The Weighted Syndrome Sums Approach to VLSI Testing

ZEEV BARZILAI, JACOB SAVIR, GEORGE MARKOWSKY, AND MERLIN G. SMITH

***Abstract*—With the advent of VLSI, testing has become one of the most costly, complicated, and time consuming problems. The method of syndrome-testing is applicable toward VLSI testing since it does not require test generation and fault simulation. It can also be considered as a vehicle for self-testing. In order to employ syndrome-testing in VLSI, we electronically partition the chip into macros in test mode. The macros are then syndrome tested in sequence.**

In this paper we show the means to syndrome-test macros. We examine the size of the syndrome driver counter and establish a method of determining its minimal length. The problem of minimizing the number of syndrome references needed for testing is also investigated. It is shown that it is always possible to use one weighted syndrome sum as reference for each and every macro. The question of weighted sum syndrome-testability is addressed and methods to achieve it are discussed. A self-test architecture based on these concepts is described.

***Index Terms*—Partitioning, self-testing, syndrome-testable design, syndrome-testing.**

I. Introduction

In recent years considerable attention has been given to LSI/VLSI testing and testable design. Digital circuit manufacturers are well aware today of the need to design testability fixtures early in the design stage, or otherwise they will have to pay a higher testing bill later in the process. Using traditional testing schemes for VLSI requires high test generation and fault simulation times. New techniques are required.

Syndrome-testing [2], [4]-[6] is a step to achieve this goal. The notion of syndrome-testing is based on counting the number of ones realized by a Boolean function and comparing it to the fault-free count. Since there may be circuits and faults for which these fault-free and faulty syndromes are the same, modifications to produce a testable design are generally required. The syndrome-testable design, thus ensures that all the faulty syndromes will differ from the fault-free one by adding a small amount of I/O and logic. In a VLSI environment the chip is electronically partitioned into macros, in test mode, in order to reduce the overall test time.

In this paper we assume that the chip has been designed according to the Level Sensitive Scan Design (LSSD) [1] rules. From a testing standpoint it means that, basically, the testing problem has been reduced to testing the combinational circuitry between Shift Register Latches (SRL). Thus this task of testing the combinational logic can be accomplished by means of syndrome-testing.

In this paper we investigate a cost effective scheme for VLSI testing. Our proposal is to use syndrome-testing to test each and every macro of the chip. In Section II we consider the problem of parallel syndrome-testing of all functions involved in a macro. In Section III we determine the minimum counter size necessary to drive a multi-input-multioutput macro, where all macro outputs are tested in parallel. In Section IV we investigate ways to further reduce the number of references needed for testing by using weighted syndrome sums. The weighted syndrome sums approach has a natural appeal to self-testing because of the enormous test data savings that it may offer. The question of untestability with regard to weighted sums is then addressed, and some ways to overcome it are presented. Section V presents the self-test architecture. The paper concludes with a brief summary.

Manuscript received November 21, 1980; revised June 4, 1981.

The authors are with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

II. Syndrome-Testing of Multiple Output Circuits

Since the notion of syndrome-testing requires the application of all possible input combinations to the Circuit Under Test (CUT), a VLSI is partitioned to limit the test time to some acceptable level. Thus, assume that the VLSI chip has been partitioned into R macros (syndrome partitions), where each macro is a multiinput-multioutput digital circuit. Usually, short test times will mean smaller macros and, therefore, more macros, while longer test times will usually result in larger and fewer macros. It is important to note that there are tradeoffs between macro sizes, test times, and partitioning penalties, i.e., extra hardware and I/O pins. Fig. 1(a) describes the partitioning of the chip into macros, and Fig. 1(b) illustrates schematically a typical macro.

When we go to syndrome-test the chip we must make sure that all macros are tested properly. Those macros which have disjoint sets of inputs can always be tested in parallel. In the other cases where the macros are interconnected a sequential syndrome test procedure may be used.

The question arises as to how one should syndrome-test a given macro. If we separately syndrome-test each output function involved in the macro, this will require the repetition of the syndrome-test procedure as many times as there are output functions in the macro. This can be a very time-consuming process because of the multiplicity of the syndrome-test runs. Another way to achieve this same purpose is to try to syndrome-test all the output functions in parallel, namely, to use only one syndrome-test procedure exercising the totality of the macro inputs and obtaining all output syndromes in one pass. The following theorem shows that by using the method of parallel syndrome-testing, the true syndromes are recorded at the outputs of the macro.

