Detection of multiple faults in two-dimensional ILAs

Martine Schlag and F. Joel Ferguson

UCSC-CRL-95-30
June 13, 1995

Associate Professors of Computer Engineering
Baskin Center for
Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

Two-dimensional arrays are suitable for VLSI implementation because of their regular structure and relative ease of test. We provide test sets proportional to the sum of the two dimensions of the array for a large class of cells, which allow us to test rows (or columns) of cells of the array independently. Previous research has found constant length test sets for array multipliers under the single faulty cell model if the array is modified and otherwise test sets are proportional to the number of cells. We can verify the full adder array of a combinational $n \times m$ multiplier in $O(n + m)$ tests under the Multiple Faulty Cell (MFC) model. We show that no constant length test set exists for this array under the MFC model. The entire multiplier, including the AND gates which generate the summands, can be verified after applying the same modifications which make the multiplier C-testable under the single faulty cell model. Finally we show an error in the proof of a commonly accepted theorem involving testing two-dimensional arrays for multiple faults.

Contents

1. Introduction

An iterative logic array (ILA) is a circuit consisting of identical cells of combinational logic arrayed with a regular interconnection pattern. An n-bit ripple-carry adder consisting of n full adders is an example of a unidirectional one-dimensional ILA. The arrays we shall consider are *unidirectional* in that signals only flow in one direction between adjacent cells.

The problem of testing one-dimensional and two-dimensional arrays for single faults has been studied extensively [1, 2, 3, 4, 5, 6, 7, 8] with considerable attention on arithmetic circuits. Researchers have modified the array multiplier to make it *C-testable* [2], that is, testable with a constant number of tests independent of the size of the array multiplier [9, 7, 10]. Their test sets exploit the regularity in the array's iterative structure under the *single faulty cell* model in which,

1. at most one cell is faulty,
2. and the fault may alter the cell's output function in any arbitrary way, as long as the cell remains combinational and the fault is permanent.

Detecting these faults requires the exhaustive testing of every cell with tests which guarantee that any error appearing at the output of a cell will appear as an error at the array's primary outputs. This fault model subsumes the traditional single line stuck-at fault model and covers all multiple line stuck-at faults restricted to a single cell.

Multiple defects which often occur on an IC [11], may affect multiple cells in the array. Multiple faulty cells may not be detected by tests based on a single faulty cell fault model. In this paper, we consider the more general *multiple faulty cell* (MFC) model[12, 13] in which,

1. any number of cells in the array can be faulty,
2. and each faulty cell may have its output function altered in any arbitrary way, as long as each cell remains combinational and the fault is permanent.

Dias developed methods for one-dimensional arrays under this model. He called this *Truth Table Verification* because some multiple faults do not change the function of the truth table of the array. Since these are undetectable the truth table of the array is verified for all multiple faults. For a class of one-dimensional ILAs, he developed a procedure which constructs a constant number of tests independent of the number of cells in the array. Prasad and Gray provided test sets of length proportional to the number of cells ($O(nm)$ tests for an $n \times m$ array) for a class of two-dimensional arrays under the MFC model[14]. We provide test sets of length proportional to the sum of the dimensions of the array ($m+n$ for an $n \times m$ array) which apply to cells meeting Dias' requirements and a weaker version of Prasad and Gray's requirements. In his thesis, Cheng also provided sufficient conditions for a test set to verify two-dimensional ILAs under the MFC model. He applied his result to this same class to derive test sets of length proportional to the number of rows, independent of the number of columns. Unfortunately, there is a fallacy in the proof. We explain the error in the appendix. We also provide lower bounds for two of the arrays to which we apply our results to show that they are not *C-testable*. Any cell can be modified to meet our requirements, by adding at most one horizontal and one vertical connection.

As in the case of Dias, Prasad and Gray as well as Cheng, our tests do not verify that each cell is correct under the MFC model. This is in contrast to the tests for arrays under the single faulty cell model. If multiple faulty cells are allowed, it is possible for the array to have a correct truth table, even though its cells do not. Adjacent cells may select any encoding for their shared wires as long as they agree. Of course, the cells whose inputs and outputs are external to the array must follow the encoding expected of a correct cell on their external wires. Prasad and Gray handle this problem by showing that subsections of the array are correct for some encoding of their outputs. Cheng's approach is to verify that the outputs of each cell has a one-to-one correspondence with the correct cell's. Our approach is to show that each cell is correct with respect to a fixed encoding convention defined by a set of preliminary tests. We find that this approach simplifies and clarifies the arguments of correctness.

In Chapter 1.1 we show that for a certain class of cells (*column-separable cells*), we can apply tests to the array which allow us to test each row independently. These tests exist for a restricted class of cells whose functions allow values to propagate down the columns independently. The $n + m + 1$ tests allow us to reduce the problem of testing an $n \times m$ two-dimensional array to the problem of testing n one-dimensional arrays of m cells. In Chapter 1.1.2, we review Dias' work on testing one-dimensional arrays with multiple faulty cells, which provides constant length test sets for a large class of arrays. Chapter 2 builds on the results in Chapter 1.1 to develop a test set for a carry-save array multiplier linear in the size of its operands. The test set derived from Chapter 1.1 is modified and augmented in Chapter 2.2 to deal with the hybrid structure of the array, and then with the incorporation of the AND gates from the summand generator into the full adder cells. An $\Omega(n/\log n)$ lower bound on the FA array of a multiplier shows that this test set is within a $\log n$ factor of optimal.

1.1 Testing 2-D ILAs with tests sets proportional to the sum of the dimensions

In this section, a test set for a class of two-dimensional ILAs is constructed by reducing the problem of testing a two-dimensional array to that of testing one-dimensional ILAs. The class of two-dimensional arrays to which this method applies consists of the ILAs composed of a cell whose function allows the vertical propagation of values while fixing a specific horizontal input value. The exact definition is given in Section 1.1.1.

1.1.1 Reducing testing of 2-D ILAs to testing of 1-D ILAs

The cells are indexed by row i and column j in the direction of data flow. Each cell $C_{i,j}$ has a row input (x) and row output ($h(x, y)$), a column input (y) and a column output ($v(x, y)$). Each connection between adjacent vertical or horizontal cell will be referred to as a 'wire' although it may carry more than two values and be implemented with more than one physical wire. The row inputs to the array are $H_1, \ldots, H_n$ and the column inputs are $V_m, \ldots, V_1$, where m is the number of columns and n is the number of rows. The column outputs of the array are $V'_m, \ldots, V'_1$, and the row outputs are $H'_1, \ldots, H'_n$. Cell $C_{i,j}$ is on the k^{th} diagonal if $i + j - 1 = k$. Figure 1.1 illustrates a 5 by 4 ILA. An input to the array is denoted by $(\vec{V}, \vec{H})$ where $\vec{V}$ is the vector of left to right columns inputs and $\vec{H}$ is the vector of top to bottom row inputs. The behavior of a cell is described by a pair of functions,

$$h(x, y) : X \times Y \to X \text{ and } v(x, y) : X \times Y \to Y,$$

where X and Y are the sets of all possible correct or incorrect horizontal and vertical signals, respectively. If vertical wires are implemented by k physical wires and horizontal wires are

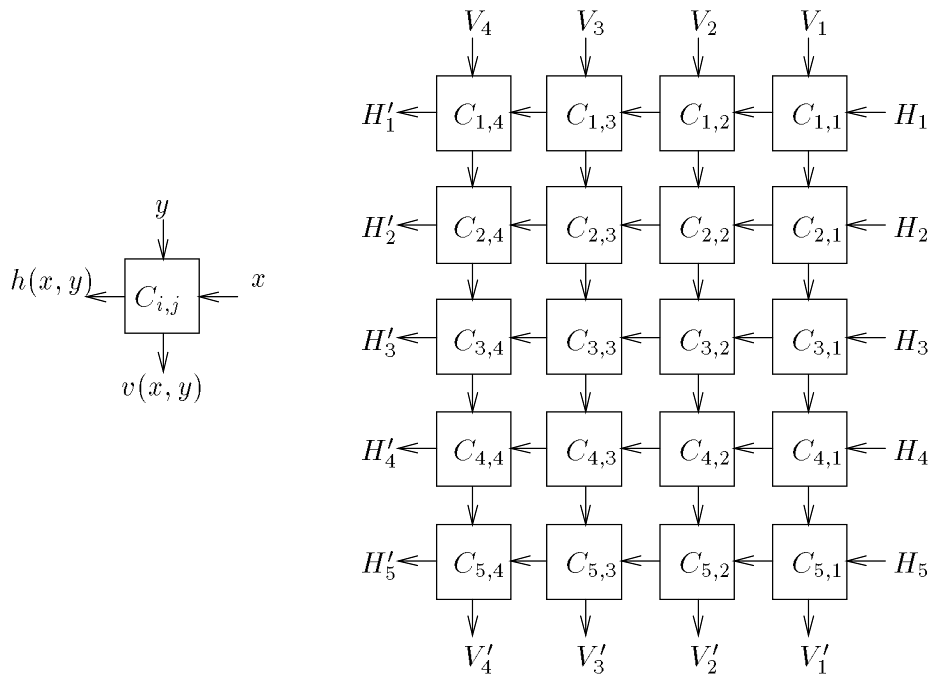
y
$h(x,y)$
$C_{i,j}$
x
$v(x,y)$
V_4 V_3 V_2 V_1
H'_1 $C_{1,4}$ $C_{1,3}$ $C_{1,2}$ $C_{1,1}$ H_1
H'_2 $C_{2,4}$ $C_{2,3}$ $C_{2,2}$ $C_{2,1}$ H_2
H'_3 $C_{3,4}$ $C_{3,3}$ $C_{3,2}$ $C_{3,1}$ H_3
H'_4 $C_{4,4}$ $C_{4,3}$ $C_{4,2}$ $C_{4,1}$ H_4
H'_5 $C_{5,4}$ $C_{5,3}$ $C_{5,2}$ $C_{5,1}$ H_5
V'_4 V'_3 V'_2 V'_1

Figure 1.1: A 5 by 4 array.

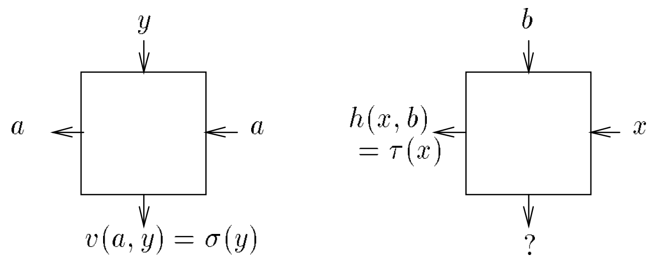

Figure 1.2: Column-separable cells.

implemented by g physical wires then $|Y| = 2^k$ and $|X| = 2^g$. Figure 1.1 also depicts a typical cell.

Definition 1: A cell's function is *column-separable* if there exist $a \in X$ and $b \in Y$ such that

1. $v(a,y)$ is a permutation on Y, $(\sigma(y))$,
2. $h(a,y) = a$,
3. $h(x,b)$ is a permutation on X, $(\tau(x))$,
4. and $v(a,b) = \sigma(b) = b$.

Since $h(a,y) = a$, we have $\tau(a) = h(a,b) = a$. Figure 1.2 illustrates these conditions. Note that any cell can be modified to be column-separable by adding at most one vertical and one horizontal connection.

