Test Response Compaction Using Multiplexed Parity Trees

Krishnendu Chakrabarty, *Member, IEEE,* and John P. Hayes, *Fellow, IEEE*

Abstract— **Built-in self-testing requires test response streams from many observation points to be merged (space compaction) and compressed (time compaction) into a short signature. The compaction circuits should be transparent to error propagation in order to minimize aliasing, which occurs when a faulty response maps to the fault-free signature. We investigate the use of multiplexed parity trees (MPT's) for zero-aliasing space compaction. MPT's combine the error propagation properties of multiplexers and parity trees, and ensure zero aliasing via multistep compaction. We present two design techniques based on MPTs—output selection and fanout insertion—that eliminate aliasing for both deterministic and pseudorandom test sets. Our experiments with the ISCAS benchmark circuits show that zero aliasing can be achieved with small test sets and moderate hardware overhead. We also demonstrate that a very high percentage of single stuck-line faults in the compaction circuit are detected by the test patterns applied to the circuit under test.**

I. Introduction

BUILT-IN SELF-TESTING (BIST) is an attractive solution to the problem of testing complex VLSI circuits. However, as integrated circuit complexity increases, high-quality BIST can only be achieved if a large number of internal nodes are monitored during testing. Due to testing time and area considerations, the test responses from the different internal nodes must be merged into a signature, as shown in Fig. 1. However, test response compaction introduces the problem of aliasing, which occurs when a faulty circuit's signature maps to the fault-free signature. This reduces fault coverage, which is a serious problem when, as is often the case, high fault coverage (99% or more) is mandated. Design techniques that eliminate aliasing are therefore of considerable interest.

Multiple-input signature registers (MISR's) are often used to merge k response streams into a single composite signature. However, MISR's introduce aliasing and make the fault coverage hard to determine [1]. Although aliasing can be eliminated for MISR-based compaction schemes, this usually involves excessive computation in the design process [17]. MISR's require a latch or flip-flop for every observable output of the circuit under test, which may be infeasible because of high area overhead costs [4], [22]. Alternative approaches to space compaction include quadratic functions [11], output data modification [14], parity trees [18], and programmable space compaction [19], [22]. These techniques use probabilistic error models and search techniques such as genetic algorithms; however, aliasing-free compression is not guaranteed and the error models may not adequately reflect the faulty behavior of the circuit under test.

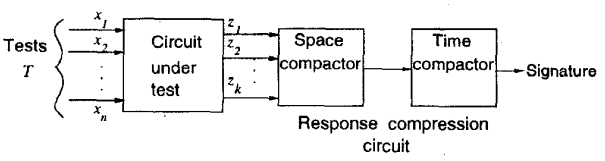


Fig. 1. A generic test response compaction scheme.

One way to handle test responses from a large number of observable test points is to replace the space compactor of Fig. 1 by a k-way multiplexer and apply the test set $\mathcal{T}k$ times. Although this is often done for compression schemes that are designed for single-output circuits, it leads to a k-fold increase in the test application time. The test time is already high for another reason—$\mathcal{T}$ is typically a pseudoexhaustive or pseudorandom test set generated by a linear-feedback shift register (LFSR) [1]. The test time can be decreased substantially if the "reduced" test sets generated by a typical ATPG program are used; this approach achieves 100% fault coverage with far fewer test patterns. A number of ISCAS benchmark circuits [3] can be tested for all single stuck-line (SSL) faults with very few test patterns. For example, c880 requires at most 18 test patterns and c6288 can be tested with only 12 patterns [8]. We are interested here in test methods that can use such reduced test sets, as well as the more usual LFSR-derived tests. In addition, we would like to eliminate entirely any aliasing due to response compaction.

In this paper, we introduce multiplexed parity trees (MPT's) for space compaction. MPT's provide aliasing-free compaction by combining the error propagation properties of multiplexers and parity trees. We also introduce the concept of *sequential* or *multistep compaction,* which allows error propagation in multiple time-steps, and describe two techniques that use sequential compaction to ensure zero aliasing. Sequential compaction is related to sequential transparency, a term introduced in [15] to study test propagation through modules by applying a sequence of values to their control inputs.