Theorem 1: Let $y_i = f(x_{i1}, x_{i2}, \cdots, x_{ik_i})$, $i = 1, 2, \cdots, m$ be the output functions realized by the macro. Let p be the length of the counter which drives the input combinations to the macro, $p \geq k_i \ \forall\ i$. Then by running a syndrome-test procedure which exercises all 2^p possible combinations of the counter, the true syndrome appears at each output y_i.

Proof: Let M_i be the number of minterms in the function y_i, $i = 1, 2, \cdots, m$. Then, by definition the syndrome realized by output y_i is given by

$$S_i = \frac{M_i}{2^{k_i}}.$$

Let the inputs $x_{i1}, x_{i2}, \cdots, x_{ik_i}$ be connected to the bits $b_{i1}, b_{i2}, \cdots, b_{ik_i}$ of the counter. When the counter steps through all its possible input combinations each vector $\vec{x} = (x_{i1}, x_{i2}, \cdots, x_{ik_i})$ is applied 2^{p-k_i} times. Thus, the normalized ones' count appearing at the output y_i is given by

$$\frac{2^{p-k_i}M_i}{2^p} = \frac{M_i}{2^{k_i}} = S_i, \qquad i = 1, 2, ., m$$

which is the correct syndrome. Q.E.D.

III. The Syndrome Driver Counter Size

One of the key questions regarding parallel syndrome-testing of multiinput-multioutput macros is the minimum size of the input counter which drives the exhaustive test vectors, called the Syndrome Driver Counter (SDC). The length of the SDC falls within the range described by the following theorem.

Theorem 2: The length of the SDC L is bounded by

$$\max_i \{k_i\} \leq L \leq n, \qquad i = 1, 2, \cdots, m$$

where k_i is the number of inputs feeding the output function y_i, and n is the total number of inputs to the macro.

Proof: Since there are n input lines to the macro, the length of

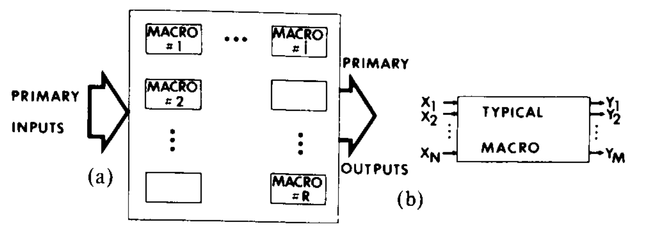

Fig. 1. (a) A VLSI chip partitioned into macros. (b) A typical macro with n inputs and m outputs.

the SDC is bounded from above by n. Also, since the SDC must be capable of driving each and every output function, it is clear that its length is bounded from below by $\max\{k_i\}$. Q.E.D.

Note that both the lower bound and the upper bound are attainable. The lower bound, for example, is attainable in the case where all the input sets driving the output functions are disjoint, while the upper bound is attainable when one of the output functions is dependent upon all input variables.

It is of utmost importance to be able to determine the minimal size of the SDC because it has a direct effect on the test time required to syndrome-test the macro. Let the minimal size SDC be q. Let, also, the bits of the SDC be denoted by $b_i, b_2, \cdots, b_q$.

The problem of determining the minimal size SDC is, therefore, the problem of constructing a function $G:\{x_1, x_2, \cdots, x_n\} \rightarrow \{b_1, b_2, \cdots, b_q\}$ where $G(x_i) = b_j$ if and only if there is a direct connection between input x_i and bit counter b_j, and certain relationships (discussed below) between the x_i's are taken into account. Note that G is a surjection or an onto mapping.

Definition 1: An input pair (x_i, x_j) is said to be *adjacent* if there exists at least one output function y_i, $i = 1, 2, \cdots, m$ which depends on both x_i and x_j.

In order to minimize the size of the SDC it is necessary to connect as many input lines to the same counter bit. A pair of input lines may be connected to the same counter bit if and only if they are not adjacent. We next show a constructive method of finding the function G.

Definition 2: A pair of inputs (x_i, x_j) is said to be *nonadjacent* if they are not adjacent.

We use the nonadjacency (NA) graph to create the function G. The NA graph is defined to have n vertices, designated by $x_1, x_2, \cdots, x_n$ to correspond to the n input lines. There is an arc between node x_i and node x_j if and only if the pair (x_i, x_j) is nonadjacent. By analyzing the NA graph it is possible to identify a maximal class of inputs that can be connected to the same counter bit. From the definition of the NA graph it is evident that the input lines $x_{i1}, x_{i2}, \cdots, x_{ik}$ can be connected to the same counter bit if they form a clique, i.e., the subgraph induced by $x_{i1}, x_{i2}, \cdots, x_{ik}$ is a complete graph.