The values on the internal wires of the array cannot be observed directly. It is possible for two arrays to differ in the values on these wires and still produce the same truth tables. There are ways to observe that these wires have or have not changed value between two tests, since they must encode all possible values. For the sake of brevity and clarity, we adopt the following convention:

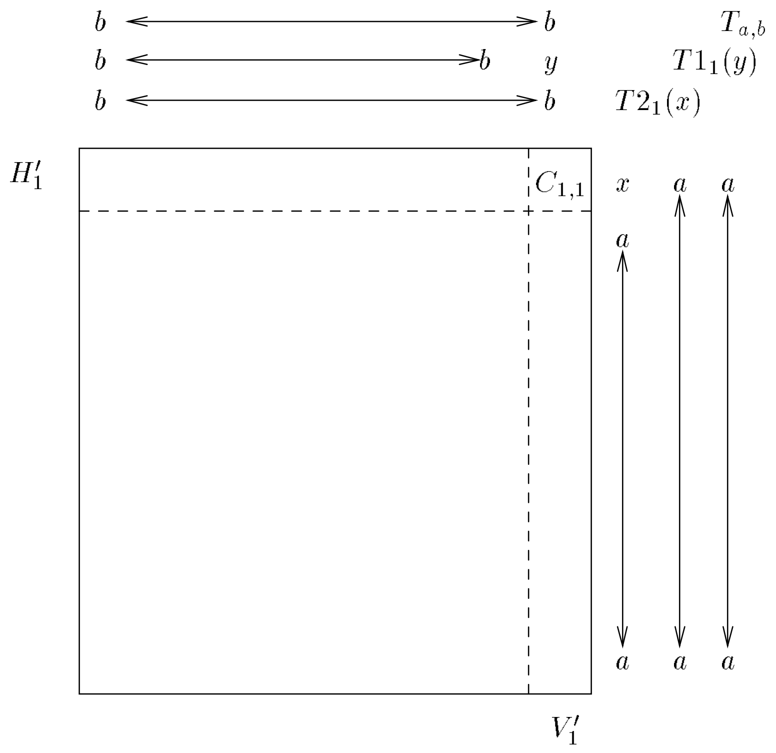

Figure 1.3: Tests for base case: $T_{a,b}$, $T1_1(y)$ and $T2_1(x)$.

Let $T_{a,b} = (b^m, a^n)$ to be the test with a on all row inputs ($H_i = a$ for all i) and b on all column inputs ($V_j = b$ for all j). The internal wires could have any value, but we assign the name 'b' to the value on each internal vertical wire when $T_{a,b}$ is applied and we assign the name 'a' to the value on each internal horizontal wire in $T_{a,b}$.

Thus the notion of value is relative to a wire's value when the array has input $T_{a,b}$.

The tests below check that each cell $C_{i,j}$ (although possibly faulty) is column-separable for a, b, and some permutations $\sigma_{i,j}()$ and $\tau_{i,j}()$. Each cell may have a different pair of permutations, since the encodings on the internal wires can be arbitrary if the cell is faulty. The $m(|Y|-1)+n(|X|-1)+1$ tests are:

$T_{a,b}$: (b^m, a^n),

$T1_j(y)$: $(b^{m-j}yb^{j-1}, a^n)$ for all $y \in Y - \{b\}$ and $1 \leq j \leq m$,

$T2_i(x)$: $(b^m, a^{i-1}xa^{n-i})$ for all $x \in X - \{a\}$ and $1 \leq i \leq n$.

Lemma 1: Suppose C is a column-separable cell. Then the $m(|Y|-1)+n(|X|-1)+1$ tests, $T1_j(y)$, $T2_i(x)$, and $T_{a,b}$, guarantee for each cell $C_{i,j}$ in an $n \times m$ array composed of C, that there are permutations $\sigma_{i,j}()$ and $\tau_{i,j}()$ such that,

1. $v(a,y)$ is a permutation on Y, $(\sigma_{i,j}(y))$,
2. $h(a,y) = a$,
3. $h(x,b)$ is a permutation on X, $(\tau_{i,j}(x))$,
4. and $v(a,b) = \sigma_{i,j}(b) = b$.

Proof: The proof is by induction on the diagonals.

Induction Hypothesis: The cells on the k^{th} diagonal behave as required.

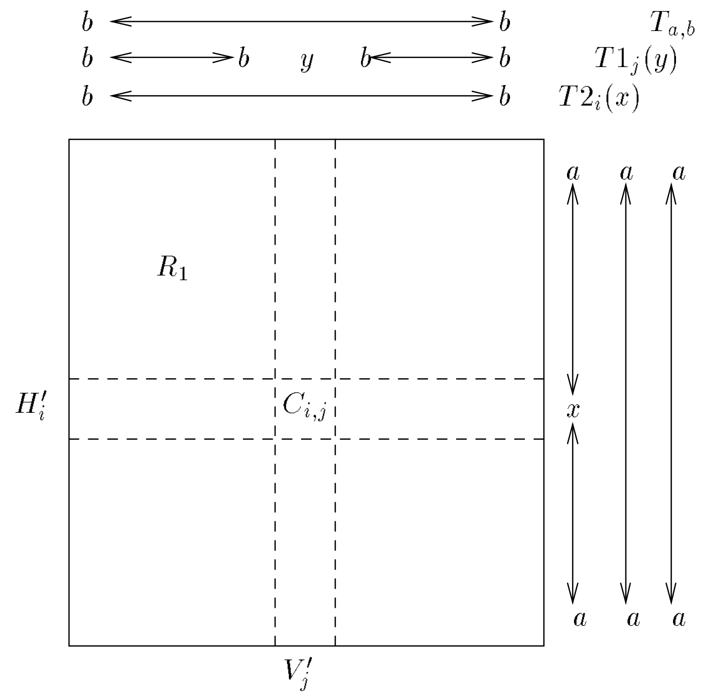

Figure 1.4: Tests for inductive step case: $T_{a,b}$, $T1_j(y)$ and $T2_i(x)$.

Base Case: $k = 1$. In this case, $i = j = 1$ and this cell's inputs are controllable since they are external inputs. We need only verify that the outputs of the cell behave properly. As illustrated in Figure 1.3 the tests $T1_1(y)$ and $T_{a,b}$ hold the row inputs at a, the inputs of columns m through 2 at b, while applying all values in Y to V_1. Since $|Y|$ different values must appear at the V'_1 output, the vertical output of cell $C_{1,1}$ must implement a permutation of Y. By our convention, this cell has the value b on its vertical output wire in $T_{a,b}$ and hence $\sigma_{1,1}$ fixes b ($\sigma_{1,1}(b) = b$). The same argument with tests $T2_1(x)$ and $T_{a,b}$ shows that the horizontal output of cell $C_{1,1}$ must implement a permutation of X and by definition this permutation fixes a. It remains only to show that $h(a, y) = a$. If $h(a, y) \neq a$ for some $y \in Y$, this error will propagate to H'_1 since $T2_1(x)$ generates all row outputs (a permutation of X) and the column inputs to the cells of row 1 in columns 2 through m are all held at b in the $T1_1(y)$ and $T2_1(x)$ tests. (If $h(a, y) \neq a$ this value will be propagated by these $m - 1$ cells in $T1_1(y)$ and appear as a "non-a" value on H'_1 just as in one of the $T2_1(x)$ tests.)

Inductive Step: $k > 1$. Assume that all cells on diagonals numbered less than k behave as required. Consider a cell $C_{i,j}$ on the k^{th} diagonal. As illustrated in Figure 1.4, the tests $T_{a,b}$ and $T1_j(y)$ hold the row inputs at a, apply all values in Y to the column j input, V_j, while holding all other column inputs at b. The cells to the right of column j have the same inputs in $T_{a,b}$ as in all of the $T1_j(y)$ tests. Hence by definition, cell $C_{i,j}$ has a on its row input in all of these tests. Since a permutation of Y must appear on V'_j, the column output of $C_{i,j}$ implements a permutation of Y. Again, by definition this permutation fixes b. The same argument with tests $T2_i(x)$ and $T_{a,b}$ show that the horizontal output of cell $C_{i,j}$ must implement a permutation of X and by definition this permutation fixes a.

It remains only to show that cell $C_{i,j}$'s horizontal output remains a during the $T1_j(y)$ tests. Consider the rectangular region of the array formed by the cells in rows 1 through $i-1$ and columns m through $j+1$ (the region R_1 in Figure 1.4). The column inputs to this region are all b in the $T1_j(y)$ tests. The row inputs to this region are the row outputs of the first $i-1$ cells in column j. By induction these cells hold their row outputs at a since their own row inputs are a in these tests. Thus this rectangular region of the array has the same inputs in tests $T1_j(y)$, $T2_i(x)$, $T_{a,b}$, and so the cells $C_{i,m}$ through $C_{i,j+1}$ have b on their column input in these tests. The $T2_i(x)$ tests show that these $m-j$ cells implement a permutation of X which fixes a when their vertical inputs are held at b. Hence if the row output of $C_{i,j}$ does not remain at a during the $T1_j(y)$ tests, this "non-a" value will be propagated by these $m-j$ cells and appear as a "non-a" value on the row i output, H_i'. □

We can also define the counterpart of "column-separable" for rows.

Definition 2: A cell's function is *row-separable* if there exist $a' \in X$ and $b' \in Y$ such that

1. $h(x,b')$ is a permutation on X, $(\tau'(x))$,
2. $v(x,b') = b'$,
3. $v(a',y)$ is a permutation on Y, $(\sigma'(y))$,
4. and $h(a',b') = \tau'(a') = a'$.

Corollary 1: Suppose C is a row-separable cell. Then the $m(|Y|-1)+n(|X|-1)+1$ tests, $T1_j(y)$, $T2_i(x)$, and $T_{a,b}$, guarantee for each cell in an $n \times m$ array composed of C, that there are permutations $\sigma'_{i,j}()$ and $\tau'_{i,j}()$ such that,

1. $h(x,b')$ is a permutation on X, $(\sigma'_{i,j}(x))$,
2. $v(x,b') = b'$,
3. $v(a',y)$ is a permutation on Y, $(\tau'_{i,j}(y))$,
4. and $h(a',b') = \sigma'_{i,j}(a') = a'$.

A cell which is column-separable for values a and b, and row-separable for values a' and b', is both column-separable and row-separable for the same pair of values, a and b'. The conditions imposed on cells by Prasad and Gray for their $O(mn)$ test set are equivalent to a cell being both column-separable and row-separable[14].

Example: Buffer Array

We consider the problem of testing a two-dimensional array of buffer cells. The behavior of a buffer cell is given by $h(x,y) = x$ and $v(x,y) = y$. This cell is both column-separable and row-separable. Any choice of values for a and b will do. If we pick, $a = 0$ and $b = 0$, then the $m+n+1$ tests are:

$T_{0,0}$: $(0^m, 0^n)$,

$T1_j(1)$: $(0^{m-j}10^{j-1}, 0^n)$ for all $1 \le j \le m$,

$T2_i(1)$: $(0^m, 0^{i-1}10^{n-i})$ for all $1 \le i \le n$.

These tests verify the truth table of each cell to be:

x	y	h	v
0	0	0	0
0	1	0	1
1	0	1	?
1	1	?	?

where ? is an unknown value and 0 is assigned to the value on a wire in test $T_{0,0}$. Note that this array is also row-separable with the same choice of values $a = 0$ and $b = 0$. Applying Corollary 1 allows us to conclude that $v(1, 0) = 0$. One last test, $T_{1,1} = (1^m, 1^n)$ is required to complete the truth table of the cell. Clearly, if any wire retains its 0 value in $T_{1,1}$, then the remaining column and/or row will transmit this 0 to an external output since the previous tests verified the outputs of all cells for the inputs 00, 01, and 10. This gives us a test set of size $m + n + 2$.

We can show that this is a lower bound on the number of tests required as follows. Consider a one-dimensional array of m buffer cells. Let F be any subset of the cells and consider the faulty array in which the intercell output of each cell in F is $h(x, y) = x \oplus y$ instead of x. Any test which applies an even number of 1's to the cells in F, will not detect this fault. An even number of inversions will occur and will not be detected. We shall show that if there are more cells than tests we can always find such a subset of cells, F.

If there are m cells, there are 2^m different subsets of cells. Suppose there are k tests $T_1, \ldots, T_k$. For any given subset S we can associate with it a vector $\vec{v}(S) = (v_1(S), v_2(S), \ldots, v_k(S))$ where $v_i(S)$ represents the parity of the number of 1's applied to cells in S in test T_i. There are 2^k different vectors. If $m > k$, then $2^m > 2^k$ and there are more subsets than vectors. This means we can find two distinct subsets S_1 and S_2 such that $\vec{v}(S_1) = \vec{v}(S_2)$. The non-empty subset $F = (S_1 - S_2) \cup (S_2 - S_1)$ has $\vec{v}(F) = (0, 0, \ldots, 0)$. Hence if $m > k$, then for any k tests we can find a subset of cells F such that each one of the k tests applies an even number of 1's to the cells in F. This test set then fails to detect the faulty array constructed from F.