The organization of the paper is as follows. In Section II, we discuss fault sensitization parity, and present experimental data showing that space compaction using parity trees

Manuscript received December 16, 1994; revised April 2, 1996. This work was supported by the National Science Foundation under Grants MIP-9200526 and MIP-9503463. This paper was recommended by Associate Editor W. K. Fuchs.

K. Chakrabarty is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA.

J. P. Hayes is with the Advanced Computer Architecture Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA.

Publisher Item Identifier S 0278-0070(96)07185-0.

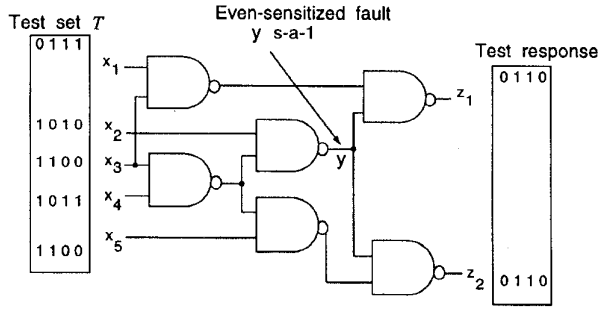


Fig. 2. The ISCAS c17 benchmark circuit with a minimal test set $\mathcal{T}$ for all SSL faults.

introduces very little aliasing. Section III presents a test generation approach that often achieves zero aliasing with parity trees. In Section IV, we describe two MPT-based compaction techniques, namely output selection and fanout insertion, that guarantee zero aliasing. Finally, Section V presents extensive experimental results to demonstrate that the parity tree can be adequately tested for SSL faults with the test patterns applied to the circuit under test.

II. Parity Tree Compaction

We first present experimental data showing that most SSL faults in the ISCAS combinational benchmark circuits are sensitized to an odd number of primary outputs. This suggests that a parity function is a good candidate for a space compaction circuit. We study the resulting parity compaction approach, which compresses a k-bit test response stream to a 1-b stream. Reddy *et al.* [18] have previously studied space compaction using parity, but they have not addressed the problem of eliminating aliasing.

Test generation programs are typically designed to produce tests that create a sensitized path from a fault site to a single observable output. If every test behaved this way, then we could combine all the outputs of a k-output circuit using a space compactor that implements the (even or odd) parity function $z = z_1 \oplus z_2 \oplus \cdots \oplus z_k$. Every error appearing on some z_i would then be transferred to z, thus guaranteeing detection at the output z of the parity generator. However, we would expect some tests to propagate errors to $m > 1$ primary outputs. If m is odd, the fault will still be detected at z. Aliasing occurs if m is even for all tests that detect a particular fault. We therefore pose the question: How many faults are sensitized to an even number of primary outputs by every test pattern that detect them? The following discussion addresses this question.

A test pattern t is *odd-sensitizing* for a fault f in a multiple-output circuit if it causes error propagation from f to an odd number of primary outputs. A test set $\mathcal{T}$ is odd-sensitizing for a fault f if there exists at least one test pattern t in $\mathcal{T}$ that is odd-sensitizing for f. The fault f is then said to be *odd-sensitized* by $\mathcal{T}$. A fault that is not odd-sensitized is called *even-sensitized.* A test set $\mathcal{T}$ is odd-sensitizing with respect to a set of faults F if it is odd-sensitizing for every fault f in F.

TABLE I
Even and Odd-Sensitized Faults in the ISCAS Combinational Benchmark Circuits for (a) Reduced Test Sets Generated by COMPACTEST. (b) Reduced Test Sets Generated by ATALANTA. (c) Pseudorandom Test Sets Generated by FSIM

ISCAS benchmark circuit	Number of test patterns	Number of detectable faults	Detectable even-sensitized faults	Percentage of odd-sensitized faults	CPU time (sec)
c17	4	22	1	95.45	0.22
c432	48	520	9	98.26	1.12
c499	59	750	63	91.60	1.52
c880	30	942	0	100	2.03
c1355	95	1566	8	99.48	5.20
c1908	129	1870	14	99.25	9.55
c2670	75	2628	79	96.97	31.40
c3540	113	3287	77	97.67	96.78
c5315	59	5291	87	98.36	120.72
c6288	23	7710	105	98.64	159.48
c7552	88	7419	58	99.22	249.07

(a)

