design of TSC checkers for $m/2m$ codes," *IEEE Trans. Comput.*, vol. C-37, pp. 301-309, Mar. 1988.
[20] I. Janch and B. Courtois, "SCD checkers, cellular checkers and multichecker structures," IMAG Rep., RP 476, Nov. 1984.
[21] J. Smith, "The design of TSC check circuits for a class of unorder codes," *J. Design Automat. Fault-Tolerant Comput.*, vol. 1, pp. 321-342, Oct. 1977.
[22] N. Jha and J. Abraham, "Techniques for efficient MOS implementation of TSC checkers," in *Proc. FTCS-15*, June 1985, pp. 430–435.

The Design of a Testable Parallel Multiplier

SUNG JE HONG

Abstract—**A parallel multiplier can be usually broken into the summand-generator and the summand-counter. The summand-generator generates summands and the summand-counter adds them up. The summand-generator is easy to test because all inputs are directly controllable and all faults automatically propagate to primary outputs. However, the summand-counter is difficult to test due to poor controllability. This paper introduces a modified summand-generator, which provides 100 percent controllability of summands using one extra pin. This new summand-generator is testable with 19 vectors. With this summand-generator, summand-counters can be constructed to be testable with minimal use of adder cells and no extra hardware. This summand-counter is testable with $3n+41$ vectors. In summary, a parallel multiplier can be designed testable with $3n+60$ vectors using only one extra pin.**

Index Terms—**C-testable, design for testability, 100 percent controllability, parallel multiplier, summand-generator, summand-counter.**

I. Introduction

As the circuit complexity increases, testing and test generation becomes harder mainly due to the high device-to-pin ratio. As this ratio increases, controllability and observability usually decreases, thus increasing the complexity of test generation. To alleviate the test problem, a design approach called *design for testability* (DFT) [1] is often employed. Most DFT techniques require extra hardware, i.e., extra devices and/or some extra testing inputs (pins), and this hardware overhead often results in performance degradation. This paper describes the design of an easily testable parallel multiplier with minimal hardware overhead.

Generally, a parallel multiplier can be divided into two blocks, the summand-generator and the summand-counter, as shown in Fig. 1. The summand-generator takes the multiplicand (a_i's) and the multiplier (b_i's) as inputs and produces the summands ($p_{i,j}$'s) as outputs. The summand-counter takes the summands as inputs and produces the product (s_i's) as outputs. The summand-generator and the summand-counter are usually constructed with AND gates and adder cells (i.e. full-adders and half-adders), respectively. The summand-generator is easy to test because all inputs are directly controllable and all faults automatically propagate to primary outputs through the summand-counter. However, the summand-counter is difficult to test due to poor controllability, i.e., the difficulty in controlling the inputs (i.e., summands) from the primary inputs (namely the multiplicand and

Manuscript received June 15, 1987; revised January 15, 1988 and April 16, 1989.

The author is with the Department of Computer Science, Pohang Institute of Science and Technology, Pohang 680, Korea.

IEEE Log Number 8932030.

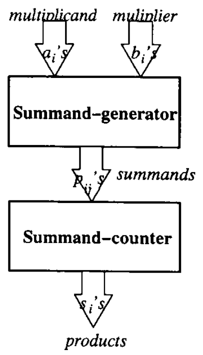


Fig. 1. A general structure of multiplier.

the multiplier) through the summand-generator. This is due to the fact that some input patterns cannot be applied to adder cells. For example, there is no set of values for the multiplicand and the multiplier, which generates the pattern 011 (0 for the x-input and 1 for the y-input and z-input) to the bottom leftmost cell $C(5, 4)$ in Fig. 3. This is due to the use of half-adders. In general, the use of half-adders reduces the controllability of blocks that they drive since half-adders cannot generate the output 11. One way to solve this problem is to use full-adders instead of half-adders. The multipliers proposed in [2] and [3] use this approach. The main drawback of this approach is the increase on the input count, i.e., at least five extra pins. The approach, proposed in this paper, modifies the summand-generator to provide 100 percent controllability of summands. *100 percent controllability of summands* implies that arbitrary values of summands can be provided by appropriate values of the primary inputs (the multiplicand and the multiplier) through the summand-generator. The modified summand-generator requires only one extra pin. With this modified summand-generator, the summand-counter can be constructed testable using the minimum number of adder cells but no extra devices or pins.