Each clique in the NA graph defines a collection of inputs that can be connected to a single counter bit. Our objective is, therefore, to find all possible maximal cliques and then go through a prime-implicant-like covering procedure [3] to determine the minimum set of such cliques that cover all possible inputs. Each maximal clique obtained in the minimal cover will correspond to one bit counter. Thus the number of maximal cliques appearing in the minimal cover determine the minimal size of the SDC. Unfortunately, this problem is well known to be NP-complete and there is no known algorithm for solving it that is efficient in all cases. In many practical cases, though, it is possible to get good answers efficiently.

The function G is established in the following way. Let the set of maximal cliques appearing in the minimal cover be

$$\{\mathrm{MC}_i \mid \mathrm{MC}_i = \{x_{i1}, x_{i2}, \cdots, x_{ik_i}\}, \qquad i = 1, 2, \cdots, q\}.$$

The bit counters are then defined by referring each such MC to a different bit, namely,

$$\mathrm{MC}_i \Leftrightarrow b_i, \qquad i = 1, 2, \cdots, q.$$

Note that in general the cliques are not disjoint. In particular, suppose input x_i is included in several cliques, i.e.,

$$x_i \in \{\mathrm{MC}_{j1}, \mathrm{MC}_{j2}, \cdots, \mathrm{MC}_{jl}\}$$

then there are l choices for defining $G(x_i)$, namely,

$$G_r(x_i) = b_{jr}, \qquad r = 1, 2, \cdots, l.$$

Note, therefore, that there are multiple G functions possible, all of them leading to the same minimum counter size.

Example 1: Consider an eight input, five output macro, whose output functions are given by

$$y_1 = f_1(x_1, x_2, x_3, x_4)$$
$$y_2 = f_2(x_6, x_7, x_8)$$
$$y_3 = f_3(x_3, x_4, x_5, x_6)$$
$$y_4 = f_4(x_3, x_5, x_6, x_7)$$
$$y_5 = f_5(x_1, x_4, x_7, x_8).$$

According to Theorem 2, we can already identify the range of the SDC length

$$4 \le L \le 8.$$

To determine the minimum size SDC q we next create the NA graph (Fig. 2). The set of MC's for the graph of Fig. 2 is

$$\{\{x_2, x_5, x_8\}, \{x_1, x_5\}, \{x_1, x_6\}, \{x_2, x_6\}, \{x_2, x_7\}, \{x_3, x_8\}, \{x_4\}\}.$$

Operating a minimal covering procedure on these sets yields three possible covers

$$A_1 = \{\{x_2, x_5, x_8\}, \{x_1, x_6\}, \{x_2, x_7\}, \{x_3, x_8\}, \{x_4\}\}$$
$$A_2 = \{\{x_1, x_5\}, \{x_2, x_6\}, \{x_2, x_7\}, \{x_3, x_8\}, \{x_4\}\}$$
$$A_3 = \{\{x_1, x_5\}, \{x_1, x_6\}, \{x_2, x_7\}, \{x_3, x_8\}, \{x_4\}\}.$$

Note that the minimal size SDC for this example is 5. Note also that the set A_1 defines 4 possible G functions, as opposed to sets A_2 and A_3 which each define 2 functions.

Using the set A_2, the two possible G functions are

$$\begin{array}{lcl} G_1(x_1) = G_1(x_5) = b_1 & & G_2(x_1) = G_2(x_5) = b_1 \\ G_1(x_2) = G_1(x_6) = b_2 & & G_2(x_6) = b_2 \\ G_1(x_7) = b_3 & \text{or} & G_2(x_2) = G_2(x_7) = b_3 \\ G_1(x_3) = G_1(x_8) = b_4 & & G_2(x_3) = G_2(x_8) = b_4 \\ G_1(x_4) = b_5 & & G_2(x_4) = b_5 \end{array}$$

where

$$\{x_1, x_5\} \Leftrightarrow b_1$$
$$\{x_2, x_6\} \Leftrightarrow b_2$$
$$\{x_2, x_7\} \Leftrightarrow b_3$$
$$\{x_3, x_8\} \Leftrightarrow b_4$$
$$\{x_4\} \Leftrightarrow b_5.$$

The connection between the minimal size SDC and the macro inputs, as defined by the function G_1, is shown in Fig. 3.

