle \langle 11\ldots11\rangle \langle 11\ldots11\rangle]$

Figure 5 illustrates how the test pattern T_{10} produces the input vector $v_{10} = \langle 1010\rangle$ at every cell.

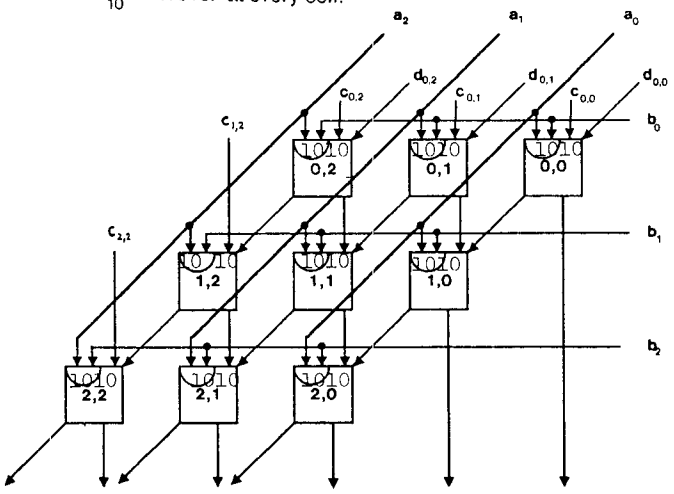

Fig. 5. Test pattern T_{10} for the MCS array.

A second group of test patterns can be constructed. Each test pattern in this group, denoted $T_{i,j}$, applies one input vector, namely v_i, to one half of the cells in the array and applies another input vector, namely v_j, to the other half of the cells. Each test pattern $T_{i,j}$ has a companion test pattern, denoted $T_{j,i}$, which applies v_j (v_i) to those cells having the input vector v_i (v_j) under $T_{i,j}$. Hence, by using the pair of test patterns $T_{i,j}$ and $T_{j,i}$ the input vectors v_i and v_j can be applied to every cell in the array. The test patterns in this group are listed below.

$T_{1,3}$ = [<00...00> <00...00> <11...11> <11...11> <01...01>]
$T_{3,1}$ = [<00...00> <00...00> <00...00> <11...11> <10...10>]
$T_{5,7}$ = [<00...00> <11...11> <00...00> <11...11> <10...10>]
$T_{7,5}$ = [<00...00> <11...11> <11...11> <11...11> <01...01>]
$T_{9,14}$ = [<11...11> <10...10> <00...00> <00...00> <10...10>]
$T_{14,9}$ = [<11...11> <01...01> <11...11> <00...00> <01...01>]

Similar to $T_{i,j}$, $T_{i,j,k,l}$ denotes a test pattern which applies each of the four input vectors, v_i,v_j, v_k, and v_l, to exactly 1/4 of the cells in the array. $T_{j,k,l,i}$, $T_{k,l,i,j}$, and $T_{l,i,j,k}$ can be similarly defined, and together with $T_{i,j,k,l}$ can be used to apply the four input vectors v_i, v_j, v_k, and v_l to all the cells in the MCS array. The third and last group of test patterns for the MCS array is listed below with $T_{8,11,12,13}$ illustrated in Figure 6.

$T_{8,11,12,13}$ = [<11...11> <01...01> <00...00> <01...01> <00...00>]
$T_{11,12,13,8}$ = [<11...11> <10...10> <01...01> <01...01> <00...00>]
$T_{12,13,8,11}$ = [<11...11> <01...01> <00...00> <10...10> <10...10>]
$T_{13,8,11,12}$ = [<11...11> <10...10> <10...10> <10...10> <01...01>]