II. Testing Methodology

Throughout this paper, a_i's and b_i's, $1 \leq i \leq n$, denote the multiplicand and multiplier bits, respectively, s_i's, $1 \leq i \leq 2n$, the final product, and $p_{i,j}$'s, $1 \leq i,\ j \leq n$, the summand bits. This paper assumes that the summand-counter is constructed with adder cells, i.e., full-adders and half-adders, such that any functional fault will occur in a single adder cell. This paper also assumes that any fault in an adder cell does not destroy the combinational nature of the adder circuit. An adder cell can be tested by applying all possible input patterns (i.e., eight patterns for full-adders and four patterns for half-adders). If a cell is faulty, a faulty signal appears at its output(s) for some input pattern(s) and then propagates to a primary output as proved in Lemma 3 of [2]. Thus, any set of vectors, which provides all possible input patterns to each cell of a summand-counter, will be a test set for the summand-counter.

III. Design of Multipliers

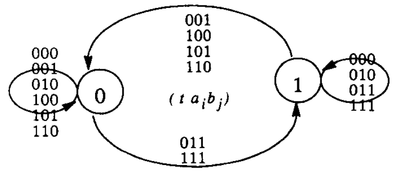
A. Design Methodology

The main difficulty in testing a multiplier lies in the poor controllability of the summand-counter. To enhance its controllability, the summand-generator is modified so that any arbitrary input value can be applied to the summand-counter. With the modified summand-generator, the summand-counter can be constructed testable without any modification. Since the summand-counter inputs are now 100 percent controllable, the test generation problem for the summand-counter can be divided into two independent subproblems:

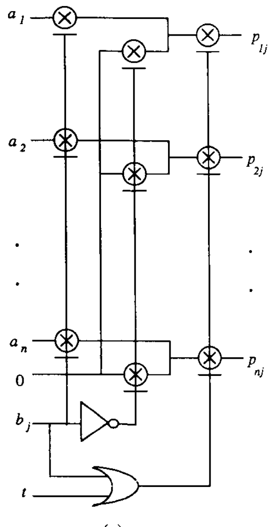
T1: Generate test vectors in terms of summands (considering summands as primary inputs).

current	inputs			next
p_{ij}	t	a_i	b_j	p_{ij}
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

(a)



(b)



(c)

Fig. 2. A modified summand p_{ij}. (a) Truth table. (b) State diagram. (c) An implementation of the jth summand row.

T2: Generate vectors of primary inputs, which provide the vectors of T1 to the summands.

The summand-counter is usually constructed as a two-dimensional adder array. There are two types of adder arrays, namely, carry–save adders and carry–propagate adders. From the testing point of view, the carry–save adder is better than the carry–propagate adder because the former has better controllability. The summand-counters discussed in this paper are constructed with carry–save arrays.

B. A Modified Summand-Generator

1) Functionality: The summand-generator generates the summands $p_{i,j}$'s, which are the logical product of multiplicand and multiplier bits, i.e., a_ib_j's. In order to provide 100 percent controllability of summand-counter, the function of each summand is modified by adding another variable t as follows. For each $1 \le i, j \le n$,

$$p_{i,j} = \begin{cases} a_ib_j & \text{if } t = 1, \\ a_i & \text{if } t = 0 \text{ and } b_j = 1, \\ \text{unchanged} & \text{if } t = 0 \text{ and } b_j = 0. \end{cases} \tag{1}$$

This modification provides two operating modes, namely *the normal mode* and *the assignment mode*. In the normal mode, t is set to 1 and the summand $p_{i,j}$ is assigned the logical product of the values of a_i and b_j. In the assignment mode, t is set to 0 and the value of $p_{i,j}$ depends on b_j. When b_j is 1, the value of a_i is assigned to $p_{i,j}$, and otherwise the value of $p_{i,j}$ does not change. Fig. 2(a) and (b) shows a truth table and a state diagram of the modified summand function. This modification introduces a sequential behavior but does not affect the normal operation of multiplication since t is set to 1 in the normal mode. Using the assignment mode an arbitrary set of values can be applied to a row of summands without affecting the values of other rows. In order to apply a vector $\langle x_n, x_{n-1}, \cdots, x_1\rangle$ to the summands of some row j, $\langle p_{n,j}, p_{n-1,j}, \cdots, p_{1,j}\rangle$, the primary inputs $\langle a_n, a_{n-1}, \cdots, a_1\rangle \langle b_n, b_{n-1}, \cdots, b_1\rangle \langle t\rangle$ can be set to $\langle x_n, x_{n-1}, \cdots, x_1\rangle$ $\langle$all 0's except the jth bit$\rangle$ $\langle 0\rangle$. Since $t = 0$ and $b_j = 1$, each $p_{i,j}$ is set to x_i for $1 \le i \le n$. Thus, the summand of the row j is assigned to $\langle x_n, x_{n-1}, \cdots, x_1\rangle$. Note that $p_{i,k}$'s for $k \ne j$ do not change their values since $b_k = 0$. In other words, the summands of other rows preserve the previously assigned values. Assigning all summands in an arbitrary way using the assignment mode requires n input vectors, one for each row of summands. However, two rows of summands with the same value can be applied simultaneously by selecting these two rows at the same time (i.e., setting the corresponding multiplier bits to 1).