ISCAS benchmark circuit	Number of test patterns	Number of detectable faults	Detectable even-sensitized faults	Percentage of odd-sensitized faults	CPU time (sec)
c17	5	22	1	95.45	0.22
c432	61	520	7	99.42	1.13
c499	63	750	9	99.87	1.39
c880	66	942	0	100	2.09
c1355	87	1566	7	99.55	5.16
c1908	127	1870	7	99.63	9.55
c2670	127	2628	5	99.81	37.70
c3540	174	3287	17	99.66	101.42
c5315	137	5291	45	99.94	138.11
c6288	40	7710	9	99.88	160.14
c7552	236	7419	59	99.21	281.12

(b)

ISCAS benchmark circuit	Number of test patterns	Number of detectable faults	Detectable even-sensitized faults	Percentage of odd-sensitized faults	CPU time (sec)
c432	2016	520	4	99.23	10.00
c499	2016	750	8	98.23	16.27
c880	3072	942	34	96.39	47.90
c6288	128	7710	10	99.87	177.13

(c)

For example, the ISCAS benchmark circuit c17 shown with a minimal test set in Fig. 2 has just one even-sensitized SSL fault. For most circuits, a surprisingly large number of faults are odd-sensitized by reduced test sets generated by a typical ATPG program. For example, Fujiwara and Yamamoto [7] have shown that 83–100% of the SSL faults in the combinational part of the ISCAS 89 benchmark circuits are odd-sensitized by their reduced test sets. They found that, on average, 94.6% of the SSL faults are odd-sensitized by these reduced test sets.

To investigate the odd sensitization properties of SSL faults further, we carried out an independent set of experiments for the ISCAS 85 benchmark circuits [3] using the COMPACTEST [16] and ATALANTA programs to generate reduced test sets that detect 100% of all detectable SSL faults. We

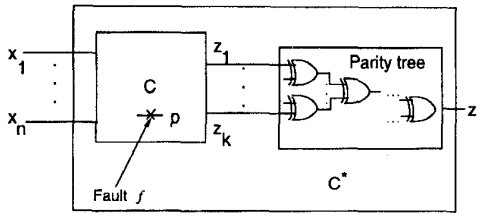


Fig. 3. Illustration of Theorem 1.

also used the FSIM fault simulation program [13] to generate pseudorandom test sets for the four smaller benchmark circuits. For each circuit, we determined the sensitization parities of all the faults in the collapsed fault lists by running a fault simulation program on a Sun Sparc 10/41 workstation. The results, listed in Table I, indicate that even-sensitized faults seldom occur, and that on average, 97.7 and 99% of the faults are odd-sensitized by COMPACTEST and ATALANTA, respectively. The CPU times required to identify the even-sensitized faults are also listed. These results show that parity compaction provides very high fault coverage.

It follows that in order to always achieve zero aliasing, we have to eliminate the small number of even-sensitized faults. This can be done either by altering the test set (one-step compaction) or by modifying the parity tree (multistep compaction). In the following section, we address the problem of generating a test set that guarantees odd sensitization for all SSL faults.

III. One-Step Compaction

Let $\mathcal{T}$ be a (reduced) test set that covers all the detectable faults F in a circuit. If $\mathcal{T}$ is not odd-sensitizing with respect to F, we can expand it to a larger test set $\mathcal{T}'$ that is odd-sensitizing with respect to F. Consider the c17 circuit once again, which has the minimal test set $\mathcal{T} = \{t_{15}, t_{21}, t_{26}, t_{18}\}$ shown in Fig. 2. To detect the even-sensitized fault y s-a-1, we add a test t_9 for y s-a-1 such as $x_1x_2x_3x_4x_5 = 01001$ to obtain $\mathcal{T}' = \mathcal{T} \bigcup \{t_9\}$. This complete odd-sensitizing test set for the c17 circuit was obtained in *ad hoc* fashion. An algorithmic approach for generating an odd-sensitizing test set for any circuit is suggested by the following theorem.

Theorem 1: Let $\mathcal{C}$ be the circuit under test with k outputs $z_1, z_2, \cdots, z_k$. Let $\mathcal{C}^*$ be the circuit with output $z = z_1 \oplus z_2 \oplus \cdots \oplus z_k$ constructed from $\mathcal{C}$ as shown in Fig. 3. A test pattern t is odd-sensitizing for the fault p s-a-d in $\mathcal{C}$ if and only if it detects the fault in $\mathcal{C}^*$.