We can apply this argument to both the rows and columns of a two-dimensional array of buffers to show that at least $\max\{n, m\}$ tests are required. Asymptotically, the number of tests required for a two-dimensional array of buffer cells is no greater than for a one-dimensional array under the MFC model.

The purpose of verifying the "column-separability" of each cell is to ensure that the inputs to a row can be controlled from the primary inputs of the array and that any faults in the row that modify the truth-table of the array will be observed at the primary outputs. Having verified the column-separability of all cells, it then suffices to apply tests to verify the rows.

Theorem 1: If a cell C is column-separable, then under the MFC model the truth table of an $n \times m$ array composed of C can be verified in $m(|Y| - 1) + n(|X| - 1) + 1 + nR(m)$ tests where $R(m)$ is the number of tests sufficient to verify the truth table of a one-dimensional array of m cells.

Proof: As discussed, the signals on the internal wires can be encoded in some arbitrary manner. As long as adjacent cells agree on an encoding for their wires, the array can still function correctly. As before to facilitate the argument, we make the convention that for an internal horizontal wire the name a is assigned to the value on this wire in $T_{a,b}$. For an internal vertical wire, the name b is assigned to the value on the wire in $T_{a,b}$. Each cell must communicate the values in Y to its neighbor below in the array and the values in X to its neighbor on the left. We make the convention that the encoding of Y on the vertical output of cell $C_{i,j}$ is determined by $T1_j(y)$ and corresponds to $\sigma^i(y)$ (the result of applying $\sigma()$ i times to y), where $\sigma(y)$ is the permutation $v(a, y)$. That is, the value on this vertical wire during $T1_j(y)$ is the encoding of $\sigma^i(y)$ for this wire. Similarly, the value on $C_{i,j}$'s horizontal wires during the $T2_i(x)$ test is the encoding of $\tau^j(x)$. Thus encodings on the internal wires are defined by the cells generating them.

Let T_R denote the $R(m)$ tests required to verify the truth table of a row of m cells under the MFC model. Then the tests are,

$T_{a,b}$: (b^m, a^n),

$T1_j(y)$: $(b^{m-j}yb^{j-1}, a^n)$ for all $y \in Y - \{b\}$ and $1 \leq j \leq m$,

$T2_i(x)$: $(b^m, a^{i-1}xa^{n-i})$ for all $x \in X - \{a\}$ and $1 \leq i \leq n$,

$T3_i(r)$: $(\sigma^{1-i}(\vec{r_y}), a^{i-1}r_x a^{n-i})$ for all $1 \leq i \leq n$, where the test $r \in T_R$ has column inputs $\vec{r_y}$ and row input r_x, and $\sigma^{1-i}(\vec{r_y})$ is the result of applying σ^{1-i} to each entry in r_y.

By Lemma 1 we know that the tests $T1_j(y)$, $T2_i(x)$ and $T_{a,b}$ guarantee that each cell is correct for any input involving either a or b. Our convention about the encodings on the internal wires ensures that each cell implements $\sigma()$ on its vertical output and that its horizontal output is a when its horizontal input is a.

We shall show that each row implements a correct one-dimensional array according to its column input/output encoding conventions. Consider the tests $T3_i(r)$ for all $r \in T_R$. The input to cell $C_{i,j}$ in $T3_i(r)$ is the encoding of $\sigma^{i-1}(\sigma^{1-i}(r_y^i)) = r_y^i$ according to the conventions since all horizontal wires except for those in row i are a in these tests. Hence, the values on the column inputs of the cells in $T3_i(r)$ are the properly encoded values of r_y. The row input is set directly to r_x. The row output is directly observable, so we only need show that the column outputs are correct. Suppose the column output of $C_{i,j}$ in $T3_i(r)$ is the encoding of y' instead of the encoding of y, the correct output. Then the cells in column j map y' to $\sigma^{n-i-1}(y')$ which appears at the column j output, V_j'. Since the V_j' should be $\sigma^{n-i-1}(y)$, an error will be detected.

Hence each row implements a correct one-dimensional array with respect to the encoding conventions. Since the encoding conventions on the external inputs and outputs of the array are observable, they coincide with the correct ones if no error is detected. If there is an input which causes an erroneous output, there has to be a first row in which the row's output is incorrect with respect to its encodings. Since we have ensured that this cannot be the case, the truth table of the array is verified. □

The analogous result for row-separable cells is:

Corollary 2: If a cell C is row-separable, then under the MFC model the truth table of an $n \times m$ array composed of C can be verified in $m(|Y|-1) + n(|X|-1) + 1 + mC(n)$ tests where $C(n)$ is the number of tests required to verify a one-dimensional array of n cells.

As mentioned earlier, any cell can be made column or row-separable by adding at most one physical wire in either connection. If there are "don't care" inputs in the cells definition, these may be used to obtain the required a and b values without additional intercell wires. We next focus on the problem of testing one-dimensional arrays for multiple faults in a constant number of tests. If $R(m)$ is independent of m, Theorem 1 provides an $O(n+m)$ test set.

Example: 2-D Array of Full Adders

The heart of an array multiplier is an array of full adders. This is the case for the carry-propagate array multiplier, the carry-save array multiplier, and the Booth multiplier. An $n \times m$ array of full adders is column separable with the values $b = 1$ and $a = 0$ assuming the product inputs of each full adder are held at 0.

If we pick, $a = 0$ and $b = 1$, then the $m + n + 1$ tests required in Lemma 1 are:

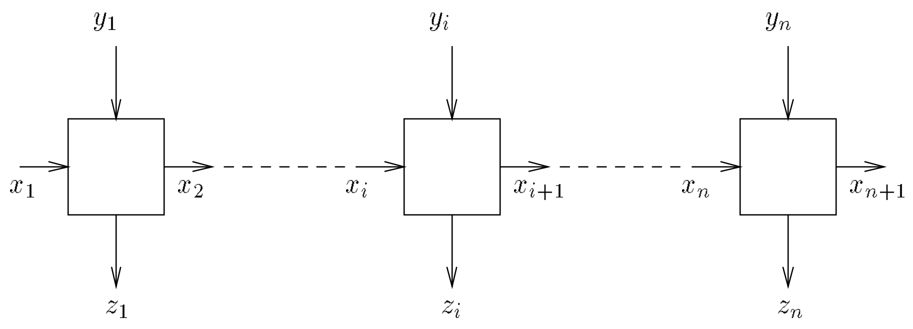

Figure 1.5: A general one-dimensional ILA.

$T_{0,1}$: $(1^m, 0^n)$,

$T1_j(0)$: $(1^{m-j}01^{j-1}, 0^n)$ for all $1 \le j \le m$,

$T2_i(1)$: $(1^m, 0^{i-1}10^{n-i})$ for all $1 \le i \le n$.

These tests verify the truth table of each full adder to be:

y	p	x	c	s
0	0	0	0	0
1	0	0	0	1
1	0	1	1	?

where ? represents an unknown value and 0 is the name assigned to the value on all horizontal wires in the test $T_{0,1}$, while 1 is the name assigned to all values on the vertical wires in $T_{0,1}$.

Cheng and Patel obtain a minimum test set under the MFC model for the ripple-carry adder with only 11 tests[15]. Applying the Cheng and Patel tests to each row requires only $8n + 3$ tests since three of the tests are the same for all rows. The total number of tests is $m + n + 1 + 8n + 3 = 9n + m + 4$. Note that these tests require that we observe the carry outputs of the full adders in the last column. In a multiplier, these final carries are combined by an adder to obtain the product. In Chapter 2, we show how to test a multiplier array even though its final carries are not directly observable.

1.1.2 Detection of all Multiple Faults in 1-D ILAs

We now restate and prove Dias' Theorem 3. We do this to show how a constant size test set for a one-dimensional array can be derived, and to provide a simpler proof for the theorem found in [12]. Whenever possible we use the notation and definitions in [12].

Figure 1.5 shows an n-cell one-dimensional array. The inputs $y_1, \ldots, y_n$ and x_1 are directly controllable, and the outputs $z_1, \ldots, z_n$ and x_{n+1} are directly observable. Each cell in an ILA can be viewed as a sequential circuit in which the x is the state, y the input, and z the output. A n-cell ILA can then be viewed as a sequential circuit with the input sequence $y_1, y_2, \ldots, y_n$, output sequence $z_1, z_2, \ldots, z_n$, initial state x_1, and final state x_{n+1}. To continue this analogy, a flow table which specifies the output and next state based on the input and present state can be generated for the ILA's basic cell. We use the ILA/sequential circuit analogy to provide a language and concepts from checking experiments in sequential circuits. One such useful concept is a *set of identifying sequences* (SIS).

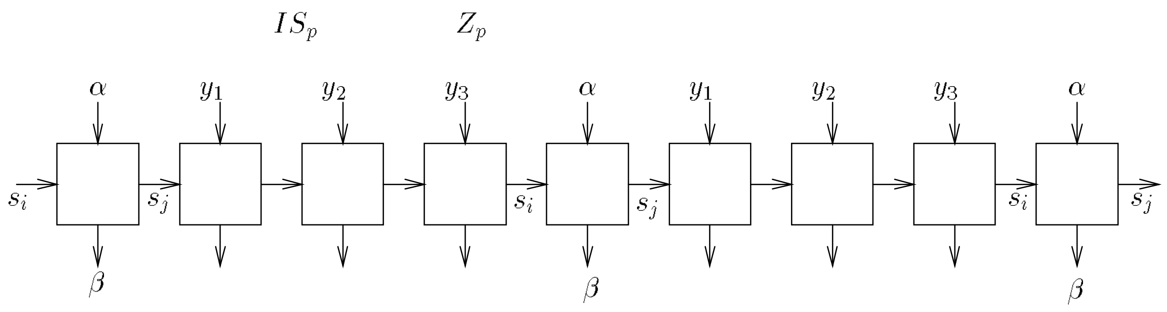

Figure 1.6: A loop test of length four.

Definition 3: An SIS is a set of input sequences (IS) $\{IS_1, IS_2, \ldots, IS_r\}$ such that for any two states s_i and s_j there is an IS_p which produces different output sequences for these two states. If $\{IS_1\}$ is an SIS, then IS_1 is called a *distinguishing sequence* since its application alone will determine the state of the machine when it was applied.

Definition 4: A *test* is denoted by $(s_i, \bar{J})$ where $\bar{J}$ denotes a sequence of inputs of length n for $y_1, y_2, ...y_n$ formed by repeating the sequence J.

Let

$$t : s_i \overset{\alpha/\beta}{\rightarrow} s_j$$

be a transition in the flow table of the basic cell with present state s_i, and input α, which produces the next state s_j and the output β.

Definition 5: A loop test for transition $t : s_i \overset{\alpha/\beta}{\rightarrow} s_j$ with IS_p, is the test $L_p(t) = (s_i, \bar{J})$ where,

1. J is the concatenation of α, IS_p, and Z_p,
2. IS_p is in the SIS, and
3. the input sequence Z_p drives the flow table back to state s_i from the state resulting from the application of αIS_p to s_i.

The length of $L_p(t)$ is $\mid L_p(t) \mid = 1 + \mid IS_p \mid + \mid Z_p \mid$.

Figure 1.6 shows an ILA with a loop test of length 4. The IS_p consists of the sequence y_1, y_2 and the Z_p consists of the one single input sequence y_3. If the flow table of the basic cell of an array is reduced then there is always an SIS, and if the components of the flow table are all strongly connected then we can always find an input sequence Z_p to form a loop test for any transition and IS_p. These are the two requirements needed to guarantee the existence of $L_p(t)$ for every transition.

Definition 6: A cell C is said to be *Dias-testable* if the flow table obtained from its functions $(h(x, y), v(x, y))$ by considering its horizontal input to be its state, is reduced and has only strongly connected components.

The modification to make a cell column-separable can at the same time ensure that the cell meets Dias' requirements, by making the permutation corresponding to $v(x, b)$ cycle through all of the states. This ensures that its flow table is strongly connected. If the flow table is subsequently reduced, then the resulting cell will still be column-separable and satisfy Dias' requirements.