2) An Implementation: The detailed implementation of the basic cells (i.e., the summand-generator and adder cells) is not the main focus of this paper, but an implementation of the modified summand-generator is shown in Fig. 2(c). Using MOS technology, the output of transmission-gate (namely, pass-transistor) is usually valid for at least one microsecond and one multiplication for a reasonably big multiplier takes less than 100 ns (e.g., 45 ns for 16×16 bits [6], and 70 ns for 24×24 bits [7]). Thus, the output of the pass-transistor needs to be reassigned at least every ten multiplications. As will be discussed later, test vectors for summand-counter assign each summand within nine multiplications.

C. Summand-Counter

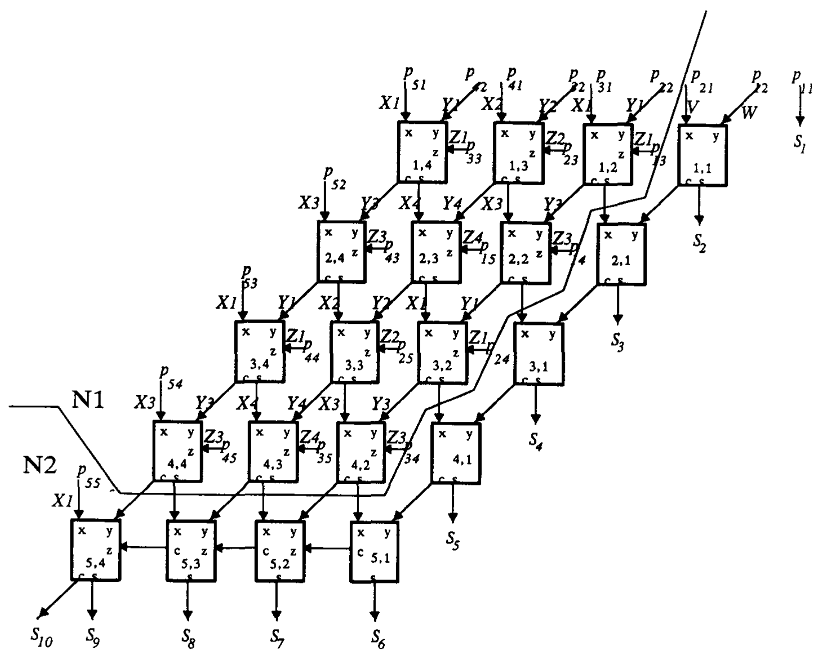
1) A Minimal Counter: A *counter CNT* $(i_1, i_2, \cdots, i_m\colon n)$ is a l-input n-output function (where $l = \sum_{j=1}^{m} i_j$), which adds i_k bits of weight 2^{k-1} $(1 \le k \le n)$ and generates the sum in n output bits. The summand-counter for an $n \times n$ multiplier can be expressed as CNT$(1, 2, \cdots, n-1, n, n-1, \cdots, 2, 1\colon 2n)$. Whenever a counter is constructed with full-adders and half-adders, the numbers of full-adders and half-adders can be obtained by the following theorems.

Theorem 1: The number of full-adders needed in constructing a counter is the difference between the numbers of inputs and outputs.

Theorem 2: The minimum number of half-adders needed to construct any counter is the number of 0's in outputs when 1's are applied to all inputs.

Theorem 3: The $n \times n$ summand-counter needs n^2–$2n$ full-adders and at least n half-adders.

2) A Summand-Counter: The following procedure constructs a summand-counter (known as a *carry-save array multiplier* [4]).