IV. The Weighted Syndrome Sums

In Section II we have shown that it is possible to syndrome-test a multiinput–multioutput macro in one test procedure. A brute force implementation of this parallel syndrome-testing is to use m references one for each output syndrome. In a VLSI environment this may mean tens or hundreds of syndrome reference words. Since this trend in increasing density is expected to grow in the future, it may be attractive to try to reduce the number of references. Furthermore, one of the major alternatives to VLSI testing is to use a built-in test. This

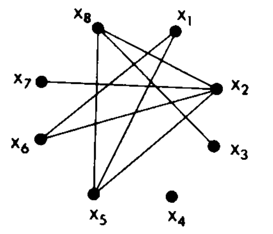

Fig. 2. The NA graph for Example 1.

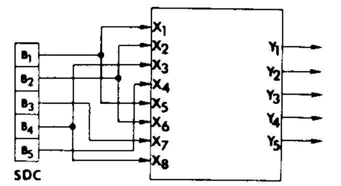

Fig. 3. The connection diagram between the SDC and the macro inputs as implied by the function G_1.

reduction in the number of output references makes the notion of syndrome-testing a very attractive approach to self-testing because it reduces the area overhead needed for implementation.

In this section we show a method of combining output syndromes in such a way as to reduce dramatically the number of references needed for syndrome-testing.

Let $\vec{S} = (S_1, S_2, \cdots, S_m)$ be a vector representing the output syndromes of the macro. Let also the function

$$g = \sum_{i=1}^{m} w_i Z_i$$

be a linear combination of the variables $Z_1, \cdots, Z_m$ with coefficients $w_1, w_2, \cdots, w_m$, where w_i is an integer, $i = 1, 2, \cdots, m$.

Definition 3: A weighted syndrome sum, (WSS), for an m-output macro with coefficients w_i, $i = 1, 2, \cdots, m$ is defined as

$$\mathrm{WSS} = g(\vec{S}).$$

Our intention is to use as references one or more WSS's instead of the complete collection of output syndromes. From here on we assume that each output function has been designed to be syndrome-testable [4], [5]. It is important to note that when using WSS's as references, as opposed to a complete collection of output syndromes, there may be undetectable faults, in the sense of the following definition.

Definition 4: Let the faulty syndromes induced by a fault f be denoted by S_i^f, $i = 1, 2, \cdots, m$. Let also $\Delta\vec{S} = \vec{S} - \vec{S}^f$. Then the fault f is said to be *weighted sum syndrome untestable* (WSSU) if

$$g(\Delta\vec{S}) = \sum_{i=1}^{m} w_i \Delta S_i = 0.$$

The following example demonstrates the concept of WSSU.

Example 2: Consider the circuit of Fig. 4. Suppose we use only one weighted sum with coefficients 2 and 3, namely let

$$g = 2Z_1 + 3Z_2.$$

Let input line x_3 be stuck at zero. Then the fault-free and faulty syndromes of the outputs are

$$S_1 = \frac{5}{16} \quad S_1^f = \frac{1}{8}$$

$$S_2 = \frac{3}{8} \quad S_2^f = \frac{1}{2}$$

Thus

$$\Delta S_1 = \frac{3}{16} \quad \Delta S_2 = -\frac{1}{8}$$

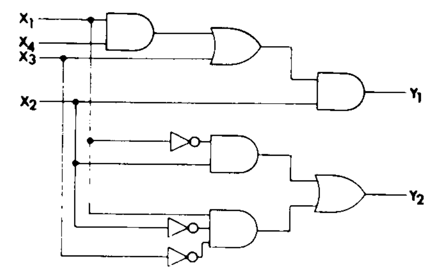

Fig. 4. The circuit for Example 2; the fault $x_3/0$ is WSSU.

and, therefore,

$$g(\Delta\vec{S}) = 2\Delta S_1 + 3\Delta S_3 = 0.$$

According to Definition 4 the fault $x_3/0$ is WSSU.

We would like to emphasize that as long as all coefficients are nonzero a fault which affects only one output function can never yield a WSSU condition, since every such fault will change the corresponding syndrome and therefore change the WSS. The choice of coefficients for a WSS is very important. In fact, the following theorem shows that we can always find coefficients for a WSS such that every single fault is WSS-testable.