We now restate Procedure 1 from Dias [12].

Procedure 1: Consider an ILA whose basic cell has M states and N possible input vectors and is Dias-testable. Let the transitions in this flow table be labeled as $t_1, t_2, \ldots, t_{MN}$, and the ISs in the SIS chosen for testing be labeled as $IS_1, IS_2, \ldots, IS_r$. A test set for this array can be generated as follows.

```
For $i = 1$ to $MN$, do
    For $j = 1$ to $r$, do
        For $k = 0$ to $|L_j(t_i)| - 1$, do
            Apply $k$ shifts of $L_j(t_i)$ to the array
        End
    End
End
```

The test set derived by Procedure 1 completely exercises each cell in the fault-free array. It is shown in Theorem 2 that this test set is sufficient for verifying the truth table of the array under the MFC model.

Theorem 2: For a Dias-testable cell, the test set derived by Procedure 1 detects all faults under the MFC model that change the truth table of the array.

Proof: As discussed earlier, it cannot be shown that each cell is 'correct' (implements the cell's flow table exactly) since the array would still function correctly (have the same truth table) if the adjacent cells were to agree on an encoding of the states other then the one used in the flow table. Instead we shall verify the truth table of the first i cells of the array for some encoding of the values on x_{i+1}. Since we can directly observe x_{n+1} we can verify that its encoding is the same as a 'correct' cell's.

Induction Hypothesis: Cells 1 through i each implement the correct flow table where the output x_{i+1} is encoded by some permutation of the states.

Base Case: $i = 1$. In this case, all inputs are directly controllable. The z_1 output is exhaustively tested by the 0-shifts of all of the $L_p(t)$ tests. Since the graph is strongly connected there is a transition into every state. If the x_2 signal does not implement some encoding (permutation) of the correct states, then either there are two transitions into some state s_j which result in different values on x_2 or there are transitions into two distinct states s_j and s_k which result in the same value on x_2. Since there is a transition into each of the M states, the former implies the latter by the pigeonhole principle. So in either case, there is some transition $t : s_h \overset{\alpha/\beta}{\rightarrow} s_j$ for which x_2 assumes the same value as for $t' : s_g \overset{\gamma/\omega}{\rightarrow} s_k$ with $s_j \neq s_k$. In this case, the loop test $L_q(t)$ where IS_q is the sequence which can distinguish between s_j and s_k will produce the same output as $L_q(t')$ which will be detected as an error.

Inductive Step: $i > 1$. Assume that the cells numbered from 1 to $i - 1$ form an array which behaves as required. There are two ways in which the first i cells could fail to form an array which behaves as required.

First, suppose there is an input $(s_h, (\alpha_1, \ldots, \alpha_i))$ which does not give the correct $\beta_1, \ldots, \beta_i$ output. By induction the first $i - 1$ cells form a 'correct array'; the z_i output is the only one that could be wrong. Suppose s_k is the state resulting in the application of $\alpha_1, \ldots, \alpha_{i-1}$ to s_h and consider the loop test for the transition $t : s_k \overset{\alpha_i/\beta_i}{\rightarrow} s_j$. Since the first $i - 1$ cells perform some encoding of the states, any loop test for t shifted so that cell i receives the start of the sequence, would result in the same x_i output as $(s_h, (\alpha_1, \ldots, \alpha_{i-1}))$ (the encoding of s_k). Hence if z_i is incorrect in $(s_h, (\alpha_1, \ldots, \alpha_i))$ it will be incorrect in the appropriate shift of any loop test for t.

The second manner in which the array could fail is if the output x_{i+1} does not result in an encoding of the states from the flow table. As in the base case, this can only happen if it produces the same value for two transitions into distinct states. Again, the loop tests for these two transitions with an IS which can distinguish between the two resulting states, will produce the same output when different outputs are required. □

Example: Ripple-Carry adder

A full adder is the basic cell of a ripple-carry adder. Any input for the full adder is a distinguishing sequence forming an SIS with a single IS. However, choosing $y = 01$ (or 10) provides a next state equal to s_i, which makes the Z_1 sequence unnecessary if the transition is to the same state. Dias makes this observation in [12], and produces a sequence of length 8 which exercises all 8 transitions. He claims that this test and its 7 shifts are sufficient. Unfortunately, in the proof of Theorem 2 it is important that the same IS's be used in all the loop tests to determine that x_i's properly encode the states. The proof hinges on the observation that the rest of the (possible faulty) array cannot produce different outputs for the same input. Different IS's would detect an error if they were applied to correct cells, but might fail to produce an error on faulty cells. Aboulhamid pointed out this error with a faulty array which was not detected by Dias' tests, and provided a test set that detects all multiple faults in a one-dimensional array of full adder cells in 16 tests [13]. We apply Dias' theorem to obtain a test set of 14 vectors. Instead of creating, one sequence with all 8 looptests separated by an SIS which must be shifted 15 times, we provide 6 sequences as described in Procedure 1 which must each be shifted once and another sequence which must be shifted three times. This latter test combines the loop tests for the two transitions which change the state. As a result, we have two loop tests which apply $y = 10$ to all cells (one with $x = 0$ and one with $x = 1$). The shifts of these tests are the same tests, giving us a total of 14 tests. Table 1.1 contains the 14 tests that verify the truth table of the ripple-carry adder under the MFC model. The *pattern* listed in each test should be repeated to form an input of size n.

test	x_1	*pattern*	test	x_1	*pattern*
t^{0}_{00}	0	00,10	t^{1}_{11}	1	11,10
t^{0s}_{00}	0	10,00	t^{1s}_{11}	1	10,11
t^{0}_{01}	0	01,10	t^{1}_{01}	1	01,10
t^{0s}_{01}	0	10,01	t^{1s}_{01}	1	10,01
t^{0}_{10}	0	10,10	t^{1}_{10}	1	10,10
$t^{0}_{11,00}$	0	11,10,00,10	$t^{1}_{11,00}$	1	10,00,10,11
$t^{2}_{11,00}$	1	00,10,11,10	$t^{3}_{11,00}$	0	10,11,10,00

Table 1.1: Test set for ripple-carry adders.

Cheng and Patel [15] obtain a smaller test set of 11 patterns by using different IS's in the looptests but with tests which verify that different IS's applied to a cell are operating with the same encoding convention. They show that 11 is the minimum number of tests required to verify a row of full adders under the MFC model.

2. Testing an array multiplier under the MFC model

In this section we apply the techniques from the previous section to a carry save array multiplier. The test set must be modified and augmented to handle the non-orthogonal nature of the multiplier array and the fact that the array inputs are not independently controllable. An $r \times q$ array multiplier has two parts: the summand generator which consists of rq AND gates and generates rq one-bit products, and the summand counter which adds these to generate an $r+q$ bit product. This is shown in Figure 2.1. The summand generator can be organized as a two-dimensional ILA of AND gates and the summand counter can be organized as a two-dimensional ILA of adder cells.

The summand counter for a carry save array multiplier can be implemented as a two-dimensional rectangular ILA of full adders and a ripple carry adder combined as shown in Figure 2.2. The hybrid ILA for the carry save multiplier summand counter (CSM-SC) of Figure 2.2 can be redrawn as shown in Figure 2.3. It consists of two subarrays, a two-dimensional array of full adder cells and a ripple carry adder, also composed of full adder cells. We shall refer to these two portions as the FA array and the RCA, respectively. Since the summand counter for a $r \times q$ multiplier has r x-inputs and $r+q$ y-inputs, we will call it an $r \times (r+q)$ CSM-SC. The n rows to an $n \times m$ CSM-SC are numbered by the subscript of the row's x-input, and the m columns by the subscript of the column's y-input.

The vertical, product, and horizontal inputs to the cells of the FA array component of the CSM-SC are called the y, p, and x inputs, respectively. The horizontal (or carry) and the vertical (or sum) outputs of the individual cells are called the c and s outputs, respectively. The inputs are ordered from left to right on the cells in Figure 2.3 as $y\,p\,x$ and the outputs as $c\,s$.

The cells in the n bit RCA of the CSM-SC are numbered by row as they appear in the CSM-SC array. Thus the cell with the most significant bit product is C_1. The carry and sum outputs of each cell in the ripple carry adder are the c and s outputs, respectively, as

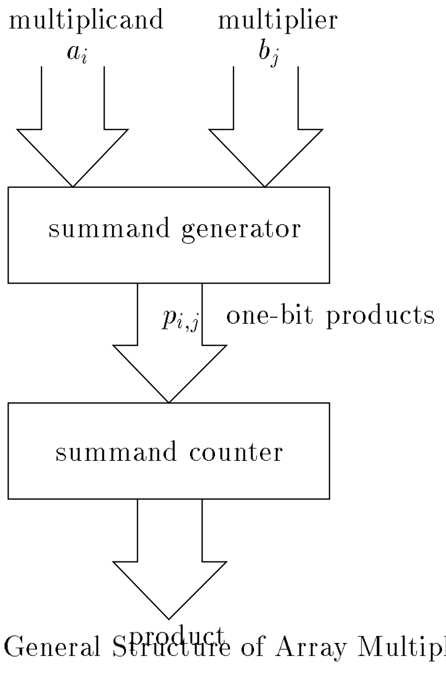

Figure 2.1: General Structure of Array Multiplier

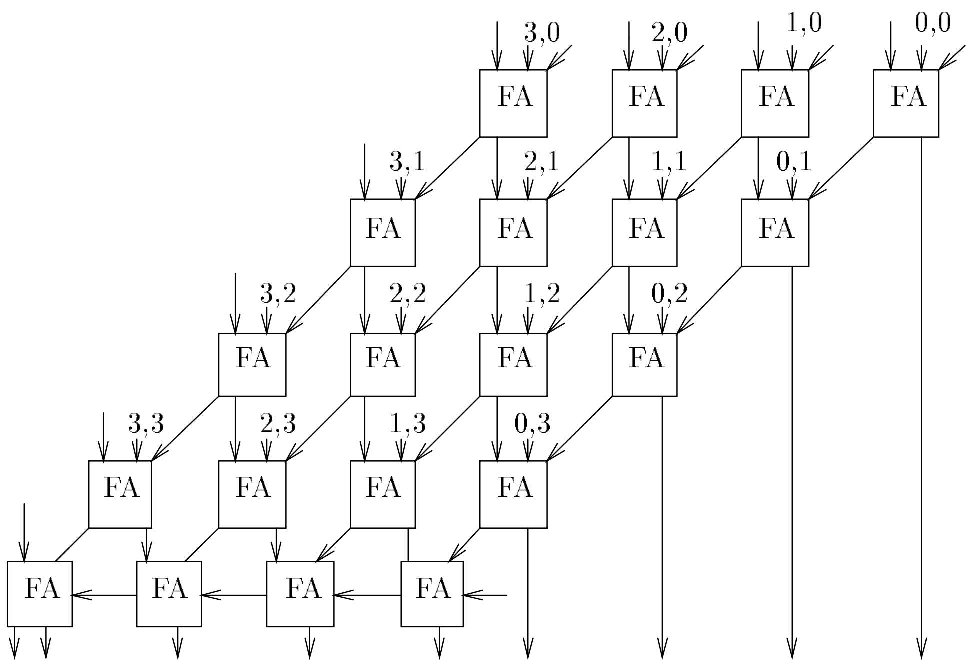

Figure 2.2: Summand counter for the 4×4 carry save multiplier.