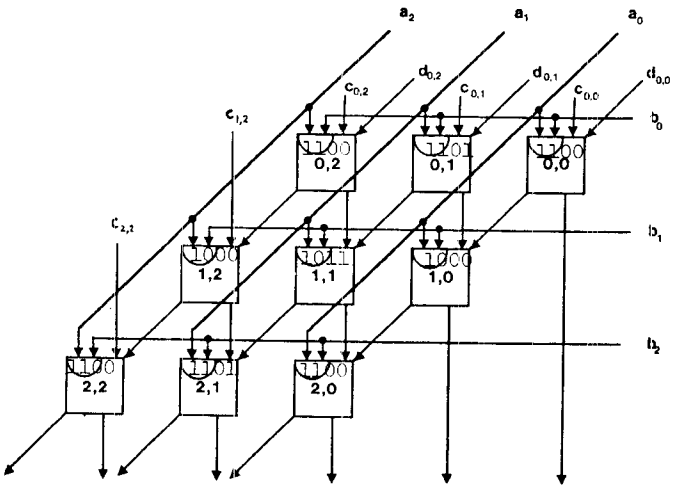

Fig. 6. Test Pattern $T_{8,11,12,13}$ for the MCS array.

It can be observed that all 16 input vectors can be produced at the inputs of every cell in the MCS array by applying the 16 test patterns from the above three groups. Furthermore, the number of test patterns is clearly independent of the array size. The above construction procedure leads to the following lemma.

Lemma 2: Given a n x n modified carry-save array multiplier, only 16 test patterns are needed to apply all 16 cell input vectors, v_0, v_1, . . . , v_{15}, to each of the n^2 cells in the array.

It remains to be shown that for each of the 16 test patterns the faulty signal produced by a faulty cell can always be propagated to the primary outputs. We use Δ to denote the value of a signal line which has been inverted due to a faulty cell. Hence, if cell (i,j) is faulty, then the three possible faulty output conditions are: $\langle\Delta,y\rangle_{i,j}$, $\langle x,\Delta\rangle_{i,j}$, or $\langle\Delta,\Delta\rangle_{i,j}$. Consequently, Lemma 1 can be restated as: given a fault-free cell (k,l), if only the c or the d input has the value Δ then the x output will have the value Δ. Clearly, if a Δ is propagated to a primary output then the fault is detectable.

Lemma 3: Any faulty signal produced by a faulty cell (i,j) in a MCS array multiplier can be propagated to a primary output.
Proof: Since all signals flow from top to bottom a signal originating from cell (i,j) cannot propagate to cell (k,l) for any k and l such that $k+l<i+j$ or $k \leq i$ or $l > j$.

Case 1: If the fault in cell (i,j) produces a faulty signal Δ on the sum output $x_{i,j}$ then a Δ will be propagated to the c input of cell (i+1,j-1) which is directly below cell (i,j). Since any faulty signal must originate from cell (i,j) the d input of cell (i+1,j-1) cannot have the value Δ because $d_{i+1,j-1} := y_{i,j-1}$ and $i+j-1 < i+j$. Hence, based on Lemma 1, the x output of cell (i+1,j-1) must also be Δ. Continuing this line of argument it can be shown that all x outputs in the same column as $x_{i,j}$ must have the value Δ. In effect, the faulty signal Δ is propagated down the column until it reaches a primary output $x_{n-1,k}$ where $k = (i+j)-(n-1)$ and $i+j > n-1$, or $x_{k,0}$ where $k = i+j$ and $i+j \leq n-1$.

Case 2: If the fault in cell (i,j) produces a faulty signal Δ only on the carry output $y_{i,j}$ then the d input to cell (i+1,j) will have the value Δ. The c input of cell (i+1,j) cannot have the value Δ because it originates from cell (i,j+1) and $j+1 > j$. Hence, using Lemma 1, the sum output of cell (i+1,j) must have the value Δ. The same argument used in case 1 can now be applied. Hence the faulty signal $x_{i+1,j} = \Delta$ will be propagated down the column to a primary output $x_{n-1,k}$ where $k = (i+j+1)-(n-1)$ and $i+j+1 > n-1$, or $x_{k,0}$ where $k = i+j+1$ and $i+j+1 \leq n-1$. Δ

Combining Lemmas 2 and 3, the following Theorem is obtained.

Theorem 2: The n x n modified carry-save array multiplier is C-testable. The number of test patterns required is 16.