Theorem 3: Suppose we are given a macro with output lines $y_1, \cdots, y_m$ and that line y_i is fed by $k_i \ge 1$ input lines. Then the WSS, with coefficients

$$w_i = 2^{\left(\sum_{j=1}^{i} k_j\right) + i}$$

detects all single faults.

Proof: We need only show that if some $\Delta\vec{S}_i \ne \vec{0}$, then $\sum_{i=1}^{m} w_i \Delta S_i \ne 0$. Let $i_o = \max\{i \mid \Delta S_i \ne 0\}$. Note that $|\Delta S_{i_0}| \ge 2^{-k_{i0}}$

$$|w_i \Delta S_i| \ge 2^{\left(\sum_{j=1}^{i} k_j\right) + i}$$

for all $i < i_o$ and $\Delta S_i = 0$ for all $i > i_o$. Thus

$$\left|\sum_{i=1}^{m} w_i \Delta S_i\right| \ge |w_{i_0} \Delta S_{i_0}| - \sum_{i<i_c} |w_i \Delta S_i| \ge 2^{\left(\sum_{j=1}^{i_0-1} k_j\right) + i_0} - \sum_{i<i_0} 2^{\left(\sum_{j=1}^{i} k_j\right) + i} > 0.$$

Q.E.D.

Note that Theorem 3 is not all that useful since the coefficients are so large, which forces us to store larger numbers. Indeed, the storage requirements are roughly the same as if we were to store the syndrome values for each output line individually. The point of Theorem 3 is to show that we can always get by with one WSS if necessary. Thus the WSS approach is no worse than simply storing the individual syndrome values. The benefit of the WSS approach is that often we can find a *small* number (ideally one) of WSS's with *small* coefficients which work for a given circuit. Below we discuss two approaches which can be used to find economical WSS schemes in many practical cases. Both cases depend on knowing something about the macro's structure. As noted earlier, if there is no overlap of various input lines any WSS having all its coefficients nonzero will get the job done. Theorem 4 generalizes this approach.

Theorem 4: If we pick r WSS's, defined by the weighted sums

$$g_i = \sum_{j=1}^{n} w_{ij} Z_j, \qquad i = 1, \cdots, r$$

such that any $r \times r$ submatrix selected from the matrix $W = [w_{ij}]_{i=1,\cdots,r;j=1,\cdots,m}$ is nonsingular, then a fault may be WSSU only if it affects at least $r + 1$ outputs.

Proof: A fault f will be WSSU if

$$\sum_{j=1}^{m} w_{ij}\Delta S_j = 0, \qquad i = 1, 2, \cdots, r.$$

We have to prove that at least $r + 1$ ΔS_j's are nonzero in the solution space. Or, if we express it differently, we have to prove that at most m-r-1 ΔS_j's are zeros in the solution space (excluding the trivial solution which corresponds to the fault-free condition). The argument goes as follows. Assume that there are m-r + k-1, $k \geq 1$, ΔS_j's which are zero. By substituting zeros for those ΔS_j's we are left with a homogeneous system of r equations in $r + 1$-k unknowns. Since all the columns of the matrix W are linearly independent because of the way we have chosen the coefficients w_{ij}, the rank of the matrix associated with the above homogeneous system is $r + 1$-k. Therefore, the solution for this homogeneous system is the trivial solution. This means that the only way we might have more than m-r-1 ΔS_j's which are zeros is by having a trivial solution which we have excluded. Q.E.D.

Corollary 1: A fault f may be WSSU only if it affects a line in an $r + 1$th degree overlap of the circuitry feeding the macro outputs.

Proof: Directly from Theorem 4.

It is very important to mention that both Theorem 4 and Corollary 1 refer to a necessary but not sufficient condition for a fault to be WSSU. Theorem 4 describes mathematical conditions for which an WSSU may exist. However, it is not guaranteed that a pattern of faulty syndromes occurring in the mathematical solution will in fact physically exist in the circuit. Thus it is very likely that most of the faults that may occur in the $r + 1$th degree overlap will end up being weighted sum syndrome-testable. Another approach is based on the number of different values which can occur as syndrome values at each output of the macro. The details are spelled out in Theorem 5.

Theorem 5: Suppose we are given a macro with output lines $y_1, \cdots, y_m$ such that d_i different syndrome values can appear at y_i as the result of a single fault or normal operation. There exists a set of r WSS's having n bit coefficients which detect all single faults if

$$\prod_{i=1}^{m} d_i < 2^{nr} + 1.$$

More generally, such a set exists if the number of different $\Delta\vec{S}$'s is $<2^{nr}$.