Proof: Let z_i^f be the faulty response corresponding to the fault-free response z_i for the test pattern t. Suppose t is odd-sensitizing for the fault in C. Then $z_1^f \oplus z_2^f \oplus \cdots \oplus z_k^f \neq z_1 \oplus z_2 \oplus \cdots \oplus z_k$, and therefore $z^f \neq z$. This implies that the fault is detected in C^*. Next suppose that the fault is detected in C^*. Then $z^f \neq z$, which implies that $z_1^f \oplus z_2^f \oplus \cdots \oplus z_k^f \neq z_1 \oplus z_2 \oplus \cdots \oplus z_k$. This is possible if and only if $z_i \neq z_i^f$ for an odd number of i's. □

It is easily seen that a complete test set for the faults in C^* is an odd-sensitizing test set with respect to the faults in C.

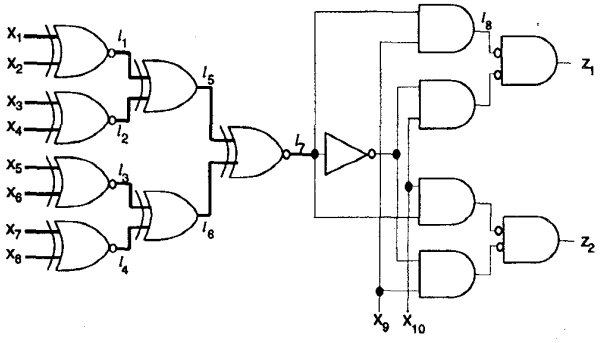


Fig. 4. The even sensitized SSL faults in the 74 180 parity generator circuit.

TABLE II
Fault Coverage Obtained for the ISCAS Circuits Using *ODD_TESTS*

Circuit C	Size of reduced test set T^* for C^*	Number of detectable faults in C	Number of detectable faults in C not detected by T^*	Percentage loss of fault coverage
c432	69	520	0	0
c499	53	750	24	3.20
c880	56	942	7	0.74
c1355	84	1566	24	1.53
c1908	144	1870	12	0.64
c2670	70	2628	815	31.01
c3540	183	3287	32	0.97
c5315	134	5291	49	0.93
c6288	41	7710	0	0
c7552	196	7419	251	3.38

Thus we can use the following straightforward procedure to derive an odd-sensitizing test set for the faults in C.

Procedure ODD_TESTS: To generate an odd-sensitizing test set for the faults in C:

1) Connect a k-input parity checker to the outputs of C to form a single-output circuit C^*.
2) Derive a complete, reduced test set $\mathcal{T}^*$ for C^* using any suitable ATPG procedure; $\mathcal{T}^*$ is an odd-sensitizing test set for the faults in C.

For the c17* circuit derived from c17, the input patterns $\mathcal{T}^* = \{t_5, t_3, t_{31}, t_{18}, t_{27}, t_{12}\}$ form a complete test set. Therefore, they form an odd-sensitizing test set for all faults in c17. This test generation approach to zero aliasing has the advantage that it is simple, can be easily automated, and does not require any circuit modification or special test-application hardware. For the c432 and c6288 circuits, zero aliasing can be achieved with complete test sets consisting of 69 and 41 patterns, respectively (see Table II).

The procedure *ODD_TESTS* has two drawbacks, however. The circuit C can contain faults that are even-sensitized by an exhaustive test set. Such detectable faults in C are undetectable in the modified circuit C^*. This implies that *ODD_TESTS* cannot, by itself, be guaranteed to find an odd-sensitizing test set for every circuit. For example, in the 74 180 parity generator circuit [20] shown in Fig. 4, the SSL faults on the bold lines are even-sensitized by every test set. Another example of a circuit with faults that are even-sensitized by