We have modified an array multiplier which is not C-testable to one which can be fully tested using a constant number of test patterns. During normal operation, the n c inputs, $\{c_{0,0}, \ldots, c_{0,n-1}\}$, and the n d inputs, $\{d_{0,0}, \ldots, d_{0,n-1}\}$, are connected to logical 0. During test mode, control of these signal lines are needed. However, all the c (d) inputs with odd subscripts j always have the same value and all the even subscripted inputs always have the same value. Hence, only two additional primary inputs are needed to control the c inputs, and two more for the d inputs. Similarly, two additional primary inputs are needed to control the n c inputs, $\{c_{0,n-1}, \ldots, c_{n-1,n-1}\}$. A total of six additional primary inputs are needed in the MCS array multiplier for control during testing. This number is independent of the array size. It is worth noting that the 16 test patterns required is optimal in the sense that each cell has four inputs and to exhaustively test each cell at least 16 test patterns to the array are required.

In designing a practical carry-save array multiplier a final row of carry-propagate adders are needed to produce the higher order product bits. We denote the MCS array multiplier with a final row of carry-propagate adders as the MCS/CP array multiplier. The 3 x 3 MCS/CP array multiplier is depicted in Figure 7.

The testing of the MCS/CP array multiplier must include the testing of the last row of full adders. Eight input vectors must be applied to each of the full adders in the last row. Fortunately, it can be shown that in the process of applying the 16 test patterns to the MCS portion of the array, all 8 input vectors are applied to every full adder in the last row. The only requirement is that the carry-in input to the last row must be controlled. It is also easy to show that the last row of full adders does not hinder faulty signal propagation to primary outputs. The least significant erroneous sum bit from the MCS subarray will produce an erroneous bit at the sum output of the corresponding carry-propagate full adder. Hence, the n x n MCS/CP array multiplier is also C-testable and requires one additional primary input than the MCS array multiplier.

Theorem 3: The n x n modified carry-save array multiplier with a final row of carry-propagate adders is C-testable and requires 16 test patterns.

4. THE BAUGH-WOOLEY ARRAY MULTIPLIER

In this section, the same testing methodology is applied to the Baugh-Wooley (BW) two's complement array multiplier [1]. Unlike the CS array multiplier the BW array multiplier is not an iterative array because it contains seven, slightly different, types of cells. All seven cell types are based on the full adder. Figure 8 illustrates each of the seven cell types and defines their logical functions and input and output notations. As Figure 9 shows, the BW array multiplier consists of a two dimensional subarray of type 3 cells and four one-dimensional subarrays (one subarray of type 1 cells, one of type 2 cells, one of type 4 cells and one of type 7 cells). There are three sets of inputs to the BW array multiplier $a = \langle a_{n-1}, a_{n-2}, \ldots a_0 \rangle$, $b = \langle b_{n-1}, b_{n-2}, \ldots, b_0 \rangle$, and $d = \langle d_{n-2}, d_{n-3}, \ldots d_0 \rangle$. The input vector d is connected to logical 0 during multiplication but is available as primary inputs during testing.

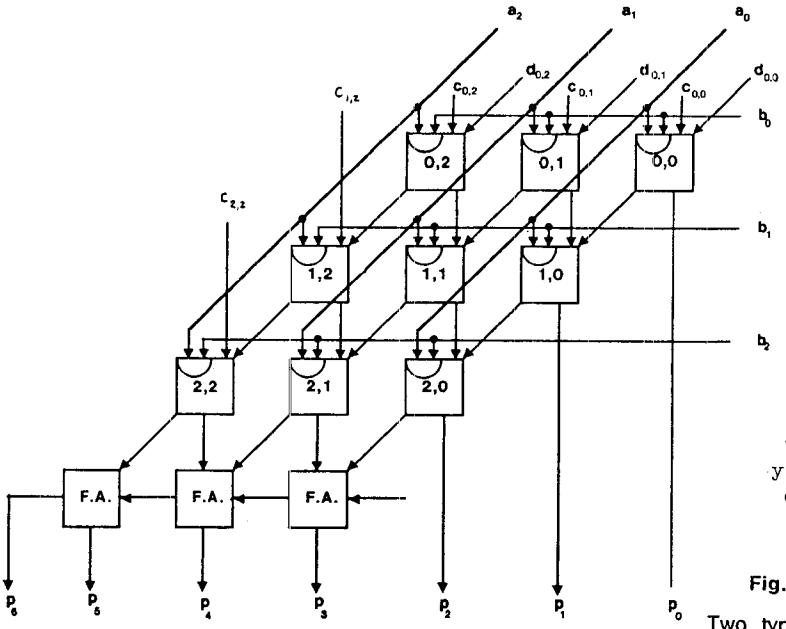

Fig. 7. The 3 x 3 MCS/CP array multiplier.