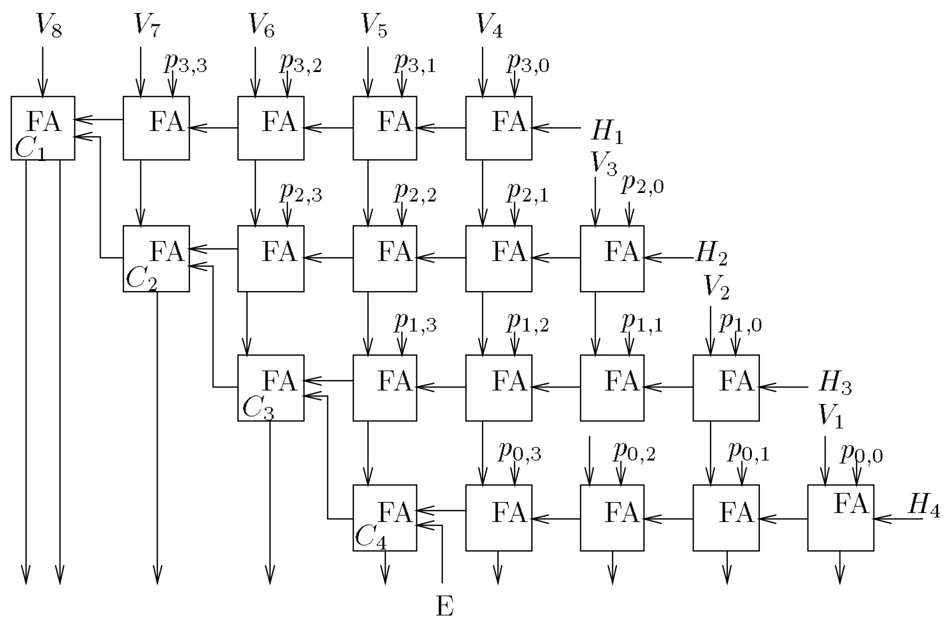

Figure 2.3: Redrawn summand counter for the 4x4 carry save multiplier.

with the FA cells. The vertical (or sum) inputs to the individual cells are the y inputs, the x input to cell j is the c output of cell $j+1$, or the array input E if $j=n$, and the p input is the c output of the leftmost cell of row j in the FA array.

There are two problems in applying the test set of Theorem 1 for the full adder array to the CSM-SC. The first is that the p terms are not independently controllable, but are functions of the multiplier and multiplicand. The second potential problem is the RCA subarray. The cells in the RCA have a signal flow opposite in direction to the cells in the FA array. Since the argument used in Lemma 1 requires the observation of the row outputs, the test set derived for the full adder array may not be valid. Fortunately, the full adder array test set from Chapter 1.1 can be augmented and modified to overcome these difficulties as described in Chapter 2.0.3. Chapter 2.1 shows that there is no constant size test set under the MFC model for the array multiplier by developing an $\Omega(n/\log n)$ lower bound on the number of tests required to verify the full adder array of an $n \times n$ multiplier. Chapter 2.2 presents a cell modification and test set to handle the incorporation of the summand generator (the AND gates) into the cells while still testing the array with a test set of size proportional to its perimeter.

2.0.3 Testing the Summand Counter

As discussed there are two problems in applying the test set of Theorem 1 for the full adder array to the CSM-SC. The first is that the p terms are not independently controllable, but are functions of the multiplier and multiplicand. Although not independently controllable, all p terms in a set of rows can be forced to 0 in the FA array by placing 0's in the bit positions of the multiplicand corresponding to those rows. All other rows have the bit pattern of the n bit multiplier shifted to the right as the row number increases. This is sufficient to test the CSM-SC.

The second potential problem is the RCA subarray. The direction of signal flow of the c output of the RCA is in the opposite direction of the s outputs of the FA array. This makes it difficult to prove that the c output of a cell in the FA array is a 0 when its inputs are 100; it invalidates the argument used in Lemma 1 because there are paths from the sum output of a cell back to the RCA cell of the same row which could allow a fault to be masked. The solution to this problem is to add additional tests to verify the truth table of the ripple carry adder under the MFC model.

The $8n$ tests, $T3_j^{000-111}$ for all $1 \leq j \leq n$ shown in Table 2.1 when added to $T1_j$, verify the truth table of the RCA in the $n \times m$ CSM-SC. In the following, $T3_j^{ypx}$ is the test which applies ypx to the inputs of C_j of the RCA, $\vec{X}$ is an n-bit vector $[X_1, \ldots, X_n]$, $\vec{Y}$ is an m-bit vector $[Y_m, \ldots, Y_1]$, and $\vec{P}$ is an n-bit vector, $[p_1, \ldots, p_n]$ where p_i denotes the value to be applied to all p inputs of the cells in row i. Hence the values being applied to the product terms are consistent with a multiplier of all 1's and $\vec{p}$ as the multiplicand.

Note that the RCA cell C_j is in column $m-j+1$ and so unlike the $T1_j$ tests, the $T3_j^{ypx}$ tests toggle (if any) column $m-j+1$, not column j. We make the same convention about encodings on internal wires as in Chapter 1.1.2. That is,

> For each wire assign the name '0' to the value on that wire when all external inputs to the array are 0, and the name '1' to the other value that the wire can hold.

Lemma 2: If the CSM-SC passes all $T1_j$ tests of the FA array and the $T3_j$ tests for all $1 \leq j \leq n$, then the truth tables of the cells in the RCA are verified under the MFC model.

$T3_j^{ypx}$	$\vec{X}$	$\vec{Y}$	$\vec{P}$	E
$T3_j^{000}$	$\vec{0}$	$\vec{0}$	$\vec{0}$	0
$T3_j^{001}$	0^j1^{n-j}	$\vec{0}$	0^j1^{n-j}	1
$T3_j^{010}$	$0^{j-1}10^{n-j}$	$\vec{0}$	$0^{j-1}10^{n-j}$	0
$T3_j^{011}$	$0^{j-1}1^{n-j+1}$	$\vec{0}$	$0^{j-1}1^{n-j+1}$	1
$T3_j^{100}$	$\vec{0}$	$0^{j-1}10^{m-j}$	$\vec{0}$	0
$T3_j^{101}$	0^j1^{n-j}	$0^{j-1}10^{m-j}$	0^j1^{n-j}	1
$T3_j^{110}$	$0^{j-1}10^{n-j}$	$0^{j-1}10^{m-j}$	$0^{j-1}10^{n-j}$	0
$T3_j^{111}$	$0^{j-1}1^{n-j+1}$	$0^{j-1}10^{m-j}$	$0^{j-1}1^{n-j+1}$	1

Table 2.1: $T3_j$ tests for $1 \leq j \leq n, m$ verifying the RCA in the CSM-SC.

Proof: The first step of the proof is establishing that each cell in the CSM-SC outputs 00 and ?1 for the inputs 000 and 100 respectively. This is verified for column j by the tests $T1_j$ and $T1_{j-1}$ which toggle y_j while holding all x and p inputs at 0 as well as the y inputs to the right of column j. Each cell in column j has its 000 input in $T1_j$ and the only input changing in $T1_{j-1}$ which can affect column j is y_j. For the final column j output to be 1 in $T1_{j-1}$, each cell in column j must have changed its sum output to 1. The rest of the proof is by induction on the cells in the RCA part of the CSM-SC.

Induction Hypothesis: The RCA cells with indices less than j behave as full adders based on the encodings defined on their inputs.

Base Case: $j = 1$. In this case, all outputs are directly observable and the y input is directly controllable, so all that must be shown is that the tests $T3_1^{abc}$ apply bc to the px inputs of C_1.

$T3_1^{000}$ Input vector is 000 by definition.

$T3_1^{001}$ Due to the direction of signal flow, the only input bit that can differ from the inputs of $T3_1^{000}$ is x. Since s changed, x must have changed.

$T3_1^{010}$ Since C_1's output in this test differs from its output for $T3_1^{000}$, its ypx inputs must be either 001, 010, or 011. If its inputs were 001 or 011 then a y output of one of the first row of cells of the FA array must be 1; otherwise the cells below the row would have all 0's and the x input to C_1 could not be 1. Suppose there is such an s output, and let s_k be the rightmost one. The inputs to column k are all 0's below the first row (since s_k was the rightmost non-zero output of the first row). By our earlier argument, the cells in column k starting from the second row will propagate a 1 down the column to the external output. But since the outputs all remained at 0 (with the exception of C_1), we know that no s output of a cell in the first row changed to 1 in $T3_1^{010}$. Hence C_1's inputs are 010 for this test.

$T3_1^{011}$ Since cell 1's output for this test differs from those for tests $T3_1^{000}$, $T3_1^{001}$, and $T3_1^{010}$, its input vector must be 011.

$T3_1^{100-111}$ Since these tests differ from the above only by a single primary input which cannot travel to any other cell in the array, they the same arguments can be used as for their 0-counterparts.

Inductive Step: $j > 1$. Assume that all RCA cells numbered less than j are correct. The s output of cell j is directly observable, and the c output is an input to the correctly functioning cell $j-1$.

$T3_j^{000}$ Input vector is 000 by definition.

$T3_j^{001}$ Due to the direction of signal flow, the only input of C_j that can differ from the inputs of $T3_j^{000}$ is the x input. Since the s output changed, this input bit changed. Since cell $j+1$ behaves correctly and its other inputs remained at 0 (due to the direction of signal flow), cell j's c output is correct.

$T3_j^{010}$ The s output of cell j differs between this test and $T3_j^{000}$. Due to the direction of signal flow, the only inputs to cell C_j that could have changed from $T3_j^{000}$ are its x and p inputs. By the same argument as in the base case, we know that no s output of row j is 1 since the rightmost such 1 would have been observed on an external s output. Hence all inputs below row j are 0 and so the x input to C_j must be 0. This leaves 010 as the only possibility for its inputs. Since all inputs above row j are 0 and C_{j-1} is functioning correctly, the c output of C_j remains at 0.

$T3_j^{011}$ The direction of signal flow guarantees that C_{j-1}'s y and p inputs are 0, thus sensitizing the c output of C_j. So both outputs of C_j are observable (s directly and c as the s output of C_{j-1}). By the same argument as in the base case, the inputs to C_j must be 011 since its output must differ from those for its 000, 001 and 010 inputs.

$T3_j^{100-111}$ These differ from the cases above by a single primary input. We would like to use the same arguments as for the cases above, however we are no longer assured that the inputs above row j remain at 0 for these tests. The problem is that the 1 which is applied down the column to reach C_j's y input, could possibly travel to any cell to the left of this column as well. However, the cells C_1 through C_{j-1} are correct. If any input to one of these cells were 1, we would observe a non-zero output on one of the sums or the final carry output. In particular we know that C_{j-1}'s inputs are 000 in $T3_j^{100}$. The arguments made for $T3_j^{000-011}$ can then be repeated for $T3_j^{100-111}$. $\square$

Because of the ripple carry adder, we cannot use the same argument as in Lemma 1 to verify column separability. We can no longer directly observe the row output of the FA array in order to verify that the carry output of a cell remains at '0' when only its column output is toggled. However, we will show that if it does toggle we will detect an error somewhere to the left of the column. The tests derived from Lemma 1 for the full adder array are:

$T_{0,1}$: $(1^m, 0^n)$,

$T1_j(0)$: $(1^{m-j}01^{j-1}, 0^n)$ for all $1 \leq j \leq m$,

$T2_i(1)$: $(1^m, 0^{i-1}10^{n-i})$ for all $1 \leq i \leq n$.

In order to incorporate a test which toggles each column while holding all inputs to the right of the column at 0, we modify the $T1_j(0)$ tests as follows:

$T1_j(0)$: $(1^{m-j}0^j, 0^n)$ for all $1 \leq j \leq m$,

For brevity we will refer to these tests as:

$T1_j$: $(1^{m-j}0^j, 0^n)$ for all $0 \leq j \leq m$,

$T2_i$: $(1^m, 0^{i-1}10^{n-i})$ for all $1 \leq i \leq n$.

Lemma 3: Assume cells $C_1, \ldots C_n$ are correct and that each cell in the FA array has outputs $cs = 00, ?1$ for inputs $xpy = 000, 001$. Then the $m+n+1$ tests, $T1_j$ and $T2_i$, guarantee that the outputs for each cell in the $n \times m$ FA array are $cs = 00, 01, 1?$ for the inputs $xpy = 000, 001, 101$ where ? is unknown.

Proof: The proof is by induction on the diagonals. Since in all the tests, the product inputs remain at 0, we shall omit them from our discussion as in Lemma 1.

Induction Hypothesis: The cells on the k^{th} diagonal behave as required.

Base Case: $k = 1$. In this case, $i = j = 1$ and this cell's inputs are controllable since they are external inputs. We need only verify that the outputs of the cell behave properly.