* *X1 - Z4, V and W denote the line vector.*

Fig. 3. The 5 × 5 summand-counter.

D1: Design procedure for the $n \times n$ summand-counter (see Fig. 3)

Step 1: (Placement of adder cells): Place an adder cell $C(i, j)$ at the jth column of the ith row (where $1 \le i \le n$ and $1 \le j \le n - 1$) such that $C(i, j)$ is a half-adder if $j = 1$ and a full-adder if $j \ne 1$.

Step 2: (Interconnection of adder cells): Connect the cells in the last row in a carry-propagate manner and other cells in a carry-save manner.

Step 3: (Connection of Summands): Connect each summand $p_{i,j}$ as follows:

1) to z to $C(i+1, j-2)$	if $1 \le i \le n - 2$ and $4 \le j \le n$,
2) to z of $C(j-1, n-1)$	if $i = n - 1$ and $3 \le j \le n$,
3) to x of $C(j, n-1)$	if $i = n$ and $2 \le j \le n$,
4) to x of $C(1, i-1)$	if $2 \le i \le n$ and $j = 1$,
5) to y of $C(1, i)$	if $1 \le i \le n - 1$ and $j = 2$,
6) to z of $C(1, i+1)$	if $1 \le i \le n - 2$ and $j = 3$.

Step 4: (Connection of products): The final product s_i's are the outputs of the cells in the first column and the last row.

D1 constructs the $n \times n$ summand-counter with $n^2 - 2n$ full-adders and n half-adders, which is minimal from Theorem 3. Fig. 3 shows the 5 × 5 summand-counter constructed using D1.

IV. Test Generation

A. The Modified Summand-Generator

The modified summand function $p_{i,j}$ provides sequential behavior. As shown in Fig. 2(b), its state diagram consists of two states and 16 state transitions. A test set must identify the two states and also check the 16 state transitions. The two states are denoted as 0 and 1, which are the values of $p_{i,j}$. Thus, the states are directly identified from the values of $p_{i,j}$. A state transition can be verified by checking two consecutive values of $p_{i,j}$. A test set verifying the 16 state transitions is shown in Table I. The first vector t_1 initializes to the state 0. Then the vectors t_2–t_7 verify six state transitions $0 \rightarrow 0$. The vectors t_8 and t_{14} verify two state transitions $0 \rightarrow 1$. The vectors t_9–t_{12} verify four state transitions $1 \rightarrow 1$. The vectors t_{13}, t_{15}, t_{17} and t_{19} verify four state transitions $1 \rightarrow 0$. The vectors t_{16} and t_{18} are mainly used to go to the state 1 to check the transitions by t_{17} and t_{19}. The 19 test vectors for the summand-generator can be constructed from each vector of Table I by setting the multiplicand bits, the multiplier bits, and the input t to the values of a_i, b_j, and t, respectively. For example, a test set for the 5 × 5 modified summand-generator is shown in Table II.

Since the circuit for the modified summand is asynchronous, a race can cause a problem. However, when either t or b_j is 1, $p_{i,j}$

TABLE I

A Test Set for the Modified Summand Function p_{ij}

test	t	a_i	b_j	state transition
t_1	1	0	0	$? \rightarrow 0$
t_2	1	0	0	$0 \rightarrow 0$
t_3	1	1	0	$0 \rightarrow 0$
t_4	1	0	1	$0 \rightarrow 0$
t_5	0	0	1	$0 \rightarrow 0$
t_6	0	0	0	$0 \rightarrow 0$
t_7	0	1	0	$0 \rightarrow 0$
t_8	0	1	1	$0 \rightarrow 1$
t_9	1	1	1	$1 \rightarrow 1$
t_{10}	0	1	1	$1 \rightarrow 1$
t_{11}	0	1	0	$1 \rightarrow 1$
t_{12}	0	0	0	$1 \rightarrow 1$
t_{13}	1	0	0	$1 \rightarrow 0$
t_{14}	1	1	1	$0 \rightarrow 1$
t_{15}	1	1	0	$1 \rightarrow 0$
t_{16}	1	1	1	$0 \rightarrow 1$
t_{17}	1	0	1	$1 \rightarrow 0$
t_{18}	1	1	1	$0 \rightarrow 1$
t_{19}	0	0	1	$1 \rightarrow 0$