The type 3 cell is the unmodified basic cell used in the carry-save array multiplier. These cells are connected in the same way as in the carry-save array multiplier. Hence, like the unmodified carry-save array multiplier, the two-dimensional subarray of type 3 cells is not C-testable. This leads to the following theorem.

Theorem 4: The Baugh-Wooley array multiplier is not C-testable.

Theorem 4 can be proved by showing that no more than one cell in any diagonal in the subarray of type 3 cells may have the input vector $\langle abcd \rangle = \langle 0100 \rangle$. To make the BW array multiplier C-testable it is sufficent to make each subarray in it C-testable while meeting the constraints imposed by the signal flow between adjacent subarrays. For instance, the a and b inputs to subarray of type 3 cells are the outputs of the type 1 cells and are not directly accessable to the tester. Hereafter a subarray of type N cells is called a type N subarray for $N \in \{1,2,3,4,7\}$.

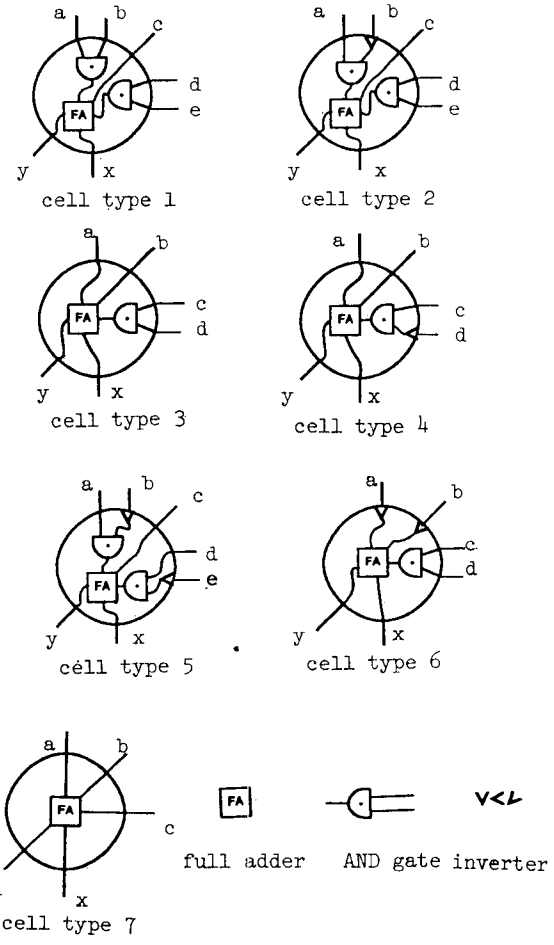

Fig. 8. The seven cell types of the BW array multiplier.

Two types of modifications to the BW array multiplier are necessary. The first modification is to add an input to the array multiplier, called e, and n-2 exclusive-OR (XOR) gates. The input e is connected to one of the inputs of each XOR gate. The other input to the XOR gate is b_i, where i = 0, 1, . . . , n-1, and the outputs of the XOR gates are connected to the b inputs of type 2 and type 5 cells. Notice that the b input of a type 2 cell is the same as the d input of the type 2 cell above it. The XOR gates allow the tester to invert the signal values to these inputs. The second type of modification is to modify the truth tables of cell types 1, 2, 3 and 5.