The tests $T1_0$ and $T1_1$ hold the row inputs at 0, the inputs of columns m through 2 at 1, while toggling input of column 1. In $T1_1$, the inputs to $FA_{1,1}$ have their 0 values and the only input that changes in $T1_0$ is the sum input to $FA_{1,1}$. The sum output of $FA_{1,1}$ must change in order to realize the change from 0 to 1 on the column 1 output S_1 since the inputs to the remaining cells in column 1 all remain constant in these two tests. Hence the sum output of $FA_{1,1}$ must change its value in $T1_0$ from the '0' value it has in $T1_1$; it is '1' in $T1_0$.

Now consider the tests $T1_0$ and $T2_1$. The cell C_1 should have outputs $cs = 01, 10$ respectively in these two tests. Since its y input is held at 1 during these two tests and C_1 is known to be correct, its horizontal inputs must both be '0' in $T1_0$ and exactly one of the two must be '1' in $T2_1$. Only the x input to $FA_{1,1}$ changes in these two tests, but there are now paths from the two outputs of $FA_{1,1}$ to the cell C_2, so it is possible for the carry output of C_2 to be '1' instead of the p input to cell C_1. However, we will show that all of the sum outputs of the full adders in row 1 are at their '0' values in $T2_1$. Suppose $FA_{1,j}$ is the first full adder in row 1 whose sum output is not '0' in $T2_1$. Since the full adders in rows 2 through n and columns 1 through $j-1$ have their '0' inputs, the sum output of $FA_{1,j}$ is propagated down column j, and appears at the sum output for column j. Hence the sum outputs of the full adders in row 1 are at their '0' values in $T2_1$; otherwise the first non-0 sum output would be detected. Rows 2 through n have all '0' inputs in $T2_1$, and so the input to the RCA cell C_1 from cell C_2 must also be '0' and the other non-primary input to cell C_1 is '1'. Hence the carry output of $FA_{1,1}$ must be '1' and this change is propagated along row 1.

It remains only to show that the carry output of $FA_{1,1}$ remains at 0 in $T1_0$. The full adders in row 1 have the same column inputs in $T1_0$ and $T2_1$. Hence a change on the carry output of $FA_{1,1}$ would be propagated to the p input of C_1. In $T1_0$, the p and x inputs to C_1 are supposed to be '0'. Since we know C_1 is correct and can observe its carry output, its p and y inputs must be '0'. Hence the carry output of $FA_{1,1}$ is '0'.

Inductive Step: $k > 1$. Assume that all cells on diagonals numbered less than k behave as required. Consider a cell $FA_{i,j}$ on the k^{th} diagonal.

The tests $T1_j$ and $T1_{j-1}$ hold the row inputs at 0, the inputs of columns m through $j+1$ at 1, the inputs of columns $j-1$ through 1 at 0, while toggling the input of column j. In both of these tests, all cells in columns $j-1$ through 1 have their 0 inputs. So the row inputs to the cells in column j are '0'. By induction the cells in column j above row i toggle their sum outputs to '1' in $T1_{j-1}$ and so we know the sum input of $FA_{i,j}$ is '1' in $T1_{j-1}$. If $FA_{i,j}$'s sum output does not toggle to '1' as well then the cells below it in column j will have the same inputs in $T1_j$ as in $T1_{j-1}$, namely the '0' value. This cannot be the case because the S_j output must toggle in these two tests and nothing to the right of column j changes. So it must be the case that the sum output of $FA_{i,j}$ toggles to '1' in $T1_{j-1}$.

Now consider the tests $T1_{j-1}$ and $T2_i$. The row i output C_i must change its value in these two tests. The sum output of $FA_{i,j}$ could have caused the row i output to have changed by way of the input to C_i from C_{i+1}. However, as in the base case, we can show that all of the sum outputs of the full adders in row i are at their '0' values in $T2_i$. Suppose not. Let $FA_{i,h}$ be the first cell in row i whose sum output is not '0' in $T2_i$. Since the cells in rows $i+1$ through n and columns 1 through $h-1$ have their '0' inputs, we know the sum output of $FA_{1,h}$ is propagated down column j, and appears at the sum output for column j. Hence the sum outputs of the full adders in row i are at their '0' values in $T2_i$ since the first non-zero output would have been observed. This means the inputs to the full adder C_i from C_{i+1} must be '0' in $T2_i$ and only the row input to C_i could cause its sum output to change from '1' to '0' in $T1_{j-1}$ and $T2_i$. Hence the carry output of $FA_{i,j}$ must be '1' in $T2_i$ and this change is propagated along row i.

We now show that the carry output of $FA_{i,j}$ did not change in $T1_j$ and $T1_{j-1}$. By induction the cells in column j above row i do not change their carry outputs in tests $T1_j$, $T1_{j-1}$ and $T2_i$; these remain at 0. The cells of the FA array (excluding the RCA cells) in the region R_1 formed by the intersection of rows 1 through $i-1$ and columns m through $j+1$ have the same inputs in these three tests. This means that the column inputs to the cells in row i to the left of column j are the same in all three tests. Since the change to '1' of the carry output of $FA_{i,j}$ in $T2_i$ was propagated to the row input of C_i, it would also be propagated there in $T1_{j-1}$. But the cells C_1 through C_{i-1} are correct and their inputs are the outputs of the cells in R_1 and plus the carry output of C_i. Thus a change on the row i input to C_i would be observed either directly on its own sum output or on one of the outputs of the cells C_1 through C_{i-1}. We can conclude that the inputs of C_i must remain 1,0,0 in $T1_j$ and $T1_{j-1}$, and hence the carry output of $FA_{i,j}$ must remain at '0' as well.

□

As in Chapter 1.1.2, we use Dias' methods to test each row independently. Observe that these tests are consistent with a multiplicand of all 0's except for a 1 corresponding to the row being tested. The multiplier is the pattern to be applied to the p inputs of the cells in the row.

$T1$ and $T2$ tests	$m+n+1$	
$T3$ tests	$7n$	($T3_j^{000}$ is the same for all j and is also $T1_m$)
T^k tests	$11n+3$	
Total	$m+19n+4$	

For an $r \times q$ multiplier, we have $n = r$ and $m = r + q$, resulting in $q + 20r + 4$ tests.

Cheng and Patel presented a minimum row test for the RCA using the MFC fault model[15]. If this row test is used instead of the one presented, there are $11n$ T^k tests of which all but $8n+2$ are shared with other tests. This results in $q + 17r + 3$ tests to detect all multiple faults in an $r \times q$ multiplier.

2.1 An $n/\log(n+1)$ lower bound on tests for the CS multiplier

This bound is achieved by showing that for any set of fewer tests, an array can be constructed which passes these tests but will produce an erroneous output for some vector not in the test set. Note that we assume here that we can only observe the final output of the multiplier, not the outputs of the full adder array.

The construction involves modifying some of the full adders so that they either add or subtract one for some inputs. Table 2.2 contains the truth tables of the correct full adder as well as the +1-full adder which adds one to the correct result whenever its x and p inputs differ, and the -1-full adder which subtracts one from the correct result whenever its x and p inputs differ.

Inputs			full adder		+1-full adder		−1-full adder	
y	x	p	c	s	c	s	c	s
0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0
0	1	0	0	1	1	0	0	0
0	1	1	1	0	1	0	1	0
1	0	0	0	1	0	1	0	1
1	0	1	1	0	1	1	0	1
1	1	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	1

Table 2.2: Truth tables of correct and faulty full adders.

Now suppose $T = t_1, t_2, \ldots, t_q$ is a sequence of test vectors and that there is a column i in the array with two distinct disjoint subsets of cells S_+ and S_- with the following property:

For each test t_k the number of cells in S_+ whose inputs have $x \neq p$ is the same as the number of cells in S_- whose inputs have $x \neq p$.

In this case, we can construct a faulty array which will behave correctly for the tests in T by replacing each cell in S_+ by +1-full adders and each cell in S_- by −1-full adders. The cells in S_+ erroneously add one whenever an input with $x \neq p$ is applied to them, while those in S_- subtract one in this case. Since each test in T always applies the same number of $x \neq p$-inputs to cells in S_+ and cells in S_-, the number of ones added will always exactly offset the number of ones subtracted in column i for tests in T. Hence this multiplier will always produce the correct result for the tests in T, but will clearly produce an erroneous result for a vector which does not apply the same number of $x \neq p$-inputs to S_+ and S_-.

We have established that any test set which will detect faulty arrays under our fault model cannot have such a column with its two subsets of cells, S_+ and S_-. We now show that if the number of tests is less than $n/\log(n+1)$ where n is the number of rows in the array, then there is such a column with the required subsets of cells.

Fix a column i. Consider an arbitrary subset S of cells in column i. Each test applies a number of $x \neq p$-inputs to the cells in S ranging from 0 up to $|S| \leq n$. Associate with S a vector, $v(S)$, of length q (the number of tests) whose k^{th} component is the integer between 0 and n corresponding to the number $x \neq p$-inputs applied to the cells of S in t_k. There are at most $(n+1)^q$ different vectors which could be associated with a specific S and the number of distinct S's is 2^n. Suppose $q < n/\log(n+1)$. Then there are more subsets than vectors and we can find two distinct subsets of cells, A and B, not necessarily disjoint, such that $v(A) = v(B)$. This means that in each t_k, the cells in A and B receive the same number of $x \neq p$-inputs. Construct S_+ and S_- as follows:

$$S_+ = A - B \qquad\qquad S_- = B - A.$$

Clearly, S_+ and S_- are disjoint and distinct. It is easy to to show that $v(S_+) = v(S_-)$ since

$$v(A - B) = v(A) - v(A \cap B) = v(B) - v(A \cap B) = v(B - A).$$

Hence if the number of tests is less than $n/\log(n+1)$, we can construct a faulty array which will pass the tests. It follows that any test set which detects all faulty arrays under our fault model from the outputs of the multiplier must have at least $n/\log(n+1)$ tests. This bound applies to an array multiplier as well as to its full adder array, since not being able to independently control the product terms can only make the test set larger.

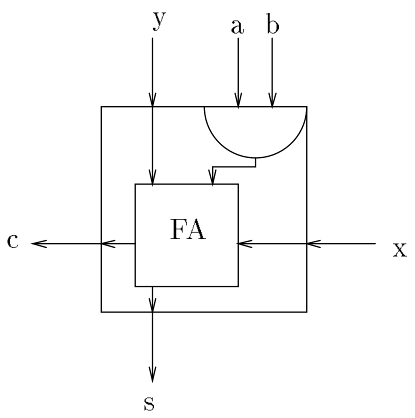

Figure 2.4: The combined cell for C-MFCM for the CS-multiplier.

2.2 Detection of All Multiple Faults in Carry-Save Multiplier

In this section, we present a modification to the multiplier so that all detectable multiple faults in the carry-save multiplier are detected by a test set whose length is proportional to the perimeter of the array. In the previous sections we constructed test sets that detect all multiple faults in the two-dimensional full adder array and the RCA. We can easily detect all multiple faults in the array of AND gates of the summand generator, but it is not guaranteed to detect multiple faults that affect both arrays.

It is common that the product generator and the summand counter be combined in the physical design of the circuit to simplify intercell interconnections. The logic for the AND gate may be incorporated into the implementation of the full adder cell. This makes it more likely that a single defect or a cluster of defects affect both the product generator and the summand counter. Clearly it is preferable to consider the basic cell of Figure 2.4 for test pattern generation.

The combined basic cell serves as a one-bit product generator and as a summand counter cell. It consists of inputs $\langle y, a, b, x \rangle$ and outputs $\langle c, s \rangle$. The logic function of the cell is as if it were a full adder and an AND gate. The inputs to the AND gate are a and b and the inputs to the full adder are y, x, and the output of the AND gate as shown in Figure 2.4. The a and b inputs to cell i, j in the carry-save multiplier are the ith bit of the multiplicand and the jth bit of the multiplier, respectively, as shown in Figure 2.5.

Our fault model is all multiple faults in the CS multiplier involving combined basic cells. We call this the combined-MFC or C-MFC model for the CS multiplier. This is a stronger fault model than treating the AND gates and the full adders separately under the MFC model. This is because there are $(2^{2^4})^2 - 1 \approx 4.3 \times 10^9$ faulty truth tables for the combined cell and the product of the faulty truth tables for the full adder and the AND gate considered separately is $(2^{2^3} - 1)^2(2^{2^2} - 1) \approx 9.7 \times 10^5$. The faults of the MFC model are a subset of the faults of the C-MFC model.