TABLE II
A Test Set for the 5 × 5 Modified Summand-Generator

	multiplicand	multiplier	
test	$< a_5a_4a_3a_2a_1>$	$< b_5b_4b_3b_2b_1>$	t
t_1	< 00000>	< 00000>	1
t_2	< 00000>	< 00000>	1
t_3	< 11111>	< 00000>	1
t_4	< 00000>	< 11111>	1
t_5	< 00000>	< 11111>	0
t_6	< 00000>	< 00000>	0
t_7	< 11111>	< 00000>	0
t_8	< 11111>	< 11111>	0
t_9	< 11111>	< 11111>	1
t_{10}	< 11111>	< 11111>	0
t_{11}	< 11111>	< 00000>	0
t_{12}	< 00000>	< 00000>	0
t_{13}	< 00000>	< 00000>	1
t_{14}	< 11111>	< 11111>	1
t_{15}	< 11111>	< 00000>	1
t_{16}	< 11111>	< 11111>	1
t_{17}	< 00000>	< 11111>	1
t_{18}	< 11111>	< 11111>	1
t_{19}	< 00000>	< 11111>	0

is independent of the previously assigned value. In order to avoid a race, it is necessary to provide vectors such that *only one variable changes value when both t and b_j become 0.* Clearly, the vectors in Table I satisfy this restriction. This restriction does not affect the normal operation since t is always 1.

B. The Summand-Counter by D1

The test set for summand-counter must provide all possible eight or four input patterns to each adder cell. In the following discussion, a pattern abc for a full-adder, where a, b, and c are either 0 or 1, implies that a, b, and c are applied to the inputs x, y, and z of that full-adder. Similarly, a pattern ab for a half-adder implies that a and b are applied to the inputs x and y of that half-adder.

1) T1: Generation of Test Vectors in Terms of Summands: Test vectors must provide the eight input patterns to all full-adders and the four input patterns to all half-adders. Note that the test vectors for the modified summand-generator (see Table II) provide either all 0's or all 1's to the summands. All 0's to the summands sets all interconnection lines to the value 0. This implies the pattern 000 to all full-adders and the pattern 00 to all half-adders. Similarly, all 1's to the summands implies the pattern 111 to all full-adders and the pattern 11 to all half-adders. Thus, in testing the summand-counter, only the remaining six patterns (001, 010, 011, 100, 101, and 110) to full-adders and two patterns (01 and 10) to half-adders need to be considered. The adder cells in the summand-counter constructed by D1 can be partitioned into two groups:

N1: All full-adders except in the last row.
N2: All half-adders and the full-adders in the last row.

First let us consider the testing of N1. Note that N1 is a regular structured two-dimensional array of full-adders such that none of them are connected to half-adders. Let a *line vector* denote the collection of values of a line when all test vectors are applied. Then there is a one-to-one correspondence between a test set and line vectors. This implies that a test set can be obtained from the line vectors of input lines. Suppose that A, B, C, D, and E are the line vectors assigned to x, y, z, s, and c of a full-adder, respectively. Then the following conditions must be satisfied.

Condition 1: A, B, and C provide the six patterns in bitwise.
Condition 2: $D = A \oplus B \oplus C$.
Condition 3: $E = \bar{D}$.

Condition 1 is due to the nature of test vectors, while Conditions 2 and 3 are due to the functionality of the full-adder. Note that

TABLE III
Line Vectors

X_1	< 000111>
Y_1	< 011001>
Z_1	< 101010>
X_2	< 100110>
Y_2	< 111000>
Z_2	< 001011>
X_3	< 010101>
Y_3	< 001011>
Z_3	< 111000>
X_4	< 110100>
Y_4	< 101010>
Z_4	< 011001>
V	< 100111>
W	< 111101>

Condition 3 holds when the input is neither 000 nor 111. In fact, all lines of N1 can be assigned with 12 line vectors of 6 bits such that the above conditions are satisfied for every cell. Let us label the full-adder $C(i, j)$ with 1, 2, 3, and 4 as follows:

1) if i is odd and j is even,
2) if both i and j are odd,
3) if both i and j are even,
4) if i is even and j is odd. (2)