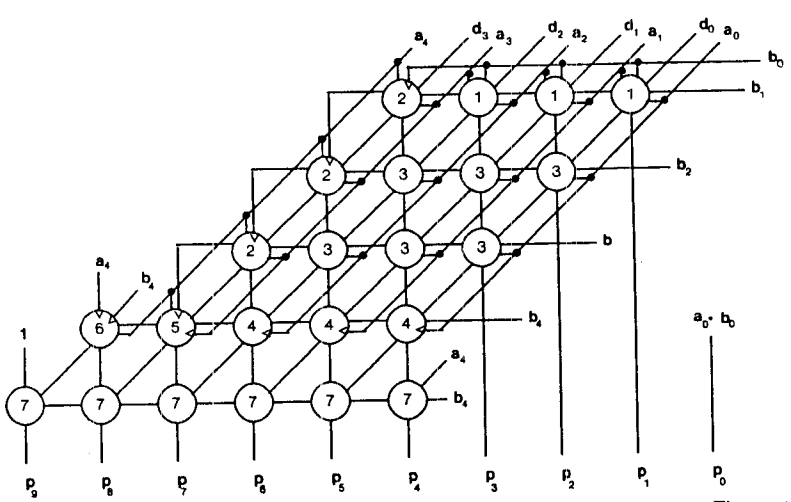

Fig. 9. The 5 x 5 Baugh-Wooley array multiplier.

The type 1 cells are modified to increase the controllability of the inputs to the type 3 subarray so as to make it C-testable. The truth table entries for <abcde> = <--1--> may be changed because when the array is multiplying these inputs to type 1 cells can never occur. The type 2 cells are modified for two reasons. The first is to increase the controllability of the inputs in the type 3 subarray. Two modified designs, type 2A and type 2B, of type 2 cells are needed. The type 2A and type 2B cells are arranged in the type 2 subarray such that no two cells of type 2A are adjacent and no two cells of type 2B are adjacent and the upper leftmost cell is type 2A. The second reason to modify type 2 cells is because the type 2 subarray is not C-testable without modification. An input vector to a type 2 cell which causes the subarray to be not C-testable is <abcde> = <00100>. Only the truth table entries for <abcde> = <0-1--> and <--1-0> may be changed because when the array is multiplying these cell input vectors do not occur. Since the type 3 subarray is a carry-save multiplier design it is modified as in the modified carry-save array. Only truth table entries <abcd> = <0-01> may be changed since all other input vectors may occur during multiplication. The type 5 cell is modified to provide controllability to the second leftmost type 7 cell. Below is a list of the modifications to the cells of the BW array multiplier in order to make it C-testable. The outputs of the input vectors that are not shown are not changed.

	<abcde> or<abcd>	unmodified <xy>	modified <xy>
Type 1:	<10101>	<10>	<01>
	<01111>	<01>	<11>
	<00100>	<10>	<11>
	<01100>	<10>	<01>
	<01110>	<10>	<11>
	<00110>	<10>	<01>
Type 2A:	<10100>	<01>	<11>
	<00100>	<10>	<01>
	<01110>	<10>	<01>
	<11110>	<10>	<11>
Type 2B:	<00100>	<10>	<11>
	<01110>	<10>	<11>
	<11110>	<10>	<01>
Type 2A & 2B:	<00110>	<10>	<01>
	<01100>	<10>	<01>
	<00101>	<10>	<01>
	<01101>	<10>	<01>
	<11100>	<10>	<11>
Type 5:	<00100>	<10>	<11>
	<01110>	<10>	<11>
Type 3:	<0100>	<10>	<11>
	<0110>	<10>	<11>

The set of test patterns for the BW array multiplier is now presented. The table below is organized into four columns. The first column identifies the test patterns to the BW array multiplier. There are 55 tests, t_0 to t_{54}. The second, third and fourth columns show the input vectors ***a***,***b*** and ***d***. In these last three columns, all individual bits not separated by a comma are repeated according to the patterns indicated. For example ***a*** = <1,1,01...01> means that a_{n-1} and a_{n-2} have the value 1 while the remaining bits alternate their values so that all bits with even subscripts have the value 1 and all bits with odd subscripts have the value 0.