Proof: If we allow our coefficients to have n bits, we have a choice of 2^{nm} different WSS's. Given r of these WSS's we wish them to have the property that for each $\Delta\vec{S}$, some $j = 1, \cdots, r$ has the property that $g_j(\Delta\vec{S}) \neq 0$. Altogether we have 2^{nmr} different systems of r WSS's having n-bit coefficients. Of these at most $2^{n(m-1)r}$ have the property that $g_i(\Delta\vec{S}) = 0$ for all $i = 1, \cdots, m$. To see this, note that since some $\Delta S_{i_0} \neq 0$, once we specify the coefficients w_{jt}, $t \neq i_o$, there is at most one choice for w_{ji_0} (there may not be any since we are dealing with integers). Observe that there can be at most

$$\prod_{i=1}^{m} d_i - 1$$

different $\Delta\vec{S}$'s. Thus the number of inadequate systems is at most

$$\left(\prod_{i=1}^{m} d_i - 1\right)2^{n(m-1)r}.$$

As long as this number is $<2^{nmr}$, i.e.,

$$\prod_{i=1}^{m} d_i < 2^{nr} + 1$$

there must be at least one adequate system. The more general statement follows in the same way. Q.E.D.

Note that Theorem 5 does not furnish an efficient way to find an adequate system. Also, the upper bound is not the tightest possible, but it does give some idea of what can be done. In general, there may not be

$$\prod_{i=1}^{m} d_i - 1$$

different $\Delta\vec{S}$'s since the values are not necessarily independent. There is a rough estimate which might be used to estimate d_i based on the following observation. Given n inputs, there are 2^{2^n} different functions but only 2^n different syndrome values. Thus if a circuit has f different single faults, one could estimate the number of different syndrome values by $\log_2(f)$.

Example 3: Fig. 5 illustrates a binary to two's complement conversion circuit with eight inputs and six outputs. Consider the weighted sum defined by

$$g = \sum_{i=0}^{5} 2^i Z_i.$$

Table I displays the output syndromes and the corresponding values of the weighted syndrome sums for all single faults that may affect two or more outputs. We denote by $b/0$ line b stuck-at 0, and by $b/1$ line b stuck-at 1. There are 45 rows in Table I corresponding to the fault-free case and all relevant faulty cases. The first six columns of the table correspond to all output lines, while the last column refers to the value of the weighted sum. Each entry of the table in the first six columns represents the corresponding syndrome value, while the entries in the last column display the value of weighted syndrome sum. As seen from Table I none of the faulty weighted syndrome sums are identical to the fault-free one, and thus the circuit is not WSSU.

Example 4 demonstrates a WSSU case and suggests possible modifications to produce a weighted sum syndrome testable circuit.

Example 4: Fig. 6 describes a 4-input-4-output circuit. Suppose we choose two weighted sums defined by

$$g_1 = \sum_{i=0}^{3} 2^i z_i$$

$$g_2 = \sum_{i=0}^{3} 2^{3-i} z_i.$$

From the previous discussion we know that the potential WSSU faults must lie in a third degree overlap between the output functions. Thus the only candidates for consideration are lines x_1 and x_3.

The candidate WSSU faults yield the following weighted syndrome sums:

Fault-free case	$WSS_1 = 2.5$	$WSS_2 = 4.25$
$x_1/0$	$WSS_1 = 2.5$	$WSS_2 = 4.25$
$x_1/1$	$WSS_1 = 2.5$	$WSS_2 = 4.25$
$x_3/0$	$WSS_1 = 3.0$	$WSS_2 = 2.25$
$x_3/1$	$WSS_1 = 2.0$	$WSS_2 = 6.25$

Thus the faults $x_1/0$ and $x_1/1$ are WSSU.

In order to correct for this untestable condition we modify one of the functions in which x_1 is involved. In this example we decide to add an extra input to the AND gate realizing Y_o. This will correct the WSSU condition as evidenced by the following results:

Fault-free case	$WSS_1 = 2.375$	$WSS_2 = 3.25$
$x_1/0$	$WSS_1 = 2.5$	$WSS_2 = 4.25$
$x_1/1$	$WSS_1 = 2.25$	$WSS_2 = 2.25$

If we check the weighted syndrome sums induced by $x_3/0$ and $x_3/1$ we may find that these faults are still testable under the above modification.