Then assign the x, y, and z inputs of $C(i, j)$ with the line vectors X_k, Y_k, and Z_k, respectively, where k is the label of $C(i, j)$. The 12 line vectors, X_1-X_4, Y_1-Y_4, Z_1-Z_4, shown in Table III, satisfy Conditions 1, 2, and 3 for every cell. The six test vectors, generated from these line vectors in bitwise, provide all six patterns to all cells in N1. Next let us consider the testing of N2. The three summands, $p_{2,1}$, $p_{1,2}$ and $p_{n,n}$, are not input to N1 and thus can be assigned arbitrarily in the test vectors for N1. However, to reduce the test size, we can assign them in such a way that a test set for N1 provides as many patterns to the cells in N2 as possible. One way is to assign $p_{n,n}$ with the same line vector as $p_{n,n-2}$, and $p_{2,1}$ and $p_{1,2}$ with the line vectors $V = \langle 100111\rangle$ and $W = \langle 111101\rangle$, respectively. Then the input patterns not provided to the adder cells in N2 by the test set for N1 are the pattern 01 to half-adders and the pattern 011 to full-adders. The procedure P1 below consisting of three steps generates test vectors. The first step generates vectors for the six input patterns for the adder cells in N1. The second and third steps generate vectors for the pattern 01 to the half-adders and the pattern 011 to the full-adders in N2. In the procedure, c_i denotes the ith test vector. Each c_i can be specified with n row vectors. The jth row vector of c_i is expressed as $(c_i)_j \equiv \langle p_{n,j}, p_{n-1,j}, \cdots, p_{1,j}\rangle$.

P1: Test Generation for the $n \times n$ Summand-Counter Constructed by D1

Step 1: (test generation for the full-adders of N1)
a. (assignment of $p_{1,1}$, $p_{2,1}$, and $p_{1,2}$): Assign $p_{1,1}$ and $p_{2,1}$ to $V = \langle 100111\rangle$ and $p_{1,2}$ to $W = \langle 111101\rangle$.
b. (assignment of other summands): Label each full-adder $C(i, j)$ with 1, 2, 3, and 4 as (2). Then assign each summand with X_k, Y_k, and Z_k depending on the type of input, where k is the label of the cell.
c. (test vector generation): Generate six vectors, c_k for $1 \le k \le 6$, from the kth bit of the corresponding line vector, i.e., each $p_{i,j}$ has the value of the kth bit of its line vector.

Step 2: (test generation for half-adders of N2): Generate $n-2$ vectors, c_{k+6} for $1 \le k \le n-2$, by setting $(c_{k+6})_i$ to $\langle 111 \cdots 1\rangle$ if $i \ne 4$ and to $\langle$all 1's except the kth bit$\rangle$ if $i = 4$.

Step 3: (test generation for full-adders of N2): Generate c_{n+5} by setting $(c_{n+5})_i$ to $\langle 111 \cdots 1\rangle$ if $i = 1$ or 2 and to $\langle 011 \cdots 1\rangle$ otherwise.

$\langle$End of procedure$\rangle$

TABLE IV
A Test Set for the 5 × 5 Summand-Counter

c_1	c_2	c_3	c_4
01011	00000	00000	11111
00101	11111	01111	10001
01101	01000	01111	10000
01111	10101	01111	10000
01000	01101	01111	10000

c_5	c_6	c_7	c_8
11111	10111	11111	11111
00000	11011	11111	11111
10111	10010	11111	11111
01010	10000	11101	11101
10010	10111	11111	11111

c_9	c_{10}
11111	11111
11111	11111
11111	01111
11011	01111
11111	01111

Note that Step 2 of P1 generates $n-2$ vectors, where n is the number of half-adders used in the summand-counter. This implies that each usage of a half-adder requires one extra test vector. A test set for the 5 × 5 summand-counter (Fig. 3) generated by P1 is shown in Table IV.

2) T2: Conversion of Test Vectors in Terms of Primary Inputs: The test vectors by P1 are specified in terms of summands. They need to be applied to the summands through the modified summand-generator. As discussed in Section III-B1, a vector c_i generated by P1 can be applied to summands in a row by row manner using the assignment mode (i.e., with $t = 0$). The same row pattern on different rows can be applied simultaneously. For every c_i generated by P1,

$$(c_i)_4 = (c_i)_6 = (c_i)_8 = \cdots \text{ and } (c_i)_5 = (c_i)_7 = (c_i)_9 = \cdots$$

Thus, each vector has five distinct row vectors. The procedure P2 below assumes that each c_i has five distinct row patterns. Then each c_i can be mapped into five vectors of primary inputs. However, due to the asynchronous behavior of summand function, test vectors need to satisfy the restriction stated in Section IV-A, i.e., when b_j changes from 1 to 0, a_i must not change. This can be done by adding one vector between the applications of two row vectors. This extra vector can be generated from the previous vector by setting only b_i's to 0. Then each vector by P1 can be mapped into nine vectors of primary inputs through the summand-generator using the assignment mode.