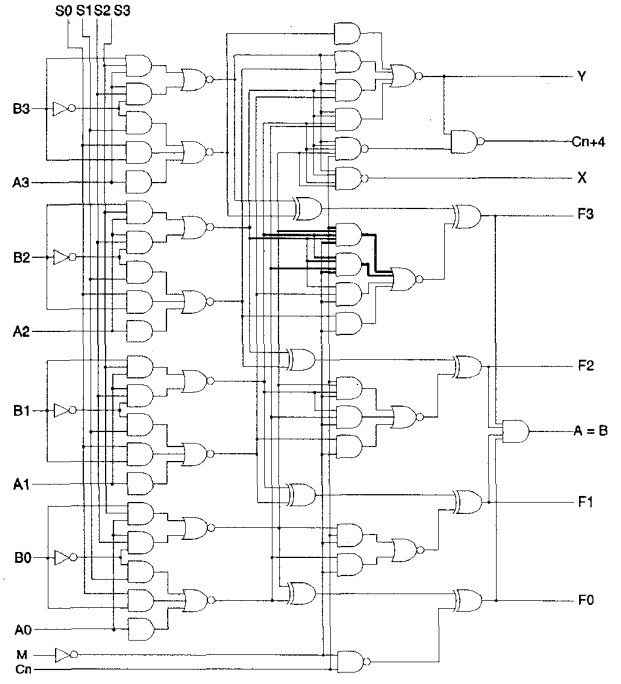


Fig. 5. The 74 181 ALU circuit with lines having even-sensitized faults highlighted.

every test set is the 74 181 ALU/function generator [20]. The logic diagram and lines with even-sensitized faults are shown in Fig. 5.

The second problem with procedure *ODD_TESTS* is that most ATPG programs backtrack excessively when they encounter a parity circuit with high fanin [2]. This was corroborated by experiments that we carried out by applying the procedure to some of the ISCAS 85 circuits. These experiments (Table II) demonstrate that *ODD_TESTS* introduces a small loss of fault coverage. For example, the c1355 circuit contains only 8 even-sensitized faults for a particular reduced test set (Table I). However, when test generation was carried out for the c1355* circuit, no test was found for 24 faults, or 1.45% of the detectable faults. In the c499 circuit, there are 63 even-sensitized faults in Table I; we can reduce this number to 24 by applying *ODD_TESTS,* but we cannot eliminate aliasing altogether.

This motivates the need for design techniques that guarantee zero aliasing. Two such techniques, based on multiplexed parity trees, are presented in the next section.

IV. Multistep Compaction

We now present multiplexed parity trees (MPT's), describe two MPT-based methods for zero-aliasing space compaction, and estimate the associated area overhead for the ISCAS benchmark circuits.

Multiplexed Parity Trees

A multiplexed parity tree $\mathcal{M}$, shown in Fig. 6 combines the error propagation properties of multiplexers and parity trees to provide zero-aliasing space compaction. The lines

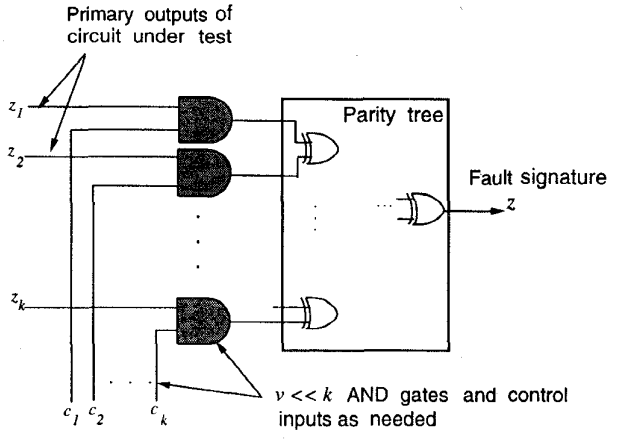


Fig. 6. A multiplexed parity tree $\mathcal{M}$.