Note that if we are using a WSS for a given macro, adding extra inputs and AND gates are essentially equivalent to dividing the appropriate coefficient of the WSS by a power of 2.

V. The Syndrome Self-Test Architecture

Fig. 7 illustrates the fundamental architecture of a chip in self-test mode. The chip has been partitioned to R syndrome partitions. We assume that each syndrome partition has LSSD chains on both inputs and outputs. The Syndrome Driver Counters (SDC) are a simple modification of input SRL chains. These SDC's work as regular SRL's in functional mode and as pure counters (or as linear feedback

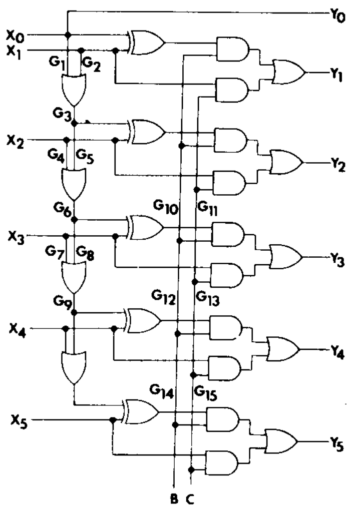

Fig. 5. A binary to two's complement conversion circuit.

TABLE I
THE SYNDROMES AND WEIGHTED SYNDROME SUMS OF VARIOUS FAULTS IN THE CIRCUIT OF FIG. 5

	Y0	Y1	Y2	Y3	Y4	Y5	WEIGHTED SUM
FAULT FREE	.5	.4375	.46875	.484375	.4921875	.49609375	30.875
X0/0	.0	.375	.4375	.46875	.484375	.4921875	29.75
X0/1	1.	.5	.5	.5	.5	.5	32.0
X1/0	.5	.25	.4375	.46875	.484375	.4921875	30.0
X1/1	.5	.625	.5	.5	.5	.5	31.75
X2/0	.5	.4375	.375	.46875	.484375	.4921875	30.125
X2/1	.5	.4375	.5625	.5	.5	.5	31.625
X3/0	.5	.4375	.46875	.4375	.484375	.4921875	30.25
X3/1	.5	.4375	.46875	.53125	.5	.5	31.5
X4/0	.5	.4375	.46875	.484375	.46875	.4921875	30.375
X4/1	.5	.4375	.46875	.484375	.515625	.5	31.375
B /0	.5	.25	.25	.25	.25	.25	16.0
B /1	.5	.625	.6875	.71875	.734375	.7421875	45.75
C /0	.5	.25	.25	.25	.25	.25	16.0
C /1	.5	.625	.6875	.71875	.734375	.7421875	45.75
G1/0	.5	.4375	.4375	.46875	.484375	.4921875	30.375
G1/1	.5	.4375	.5	.5	.5	.5	31.375
G2/0	.5	.4375	.4375	.46875	.484375	.4921875	30.375
G2/1	.5	.4375	.5	.5	.5	.5	31.375
G3/0	.5	.4375	.375	.4375	.46875	.484375	29.375
G3/1	.5	.4375	.5	.5	.5	.5	31.375
G4/0	.5	.4375	.46875	.46875	.484375	.4921875	30.5
G4/1	.5	.4375	.46875	.5	.5	.5	31.25
G5/0	.5	.4375	.46875	.4375	.46875	.484375	29.75
G5/1	.5	.4375	.46875	.5	.5	.5	31.25
G6/0	.5	.4375	.46875	.375	.4375	.46875	28.25
G6/1	.5	.4375	.46875	.5	.5	.5	31.25
G7/0	.5	.4375	.46875	.484375	.484375	.4921875	30.625
G7/1	.5	.4375	.46875	.484375	.5	.5	31.125
G8/0	.5	.4375	.46875	.484375	.4375	.46875	29.125
G8/1	.5	.4375	.46875	.484375	.5	.5	31.125
G9/0	.5	.4375	.46875	.484375	.375	.4375	27.125
G9/1	.5	4375	.46875	.484375	.5	.5	31.125
G10/0	.5	.25	.25	.484375	.4921875	.49609375	29.625
G10/1	.5	.625	.6875	.484375	.4921875	.49609375	32.125
G11/0	.5	.25	.25	.484375	.4921875	.49609375	29.625
G11/1	.5	.625	.6875	.484375	.4921875	.49609375	32.125
G12/0	.5	.25	.25	.25	.4921875	.49609375	27.75
G12/1	.5	.625	.6875	.71875	.4921875	.49609375	34.0
G13/0	.5	.25	.25	.25	.4921875	.49609375	27.75
G13/1	.5	.625	.6875	.71875	.4921875	.49609375	34.0
G14/0	.5	.25	.25	.25	.25	.49609375	23.875
G14/1	.5	.625	.6875	.71875	.4921875	.49609375	34.0
G15/0	.5	.25	.25	.25	.25	.49609375	23.875
G15/1	.5	.625	.6875	.71875	.4921875	.49609375	34.0

shift registers) in test mode. The SDC's are responsible for exercising the syndrome partitions by going through all possible combinations.