P2: Conversion of c_i-Vectors generated by P1
For each c_i, generate nine vectors as follows:

test	multiplicand $\langle a_n, a_{n-1}, \cdots, a_1\rangle$	multiplier $\langle b_n, b_{n-1}, \cdots, b_1\rangle$	t
v_1	$(c_i)_1$	$\langle 0 \cdots 0000000001\rangle$	0
v_2	$(c_i)_1$	$\langle 0 \cdots 0000000000\rangle$	0
v_3	$(c_i)_2$	$\langle 0 \cdots 0000000010\rangle$	0
v_4	$(c_i)_2$	$\langle 0 \cdots 0000000000\rangle$	0
v_5	$(c_i)_3$	$\langle 0 \cdots 0000000100\rangle$	0
v_6	$(c_i)_3$	$\langle 0 \cdots 0000000000\rangle$	0
v_7	$(c_i)_4$	$\langle \cdot \cdots 1010101000\rangle$	0
v_8	$(c_i)_4$	$\langle 0 \cdots 0000000000\rangle$	0
v_9	$(c_i)_5$	$\langle 0 \cdots 0101010000\rangle$	0

(End of procedure)

TABLE V
A Mapped Test Set of Table IV

		multiplicand	**multiplier**
test		$< a_5a_4a_3a_2a_1>$	$< b_5b_4b_3b_2b_1>$
v_1	c_1	< 01011>	< 00001>
v_2		< 01010>	< 00000>
v_3		< 00101>	< 00010>
v_4		< 00101>	< 00000>
v_5		< 01101>	< 00100>
v_6		< 01101>	< 00000>
v_7		< 01111>	< 01000>
v_8		< 01111>	< 00000>
v_9		< 01000>	< 10000>
v_{10}	c_2	< 00000>	< 00001>
v_{11}		< 00000>	< 00000>
v_{12}		< 11111>	< 00010>
v_{13}		< 11111>	< 00000>
v_{14}		< 01000>	< 00100>
v_{15}		< 01000>	< 00000>
v_{16}		< 10101>	< 01000>
v_{17}		< 10101>	< 00000>
v_{18}		< 01101>	< 10000>
v_{19}	c_3	< 00000>	< 00001>
v_{20}		< 00000>	< 00000>
v_{21}		< 01111>	< 11110>
v_{22}	c_4	< 11111>	< 00001>
v_{23}		< 11111>	< 00000>
v_{24}		< 10001>	< 00010>
v_{25}		< 10001>	< 00000>
v_{26}		< 10000>	< 11100>
v_{27}	c_5	< 11111>	< 00001>
v_{28}		< 11111>	< 00000>
v_{29}		< 00000>	< 00010>
v_{30}		< 00000>	< 00000>
v_{31}		< 10111>	< 00100>
v_{32}		< 10111>	< 00000>
v_{33}		< 01010>	< 01000>
v_{34}		< 01010>	< 00000>
v_{35}		< 10010>	< 10000>
v_{36}	c_6	< 10111>	< 00001>
v_{37}		< 10111>	< 00000>
v_{38}		< 11011>	< 00010>
v_{39}		< 11011>	< 00000>
v_{40}		< 10010>	< 00100>
v_{41}		< 10010>	< 00000>
v_{42}		< 10000>	< 01000>
v_{43}		< 10000>	< 00000>
v_{44}		< 10111>	< 10000>
v_{45}	c_7	< 11111>	< 11111>
v_{46}		< 11111>	< 00000>
v_{47}		< 11110>	< 01000>
v_{48}	c_8	< 11111>	11111>
v_{49}		< 11111>	< 00000>
v_{50}		< 11101>	< 01000>
v_{51}	c_9	< 11111>	11111>
v_{52}		< 11111>	< 00000>
v_{53}		< 11011>	< 01000>
v_{54}	c_{10}	< 11111>	< 00011>
v_{55}		< 11111>	< 00000>
v_{56}		< 01111>	< 11100>