The testing approach used in the previous sections cannot be applied using the C-MFCM model because a loop test for input $\langle y, a, b, x \rangle = 0001$ or 0011 cannot be generated due to the impossibility of applying either of these vectors to more than one cell on the ith row. The proof of this is similar to that of Theorem 1 in [9]. To make it possible to generate a loop test for each row we modify the c output truth table of each basic cell in the way presented in [9]. The change in the carry truth table is shown in Table 2.3.

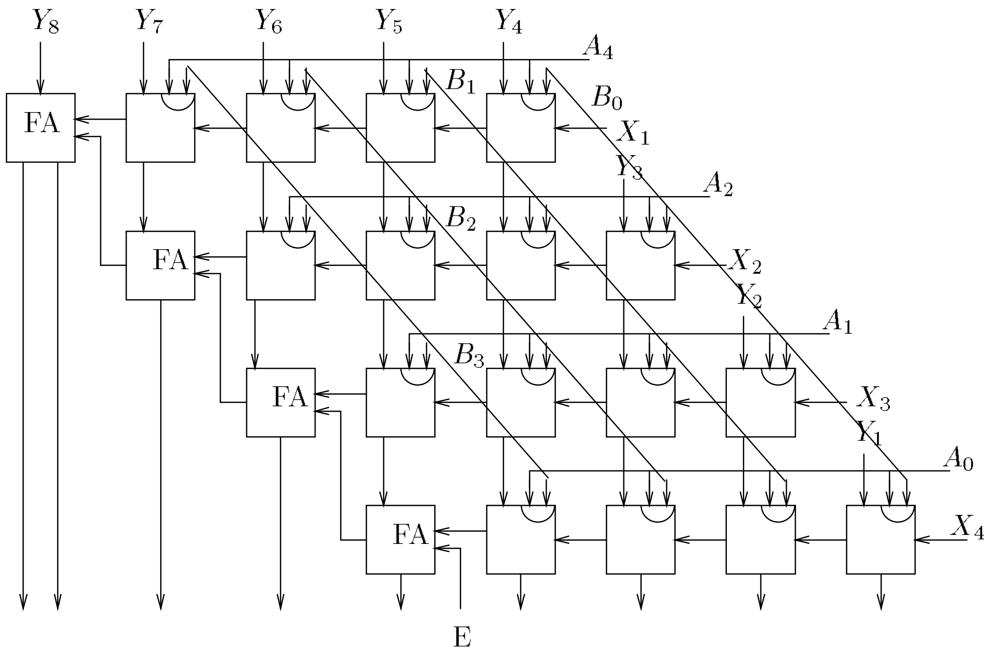

Figure 2.5: A 4×4 carry-save (CS) multiplier.

y, a, b, x	standard cell	modified cell
0000	0	0
0001	0	1
0010	0	0
0011	0	1
0100	0	0
0101	0	0
0110	0	0
0111	1	1
1000	0	0
1001	1	1
1010	0	0
1011	1	1
1100	0	0
1101	1	1
1110	1	1
1111	1	1

Table 2.3: The carry-output truth tables for the standard and modified cells.

The modified carry save (MCS) multiplier has exactly the same structure as the carry save multiplier, only the function of the basic cell is changed as reflected in Table 2.3. The function for the sum output is unchanged.

The X and Y primary inputs to the CS and the MCS multipliers are 0 in normal operation, that is, when they are multiplying two numbers. These inputs may have non-0 values only during testing. When all the X and Y primary inputs to the multiplier are assigned 0, the cell inputs $\langle y,a,b,x\rangle$ equal to 0001 or 0101 never occur in the correctly functioning multiplier [9]. Hence the MCS multiplier functions exactly the same as the carry save multiplier in normal operation and does not affect the multiplication function of the array.

We next describe tests that detect all multiple faulty cells for the MCS multiplier by describing how the test set for the carry-save multiplier differs from the test set for the array of full adders that form the summand counter in the standard array multiplier.

For any input to the MCS multiplier, all cells in any row will have either a 0 or a 1 on their a input. Hence the cells in any single row can be considered one of two state machines determined by the A input for that row. A set of Dias-tests can detect all faults in the row when $A=0$ and another set of Dias-tests can detect all faults in the row when $A=1$. We use the SIS $\langle yab\rangle = \langle 100\rangle$ for the state machine in which $A=0$ and the SIS $\langle yab\rangle = \langle 110\rangle$ for the state machine in which $A=1$.

test	X_k	$Y_{pattern}$	A_k	$B_{pattern}$	test	X_k	$Y_{pattern}$	A_k	$B_{pattern}$
T_0^{k0}	0	0,1	0	0,0	T_0^{k01}	0	1,0	0	0,0
T_1^{k0}	1	0,1	0	0,0	T_1^{k01}	1	1,0	0	0,0
T_2^{k0}	0	0,1	0	1,0	T_2^{k01}	0	1,0	0	0,1
T_3^{k0}	1	0,1	0	1,0	T_3^{k01}	1	1,0	0	0,1
T_4^{k0}	0	1,1	0	0,0					
T_5^{k0}	1	1,1	0	0,0					
T_6^{k0}	0	1,1	0	1,0	T_6^{k01}	0	1,1	0	0,1
T_7^{k0}	1	1,1	0	1,0	T_7^{k01}	1	1,1	0	0,1

Table 2.4: Row tests for A=0 for MCS multiplier.

test	X_k	Y_{pat}	A_k	B_{pat}	test	X_k	Y_{pat}	A_k	B_{pat}	test	X_k	Y_{pat}	A_k	B_{pat}
T_0^{k1}	0	0,1	1	0,0	T_0^{k11}	0	1,0	1	0,0					
T_1^{k1}	1	0,1,1	1	0,0,1	T_1^{k11}	0	1,1,0	1	0,1,0	T_1^{k12}	0	1,0,1	1	1,0,0
T_2^{k1}	0	0,1	1	1,0	T_2^{k11}	0	1,0	1	0,1					
T_3^{k1}	1	0,1	1	1,0	T_3^{k11}	1	1,0	1	0,1					
T_4^{k1}	0	1,1	1	0,0										
T_5^{k1}	1	1,1	1	0,0										
T_6^{k1}	0	1,1,0	1	1,0,0	T_6^{k11}	1	1,0,1	1	0,0,1	T_6^{k12}	1	0,1,1	1	0,1,0
T_7^{k1}	1	1,1	1	1,0	T_7^{k11}	1	1,1	1	0,1					

Table 2.5: Row tests for A=1 for MCS multiplier.

Tables 2.4 and 2.5 show the T^k tests for the kth row including the SIS and the sequence $\langle yab\rangle = \langle 010\rangle$ to bring the state back to 0 for T_6^{k1} and the sequence $\langle yab\rangle = \langle 111\rangle$ to bring the state back to 1 for T_1^{k1}. Blank entries in the table represent tests in which the loop test length is less than three. The test T_y^{kxs} is the shift of test T_y^{kx} by s and x is either 0 or 1 depending on the value of the a input to the cell.

Recall that in testing the full adder array it was necessary to show that the array was column-separable when the product terms and the internal Y inputs were 0. Since the B (multiplier) inputs to the array are the b inputs to the cells in the MCS and since they are non-0 for the tests in Tables 2.4 and 2.5, we must show that the array is column separable for non-0 values of B.

The A values of the array may remain 0 for all rows except the row being tested with the T^{k1} tests. From Table 2.5 we see that "column-separability" of the MCS multiplier array must be verified for $\vec{B} = 01010...$ and its shift, and 100100100... and its two shifts in addition to the case for $\vec{B} = 000...$ as is required for the T^{k0} tests. So the $T1_j$ and the $T2_i$ must be presented to the array six times, once for each of the six multiplicand values. The tests for the RCA remain the same so the number of tests for a $r \times q$ MCS multiplier is $6(2r + q + 1) + 30q + 7q = 12r + 38q + 1$.

3. Conclusion

We have presented a method for testing two-dimensional arrays for multiple faulty cells (under the MFC model) which applies to any iterative logic array composed of a cell which is column-separable. If in addition, a single row composed of this cell can be tested with a constant number of vectors, then the entire $n \times m$ array can be tested for all multiple faulty cells with $m(|Y|-1)+n(|X|-1)+1+nR$ vectors, where X and Y are the set of all possible values that can occur on the horizontal and vertical wires, and R is the constant number of vectors needed to test a single row.

If the basic cell in the array is not column-separable, it can be modified to be column-separable with, at most, one connection to each of two neighboring cells. If there are "don't care" inputs in the cells definition, these may be used to obtain the required a and b values without additional intercell wires. By Dias' Theorem, if a cell has a flow table which is reduced and has only strongly connected components, then a one-dimensional array composed of the cell has a constant length test set under the MFC model. The modification to make a cell column-separable can at the same time ensure that the cell meets Dias' requirements, by making the permutation corresponding to $v(x,b)$ cycle through all of the states. This ensures that its flow table is strongly connected. If the flow table is subsequently reduced, then the resulting cell will still be column-separable and satisfy Dias' requirements. The resulting array can be fully tested with $m(|Y|-1)+n(|X|-1)+1+nR$ tests.

We show that full adders are column-separable and use the techniques developed for two-dimensional ILAs to test full adder arrays. Two-dimensional full adder arrays form the core of combinational multiplier and divider circuits. We show that the full adder array embedded in the carry-save multiplier can be tested using our technique without complete access to the inputs and outputs of the full adder array. Often the product generator, which is an AND gate is embedded in the full adder cell in the implementation of a carry-save multiplier. The methods described in this paper cannot be used to provide a test for this array. However, a modification to the truth table of the cell allows us to generate a test set whose length is proportional to the sides of the array.

Finally lower bounds are presented for testing under the MFC fault model for ILAs composed of full adder cells.

References

[1] W.H. Kautz. Testing for faults in combinational cellular logic arrays. In *Proc. 8th Annu. Symp. Switching and Automata Theory*, pages 161–173. ACM, 1967.

[2] A.D. Friedman. Easily testable iterative systems. *IEEE Transactions on Computers*, C-22(12):1061–1064, December 1973.

[3] F.G. Gray and R.A. Thompson. Fault detection in bilateral arrays of combinational cells. *IEEE Transactions on Computers*, C-27(12):1206–1213, December 1978.

[4] R. Parthasarathy and S.M. Reddy. A testable design of iterative logic arrays. *IEEE Transactions on Circuits and Systems*, CAS-28(11):1037–1045, November 1981.

[5] W.-T. Cheng. Testing and error detection in iterative logic arrays. Technical Report UILU-ENG-85-2234, University of Illimois at Urbana-Champaign, August 1985.

[6] W.-T. Cheng and J. H. Patel. Testing in two-dimensional iterative logic arrays. In *Proceedings of Fault Tolerant Computing Symposium*, pages 76–81. IEEE, 1986.

[7] A. Chatterjee and J. Abraham. Test generation for arithmetic units by graph labelling. In *Proceedings of Fault Tolerant Computing Symposium*, pages 284–289. IEEE, 1987.

[8] C.-W. Wu and P.R. Cappello. Easily testable iterative logic arrays. *IEEE Transactions on Computers*, C-39(5):640–652, May 1990.

[9] J.P. Shen and F.J. Ferguson. The design of easily testable VLSI array multipliers. *IEEE Transactions on Computers*, C-33(6):554–560, June 1984.

[10] Sung Je Hong. The design of a testable parallel multiplier. *IEEE Transactions on Computers*, 39(3):411–416, March 1990.

[11] C.H. Stapper, F.M. Armstrong, and K. Saji. Integrated circuit yield statistics. *Proceedings of the IEEE*, 71(4):453–470, April 1983.

[12] F.J.O. Dias. Truth-table verification of an iterative logic array. *IEEE Transactions on Computers*, C-25(6):605–613, June 1976.

[13] E.M. Aboulhamid. Efficient testing and truth table verification of unilateral combinational iterative arrays. *Proceedings of International Conference on Computer-Aided Design*, pages 68–70, 1985.