The multiplexer (MUX) selects one out of R syndrome partitions by connecting all its outputs to the syndrome measurement circuitry. Thus, the syndrome partitions are tested in sequence. If τ_i is the time needed for testing the ith syndrome partition

$$\sum_{i=1}^{R} \tau_i$$

is the total test time. The syndrome measurement circuitry depends on which testing approach is being used. The inputs to the measurement circuitry are the outputs of the syndrome partition currently under test. It provides as output the actual syndromes to be compared with the references stored in the Read Only Storage (ROS). The comparator compares the output of the syndrome measurement circuitry to the reference stored in the ROS. A fault indication bit is turned on if a discrepancy is observed. For the weighted syndrome

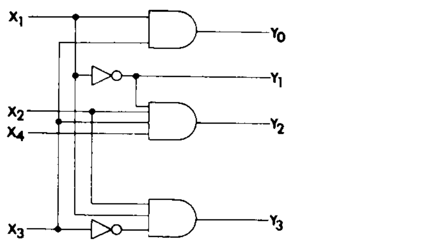

Fig. 6. An example which is WSSU for the functions defined in Example 4.

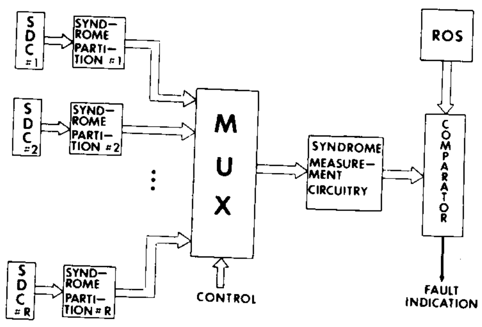

Fig. 7. The fundamental architecture for self-test.

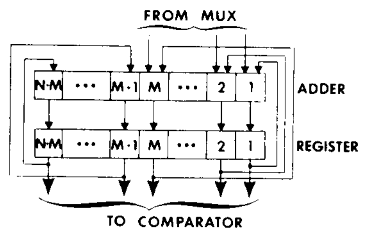

Fig. 8. Syndrome measurement circuitry for the weighted syndrome sums scheme

sums approach, with coefficients which are powers of 2, namely $w_i = 2^i$, the syndrome measurement circuitry is composed of an adder and a register (Fig. 8). If an upper bound to the number of inputs (outputs) to the syndrome partitions is $n(m)$, then the lengths of both the adder and the register is $n + m$. The weighting coefficients are implemented by connecting the MUX outputs to the proper adder inputs.

VI. SUMMARY AND CONCLUSIONS

The use of weighted syndrome sums in VLSI testing has been investigated in this paper. The design for weighted sum syndrome testability require partitioning into macros and selecting either one or several weighted syndrome sums for test reference. Thus the number of references needed for test implementation is roughly proportional to the number of macros. This low storage requirement has a natural application in self-test systems based on syndrome measurements.

REFERENCES

[1] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," in *Proc. 14th Annu. Design Automation Conf.*, pp. 462-468, June 1977.
[2] G. Markowsky, "Syndrome testability can be achieved by circuit modification," *IEEE Trans. Comput.*, vol. C-30, pp. 604-606, Aug. 1981.
[3] E. J. McCluskey, *Introduction to the Theory of Switching Circuits*, New York: McGraw-Hill, 1965.
[4] J. Savir, "Syndrome-testable design of combinational circuits," in *Proc. 9th Int. Symp. on Fault-Tolerant Computing*, pp. 137-140, June 1979.
[5] ——, "Syndrome-testable design of combinational circuits," *IEEE Trans. Comput.*, vol. C-29, pp. 442-451, June 1980; also see *IEEE Trans. Comput.*, vol. C-29, pp. 1012-1013, Nov. 1980.
[6] ——, "Syndrome-testing of 'syndrome-untestable' combinational circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 606-608, Aug. 1981.