The five vectors, v_1, v_3, v_5, v_7, and v_9, provide the necessary patterns to all rows of summands and the four vectors, v_2, v_4, v_6 and v_8, are transition vectors used to avoid the races as discussed earlier. Note that not all c_i's have five distinct row patterns. For example, c_3 has two distinct row patterns. In this case, three test vectors of primary inputs are enough to provide c_3 to all summands, i.e., two vectors to assign all rows of summands and one transition vector in between. The numbers of distinct row patterns in c_i's and the corresponding mapped test vectors of primary inputs are shown in Table VI. Table V shows the mapped test set of Table IV. The

TABLE VI
THE TEST VECTORS GENERATED BY P1

test vector	row patterns	mapped tests
c_1	5	9
c_2	5	9
c_3	2	3
c_4	3	5
c_5	5	9
c_6	5	9
c_7	2	3
c_8	2	3
.	.	.
c_{n+4}	2	3
c_{n+5}	2	3
$n+5$	total	$3n+41$

TABLE VII
COMPARISON OF TESTABLE MULTIPLIERS

	S-F	C-A	H
extra inputs	7	5	1
extra devices	n FAs	$n-1$ XORs	n ORs n^2 TGs
adder cells	n^2 FAs	n^2 FAs	n^2-2n FAs n HAs
test size	16	8	$3n+60$

* S-F: **Ferguson and Shen's multiplier [2]**
* C-A: **Chatterjee and Abraham's multiplier [3]**
* H: **The proposed multiplier**
* TG: **transmission gates**

total 75 test vectors for the 5×5 multiplier are shown in Tables II and V.

V. COMPARISONS

Table VII compares the proposed multiplier (H) with Shen-Ferguson's multiplier (S-F) [2] and Chatterjee-Abraham's multiplier (C-A) [3]. H requires only one extra input, while S-F and C-A require seven and five extra inputs, respectively. S-F needs an extra row of full-adders to produce the higher order product bits. C-A needs $n-1$ extra XOR gates to provide test vectors to the leftmost full-adders. H requires a little extra circuitry for the modified summand-generator (e.g., in the case of MOS implementation in Fig. 2(c), n^2 extra transmission gates and n OR gates). S-F and C-A are C-testable, requiring 16 and 8 test vectors, respectively. H is not C-testable but can be tested with $3n + 60$ test vectors.

VI. CONCLUSIONS

This paper presents a new scheme for an easily testable multiplier and the corresponding test generation procedures. To provide 100 percent controllability of the summand-counter, the summand-generator is modified. The modified summand-generator can be implemented with little hardware overhead. Since the summands are 100 percent controllable, the summand-counter can be constructed with the minimum number of adder cells. The multiplier is not C-testable, but can be tested with a small numbers of test vectors, i.e., $3n+60$ vectors. The proposed multiplier requires only one extra input, while other C-testable multipliers usually require at least four or five extra inputs and more adder cells along with extra circuitry. Using the modified summand-generator proposed in this paper, other types of multipliers such as Wallace trees [4] and Baugh-Wooley's multipliers [5] can be easily constructed to be testable with only one extra input. Test sets for these multipliers can be obtained using the same test generation approach presented in this paper.

REFERENCES

[1] T. Williams and K. Parker, "Design for testability—A survey," *IEEE Trans. Comput.*, vol. C-31, pp. 2-15, Jan. 1982.

[2] J. P. Shen and F. J. Ferguson, "The design of easily testable VLSI array multipliers," *IEEE Trans. Comput.*, vol. C-33, pp. 554-560, June 1984.

[3] A. Chatterjee and J. A. Abraham, "Test generation for arithmetic units by graph labelling," in *Proc. 17th Annu. Symp. Fault-Tolerant Comput.*, July 1987, pp. 284-289.

[4] K. Hwang, *Computer Arithmetic, Principles, Architecture, and Design.* New York: Wiley, 1979.

[5] C. R. Baugh and B. A. Wooley, "Two's complement parallel array multiplication algorithm," *IEEE Trans. Comput.*, vol. C-22, pp. 1045-1047, Dec. 1973.

[6] Y. Kaji, N. Sugiyama, Y. Kitamura, S. Ohya, and M. Kikuchi, "A 45ns 16×16 CMOS multiplier," in *ISSCC 84 Dig. Tech. Papers*, 1985, pp. 84-85.

[7] C. Y. Ho, R. T. Jerdonek, S. E. Noujaim, and D. L. Schumacher, "A high performance 1.5 micron CMOS 24×24 bit multiplier," in *Proc. IEEE CICC*, 1984, pp. 30-33.