test	$\langle a_{n-1},\ldots,a_0\rangle$	$\langle b_{n-1},\ldots,b_0\rangle$	$\langle d_{n-2},\ldots,d_0\rangle$	<e>
t_0	<0,0,00...00>	<00...00,0,0>	<0,00...00>	<0>
t_1	<0,0,00...00>	<11...11,1,1>	<0,00...00>	<0>
t_2	<0,1,11...11>	<10...10,1,0>	<0,10...10>	<0>
t_3	<0,0,11...11>	<10...10,1,0>	<0,01...01>	<0>
t_4	<0,0,11...11>	<01...01,0,0>	<0,01...01>	<0>
t_5	<0,1,11...11>	<01...01,0,0>	<0,10...10>	<0>
t_6	<1,0,00...00>	<00...00,0,0>	<1,11...11>	<0>
t_7	<0,0,00...00>	<00...00,0,1>	<1,11...11>	<0>
t_8	<1,1,11...11>	<10...10,1,0>	<0,11...11>	<0>
t_9	<1,1,11...11>	<01...01,0,1>	<1,00...00>	<0>
t_{10}	<1,0,00...00>	<11...11,1,1>	<1,11...11>	<0>
t_{11}	<0,0,00...00>	<11...11,1,0>	<1,11...11>	<0>
t_{12}	<1,0,10...10>	<00...00,0,0>	<0,11...11>	<0>
t_{13}	<1,1,01...01>	<00...00,0,0>	<0,11...11>	<0>
t_{14}	<1,1,01...01>	<11...11,1,1>	<0,01...01>	<0>
t_{15}	<1,1,10...10>	<11...11,1,1>	<0,10...10>	<0>
t_{16}	<1,0,10...10>	<00...00,0,0>	<0,00...00>	<0>
t_{17}	<0,1,01...01>	<00...00,0,0>	<0,00...00>	<0>
t_{18}	<0,0,10...10>	<11...11,1,0>	<0,00...00>	<0>
t_{19}	<1,0,01...01>	<11...11,1,0>	<0,00...00>	<0>
t_{20}	<1,0,10...10>	<00...00,1,0>	<1,11...11>	<0>
t_{21}	<0,1,01...01>	<00...00,1,0>	<1,11...11>	<0>

t_{22}	<0,0,00...00>	<00...00,0,1>	<0,00...00>	<0>
t_{23}	<0,1,01...01>	<00...00,0,1>	<0,00...00>	<0>
t_{24}	<1,0,10...10>	<00...00,0,1>	<0,00...00>	<0>
t_{25}	<0,1,01...01>	<11...11,1,1>	<0,00...00>	<0>
t_{26}	<1,0,10...10>	<11...11,1,1>	<0,00...00>	<0>
t_{27}	<0,1,01...01>	<00...00,0,1>	<1,11...11>	<0>
t_{28}	<1,0,10...10>	<00...00,0,1>	<0,11...11>	<0>
t_{29}	<1,0,10...10>	<11...11,1,1>	<1,11...11>	<0>
t_{30}	<0,1,01...01>	<11...11,1,1>	<1,11...11>	<0>
t_{31}	<1,1,11...11>	<00...00,0,0>	<0,00...00>	<0>
t_{32}	<1,1,11...11>	<11...11,1,0>	<0,00...00>	<0>
t_{33}	<1,1,11...11>	<11...11,1,1>	<0,00...00>	<0>
t_{34}	<1,1,11...11>	<11...11,0,1>	<1,11...11>	<0>
t_{35}	<1,1,11...11>	<11...11,1,1>	<1,11...11>	<0>
t_{36}	<0,1,11...11>	<01...01,0,1>	<0,00...00>	<0>
t_{37}	<0,1,11...11>	<10...10,1,0>	<0,00...00>	<0>
t_{38}	<0,0,11...11>	<00...00,0,0>	<1,00...00>	<0>
t_{39}	<0,1,11...11>	<00...00,0,0>	<1,11...11>	<0>
t_{40}	<0,0,00...00>	<10...10,1,0>	<1,11...11>	<0>
t_{41}	<0,0,00...00>	<01...01,0,1>	<1,11...11>	<0>
t_{42}	<0,1,11...11>	<01...01,0,1>	<1,11...11>	<0>
t_{43}	<0,1,11...11>	<10...10,1,0>	<1,11...11>	<0>
t_{44}	<0,0,00...00>	<11...11,1,1>	<1,11...11>	<0>
t_{45}	<1,0,00...00>	<10...10,1,0>	<0,00...00>	<0>
t_{46}	<1,0,00...00>	<01...01,0,1>	<0,00...00>	<0>
t_{47}	<1,1,11...11>	<00...00,0,0>	<1,11...11>	<0>
t_{48}	<1,0,00...00>	<01...01,0,1>	<1,11...11>	<0>
t_{49}	<1,0,00...00>	<10...10,1,0>	<1,11...11>	<0>
t_{50}	<1,1,11...11>	<11...11,1,0>	<1,11...11>	<1>
t_{51}	<1,1,11...11>	<00...00,0,1>	<0,00...00>	<1>
t_{52}	<1,0,00...00>	<1,0,00...00>	<1,00...00>	<0>
t_{53}	<1,0,00...00>	<1,0,00...00>	<0,00...00>	<0>
t_{54}	<0,0,11...11>	<0,1,11...11>	<1,11...11>	<0>