[14] B.A. Prasad and F.G. Gray. Multiple fault detection in arrays of combinational cells. *IEEE Transactions on Computers*, C-24(8):794–802, August 1975.

[15] W.-T.Cheng and J.H. Patel. A minimum test set for multiple fault detection in ripple carry adders. *IEEE Transactions on Computers*, C-36(7):891–895, July 1987.

[16] W-T. Cheng and J.H. Patel. Testing in two-dimensional iterative logic arrays. *Comput. Math. Applic.*, 13(5/6):443–454, 1987.

Error in Proof

Wu-Tung Cheng claims in his Ph.D.thesis[5] and in an article in *Comput. Math. Applications*[16] that if a two-dimensional iterative logic array has

1. only vertical and horizontal connections between cells,
2. a special horizontal cell input, **a**, such that h(a, z) = (a) and v(a, z) = (C(z)) for each vertical input z, and C(z), the vertical output, is a one-to-one function, and
3. each row of the array is C-testable,

then the array can be tested under the MFC model in a test size that is proportional to the number of rows in the array.

The journal article refers to the Ph.D. thesis for a detailed proof, so we shall refer to the proof in the Ph.D. thesis. This proof is based on induction by row and uses Figure 2.26 of the Ph.D. thesis which is reproduced here as Figure 3.1. In the following paragraph, the italicized words are from the thesis' proof, paragraph one, page 46.

> ...*assume that all the cells in rows 1, ..., i-1* ...[in Figure 3.1 function correctly] ...*and then consider cell (i,j) and the area within the solid line for a specific distinguishing sequence.* [The two cells (i,j+1) and (i,j+2) are given vertical input values (a distinguishing sequence) to distinguish the horizontal output of cell (i,j).] *When this cell is being tested, the inputs of this area* [the area partially enclosed by the solid line] *are cell test input (x,y), fixed special input* **a**, *and special SDSs.*

The last sentence in this paragraph contains the flaw in the argument. The inputs applied to cell (i,j) are (x,y), the special input **a** is applied to all horizontal inputs of the array except for the row being tested (row i), and the SDSs are the distinguishing sequences applied to the vertical inputs of cell (i,j+1) and cell (i,j+2). Since rows 1, ..., i-1 are assumed by induction to be functioning properly and have the special input **a** applied to the horizontal input of each cell, the SDSs are being applied to cells (i,j+1) and (i,j+2). However, the cells to the left of the leftmost solid line, cells (k,l) where k > i-1 and l < j are **not** covered by the inductive assumption and may have faults. Hence the horizontal inputs to bounded region may **not** consist of all **a**'s. If the cells enclosed by the solid line are given an erroneous value instead of the special value **a**, then the errors may collide and cancel each other. Hence the next line of the proof is unsupported:

> *With Property (2),*[1] *the* $\hat{z}$ (cell's vertical) *outputs have a one-to-one relation with the vertical outputs of column j. ... With the special input* **a**, *the one-to-one relation propagate to the vertical boundary outputs within this area.*

This last sentence shows the necessity of having the **a** input applied to columns j, j+1, and j+2 in the rows below row i.

It is possible to construct an example in which the inductive argument fails in the first row:

Example:
Figure 3.2 shows a two dimensional array with a multiple cell fault for which the inductive step does not hold. The value of the vertical output of each good cell, $\hat{v}$, is the exclusive-or

[1] Property 2 states: "For every cell, the output of every transistion is verified such that the primary output is correct and that the output state in each direction has the one-to-one relation with the correct output state of A_G in the same direction."

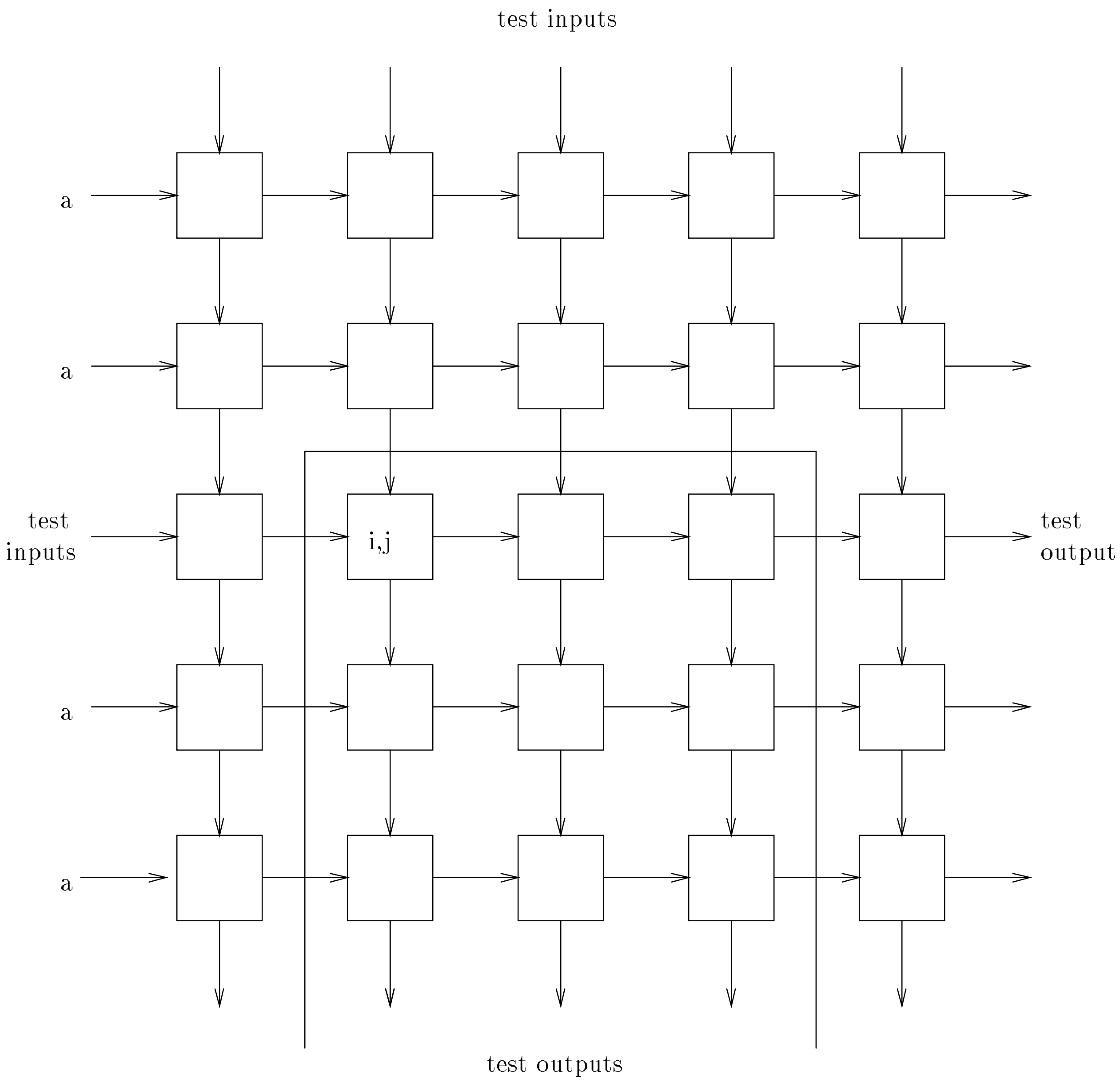

Figure 3.1: “Fig. 2.26. Testing of Case 4” in Cheng’s thesis

of the values of the vertical and horizontal inputs, v and h, respectively. The value of the horizontal output of a good cell, $\hat{h}$, is the same as the horizontal input, h. The truth table of each cell is printed within the cell where the logic values are, from left to right, $h, v, \hat{h}, \hat{v}$. If the cell is faulty, then the faulty entry is followed by an asterisk (*) in the cell’s truth table.

The MFC test set that we chose for each row is a C-test consisting of the tests shown in Table 3.1. The column labelled h is the single horizontal input to the row under test, and the column labelled v_{odd} (v_{even}) is the value applied to the vertical inputs to the array with an odd (even) subscript. These six tests apply all four input combinations to each cell in the row followed by a distinguishing sequence of length one consisting of “0”. We chose “0” to be the special horizontal value to apply to each row input of the array.

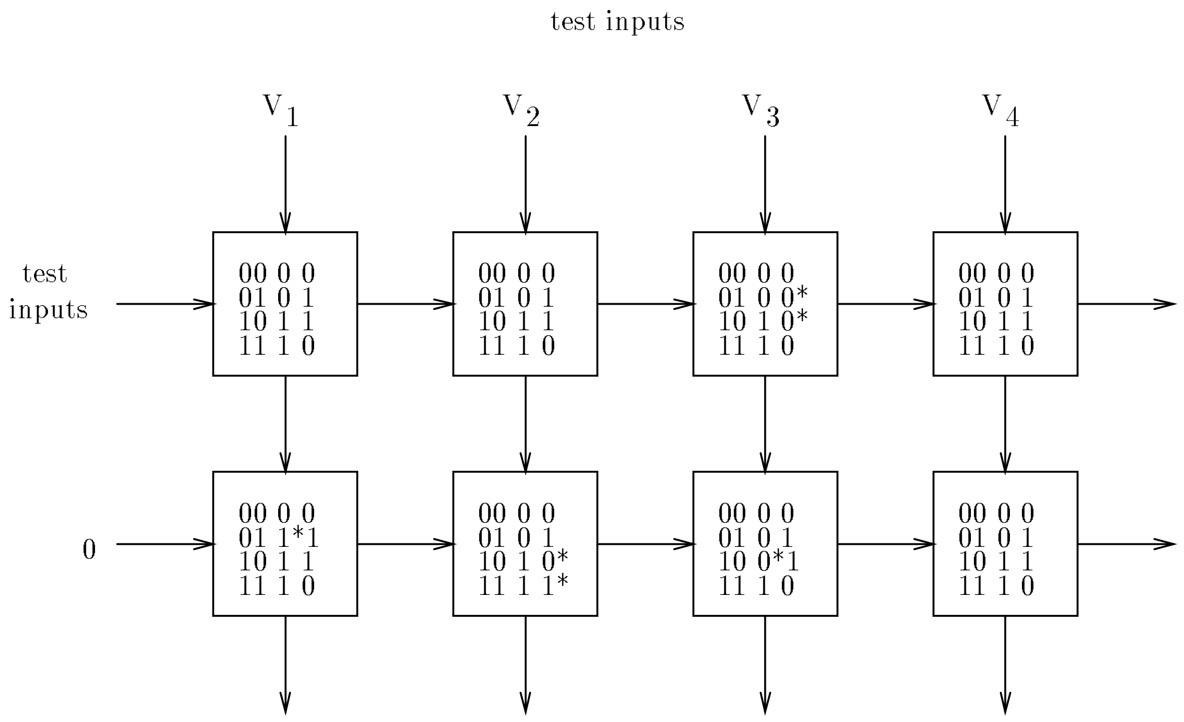

Figure 3.2: Faulty Array in which induction argument fails.

Test	h	v_{odd}	v_{even}
t_1	0	0	0
t_2	0	1	0
t_3	1	0	0
t_4	1	1	0
t_5	0	0	1
t_6	1	0	1

Table 3.1: C-test for array of figure 13.

The application of this test set to the first row does not detect the fault in the faulty cell$_{1,3}$ for the inputs 01 and 10 because a horizontal error is generated in cell$_{2,1}$ which masks it. Since the test $\{h_1, h_2, v_1, v_2, v_3, v_4\} = \{0, 0, 0, 0, 1, 0\}$ produces an error on the output of the array there is not the one-to-one correspondance that is necessary for the proof in the Ph.D. thesis. Thus the tests for row 1 are passed, **yet the induction hypothesis is false** — cell (1,3) clearly does not have vertical outputs which are in one-to-one correspondence with the vertical outputs of a correct cells nor the column 3 outputs. The test in the example failed to ensure that row 1 satisfies the induction hypothesis because of the flawed assumption in the proof: the special input **a** (0 in the example) is not applied to cell (2,2) in all tests for row 1.

We should note that in this case the tests for row 2 will indeed detect this faulty array. However, there is no assurance that this will be the case for all arrays.