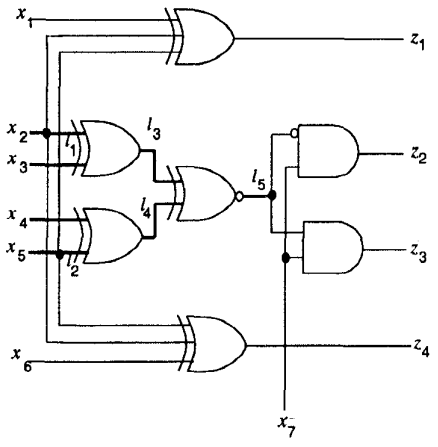
$z_1, z_2, \cdots, z_k$ are the primary outputs of the circuit under test while the c_i's are control inputs of $\mathcal{M}$ that select the z_i's to be fed to the parity tree. The number of control inputs and AND gates v varies from 1 to k depending on the even-sensitized faults in the circuit under test. The output function z of $\mathcal{M}$ can be written as $z = (a_1 + c_1)z_1 \oplus (a_2 + c_2)z_2 \oplus \cdots \oplus (a_k + c_k)z_k$, where $a_1, a_2, \cdots, a_k$ are binary variables whose 0–1 assignments determine the structure of $\mathcal{M}$. For example, let $k = 4$, $v = 2$, $a_1 = a_2 = 1$, and $a_3 = a_4 = 0$. This yields an MPT realizing the function $z = z_1 \oplus z_2 \oplus c_3 z_3 \oplus c_4 z_4$. We show later that zero aliasing can be achieved in $(v + 1)$ time-steps if $\sum_{i=1}^{k} a_i = v$. Depending on the values assigned to the c_i's and a_i's, $\mathcal{M}$ can realize various parity and multiplexer functions. These special cases are described below.

1) If a decoder is used to generate the c_i's, then $\mathcal{M}$ becomes a simple multiplexer that selects one of its data inputs $z_1, z_2, \cdots, z_k$ for direct connection to z.
2) If $v = k$ then $\mathcal{M}$ is a multiplexer that selects one of the 2^k parity functions of $z_1, z_2, \cdots, z_k$.
3) If $a_i = 1$ for all i, i.e., all the AND gates are logically disconnected, then $\mathcal{M}$ is a simple parity tree with inputs $z_1, z_2, \cdots, z_k$.

The values assigned to the control inputs of $\mathcal{M}$ affect error propagation through the parity tree. For example, if $v = k$, $c_1 = 1$, and $c_i = 0$ for all $i \neq 1$, then $\mathcal{M}$ propagates only those errors that appear on z_1. Thus we can enhance error propagation by applying the test set $\mathcal{T}$ to the circuit under test multiple times, and assigning different sets of values to the c_i's in each time-step. We term this process *multistep compaction*. For example, if $\mathcal{M}$ is a simple multiplexer that connects one of the z_i's to z, all error can be propagated through it in k time-steps. However, this leads to a k-fold increase in the test response length. We next describe two techniques for achieving zero aliasing based on MPT's and multistep compaction with only a small increase in the test response length.

Output Selection

In this method, we exploit the fact that a test pattern usually sensitizes a fault to only a few primary outputs. Therefore,



(a)

Even-sensitized faults	Primary outputs			
	z_1	z_2	z_3	z_4
x_2 s-a-0, s-a-1	1	1	1	1
x_3 s-a-0, s-a-1	0	1	1	0
x_4 s-a-0, s-a-1	0	1	1	0
x_5 s-a-0, s-a-1	1	1	1	1
l_1 s-a-0, s-a-1	0	1	1	0
l_2 s-a-0, s-a-1	0	1	1	0
l_3 s-a-0, s-a-1	0	1	1	0
l_4 s-a-0, s-a-1	0	1	1	0
l_5 s-a-0, s-a-1	0	1	1	0

(b)

Fig. 7. Zero aliasing via output selection. (a) An example circuit with even-sensitized faults. (b) Its sensitization table.

TABLE III
LOGIC OVERHEAD FOR OUTPUT SELECTION USING REDUCED TEST SETS GENERATED BY (a) COMPACTEST AND (b) ATALANTA

ISCAS benchmark circuit	Percentage overhead for parity tree	Number of primary outputs	Number of extra AND gates v	Percentage overhead for multiplexed p arity tree		CPU time (sec)
				AND gates and control logic	Total	
c432	4.17	7	2	2.18	6.35	1.15
c499	15.19	32	15	27.37	42.56	1.59
c880	6.88	26	0	0	6.88	2.03
c1355	5.82	32	3	1.40	7.22	5.21
c1908	3.21	25	3	0.96	4.17	9.57
c2670	14.21	140	8	2.32	16.53	31.44
c3540	1.43	22	9	2.00	3.43	96.98
c5315	5.57	123	10	1.54	7.11	120.91
c6288	1.29	32	18	2.76	4.05	159.53
c7552	3.48	108	10	1.07	4.55	249.36

(a)