It can be verified that the application of the above set of 55 test patterns will exhaustively exercise every cell in the modified BW array multiplier. The details of this verification is rather lengthy and is documented in [9]. The set of 55 test patterns is shown to be sufficient but not necessarily minimal. It is conjectured that a minimal test set consisting of less than 55 test patterns exists.

To show C-testability it is sufficient to show that all cells are exhaustively exercised and that any error is propagated to an output of the array. The first condition is met by the application of the test set consisting of t_0 to t_{54}. The second condition is proven by showing that the faulty-signal propagation property described earlier for CS array multipliers is also true for all cells in the modified BW array multiplier. That is, the x output will change in response to a change in one of its inputs that are x or y outputs of other cells. It is easy to verify that all cells in the unmodified BW array multiplier have this property. The interconnections between cells in the BW array multipier are similar to those in the CS array multiplier which allow faulty signal to propagate down a column, via the x outputs, in the array. While making the modifications to the original BW array multiplier design, this property is preserved with the exception of the type 1 and type 2 cells. Since type 1 cells do not have inputs generated by other cells, its ability to propagate faulty signals is unimportant. It is assumed that inputs to the array are error free. The only faulty signals that a type 2 cell is required to propagate are errors on its c input. From the description of the cell it is obvious that an error on the c input will cause a change in the x or y output. If the x output is faulty then the error will propagate to an output of the array as in the CS array. If the error manifests itself on the y output it will appear as an error on the c input of the type 2 cell below it. Since there can be no other faulty signals on the inputs of that type 2 cell, any faulty signal can be propagated to an array output. The foregoing arguments lead to the following theorem.

Theorem 7: The modified Baugh-Wooley array multiplier is C-testable and can be fully tested using only 55 test patterns.

5. IMPLEMENTATION AND EXTENSIONS

A C-testable 16 x 16 array multiplier, implementing the MCS/CP structure, is designed using the Mead and Conway [8] design approach. It is assumed that a 40-pin DIP package will be used to house the array multiplier chip. Hence, 32 I/O pins are multiplexed to be able to receive the two 16-bit operands and to output the 32-bit product. To accomplish this multiplexing, memory elements and associated control circuitry are added at the periphery of the array. The array portion consists of 256 modified basic cells and occupies an area of 99 x 55 λ^2. λ is the elementary length unit and is equal to three microns in this design. Hence, the total area of the multiplier array is 4.752 mm x 2.64 mm. The actual size of the chip, containing over 10,000 devices, is some what larger because of the additional control circuitry and bonding pads. A layout has been generated and is ready to be implemented on an NMOS chip.

It can be shown that an unmodified basic full adder cell requires approximately 86 λ^2 units of area. The total area required by the modified basic cell is 87 λ^2 units. Hence, it can be concluded that no significant increase of chip area is needed for the modified carry-save design. Furthermore the modification to the basic cell for testability does not incur any additional delay. Figure 10 illustrates the layout of the modified basic cell. We have designed a 16 x 16 array multiplier which is C-testable. Only 16 test patterns need be applied to the array to fully test the array multiplier. The same number of test patterns is required regardless of the size of the array multiplier. Furthermore, there is no increase of delay nor chip area.

The research presented in this paper can be extended in several directions. In this work, it has been assumed that only one basic cell can fail at a time. A less restrictive fault model can be developed and the C-testability can be generalized. The application of the above approach to other array multipliers can be studied. The development of algorithmic modification procedures for any given multiplier structure to generate C-testable designs seems feasible. Some of these efforts are underway at CMU. It is anticipated that the above-mentioned research will result in the development of general procedures for the design of very easily-testable array-structured VLSI circuits and systems.

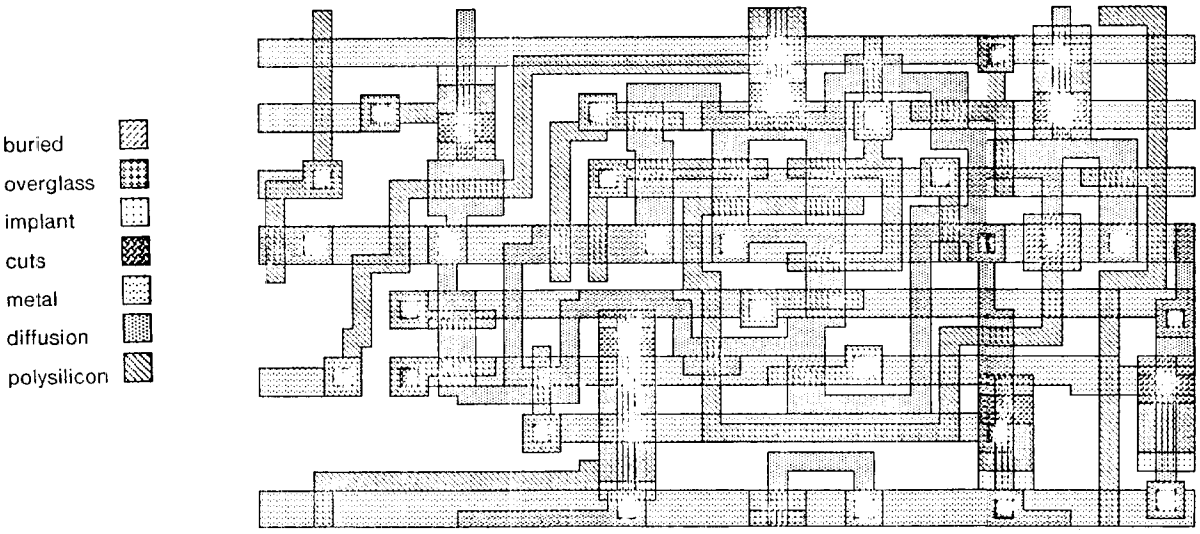

Fig. 10. The layout of the MCS array basic cell.

6. REFERENCES

[1] Hwang, K., *Computer Arithmetic, Principles, Architecture, and Design*, John Wiley & Sons, 1979.

[2] Waser, S., "High speed monolithic multipliers for real-time digital signal processing," *Computer*, pp. 19--28, October 1978.

[3] Williams, T. and K. Parker, "Design for testability -- a survey," *IEEE Trans. on Computers*, vol. C-31, January 1982.

[4] Kautz, W.H., "Testing for faults in cellular logic arrays " *Proc. of 8th Symp. on Switching Automata Theory*, pp. 161--174, 1967.

[5] Friedman, A.D., "Easily testable iterative systems," *IEEE Trans. on Computers*, vol C-22, pp. 1061--1064, December 1973.

[6] Parthasarathy, R. and S.M. Reddy, "A testable design of iterative logic arrays," *IEEE Trans. on Circuits and Systems*, vol. CAS-28, November 1981.

[7] Sridhar, T. and J.P. Hayes, "A functional approach to testing bit-sliced microprocessors," *IEEE Trans. on Computers*, vol. C-30, August 1981.

[8] Mead, C. and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley Publishing Company, Inc., 1980.

[9] Ferguson, J. and J.P. Shen, "Design of Easily-Testable VLSI Array Multipliers," Dept. of E.E., CMU, Res. Rep. No. CMUCAD-83-8, February 1983.

[10] Shen, J.P. and J. Ferguson, "Easily-Testable Array Multipliers," *Proc. of 13th Int. Symp. on Fault-Tolerant Computing*, June 1983.

Acknowledgement: The authors would like to thank Ms. Lori Rosen for her help in the preparation of this paper.