ISCAS benchmark circuit	Percentage overhead for parity tree	Number of primary outputs	Number of extra AND gates v	Percentage overhead for multiplexed p arity tree		CPU time (sec)
				AND gates and control logic	Total	
c432	4.17	7	2	2.18	6.35	1.08
c499	15.19	32	1	1.82	17.01	1.75
c880	6.88	26	0	0	6.88	2.03
c1355	5.82	32	5	1.40	7.22	5.21
c1908	3.21	25	3	0.96	4.17	9.57
c2670	14.21	140	8	0.29	14.50	37.93
c3540	1.43	22	8	1.78	3.21	102.38
c5315	5.57	123	1	1.54	7.11	138.86
c6288	1.29	32	4	0.61	1.90	162.21
c7552	3.48	108	8	1.07	4.55	283.12

(b)

we can obtain odd sensitization by selecting the outputs that are fed to the AND gates of $\mathcal{M}$. (The remaining outputs are connected directly to the parity tree.) We first explain the basic idea using Fig. 6, and then present a practical design procedure. For a k-output circuit, we use an MPT with k AND gates and k primary inputs $c_1, c_2, \cdots, c_k$. To ensure zero aliasing, we now test the circuit in $k+1$ phases. In phase 0, we set $c_i = 1$ for $1 \leq i \leq k$ and apply the test set $\mathcal{T}$ for the original circuit. In phase i, we set $c_i = 0$ and $c_j = 1$ for $j \neq i$, and again apply $\mathcal{T}$. The control logic can be implemented either by a shift register arrangement as outlined in [5], or by a demultiplexer driven by an external controller. It is easy to see that all faults in the circuit under test are now odd-sensitized. The faults that are detected by a parity tree are propagated to the output of the multiplexed parity tree when $c_i = 1$, $1 \leq i \leq k$. Faults that are even-sensitized are propagated by setting the c_i's to 0, one at a time. However, the length of the test response is now $(k+1)|\mathcal{T}|$. This is comparable to the response length obtained if the space compactor of Fig. 1 is replaced by a k-way multiplexer. We next give a systematic method for reducing the test response length.

A k-output circuit seldom needs k extra AND gates since a fault is rarely sensitized to all the primary outputs. It suffices to consider only a subset of the primary outputs to which errors due to the even-sensitized faults are propagated. We view these outputs as test observation points that are fed to the AND gates of $\mathcal{M}$. To identify them, we construct a *sensitization table* with columns denoting primary outputs $z_1, z_2, \cdots, z_k$ and rows denoting the even-sensitized faults $f_1, f_2, \cdots, f_m$. The entry in the ith row and jth column is 1 if and only if fault f_i is sensitized to output z_j by some test pattern in the test set $\mathcal{T}$. We use the sensitization table to find a (minimal) *sensitization cover*, i.e., a (minimal) set of primary outputs to which every fault is sensitized. We then introduce AND gates and control inputs in the multiplexed parity tree corresponding to the lines in the sensitization cover.

Consider the example circuit of Fig. 7(a), where the sensitization table for an exhaustive test set is shown in Fig. 7(b). The minimal cover for this table is either $\{z_2\}$ or $\{z_3\}$, from which we select $\{z_3\}$. In general, the problem of finding the optimal sensitization cover C is equivalent to the NP-hard set covering problem [6]. We have implemented a fast, heuristic covering procedure *FIND_COVER* which is described in Fig. 8. Tables III and IV present the results of applying *FIND_COVER* to the ISCAS benchmark circuits for reduced and pseudorandom test sets. (The number of test patterns is listed in Table I.) The area overhead is calculated using the weighted gate count metric, and includes the cost of the multiplexed parity tree as well as the logic required to generate its control inputs. The total overhead is less than 8% for most circuits, and for many, especially the larger ones, it is less than 5%. The experimental results also show that the overhead depends on the choice of the test set.

The test response length for output selection is $(|C| + 1)|T|$, which is considerably shorter than $(k+1)|T|$. A key advantage of this method is that the circuit under test does not have to be modified, therefore it does not introduce any performance degradation for normal operation. We next

```
Procedure FIND_COVER {To find a near-minimal sensitization cover
begin
  remove-zero-columns(); {Remove all columns of the sensitization table with only zero entries}
  while not-all-rows-removed() do
  begin
    for i, j = 1 to number-of-columns do
      if dominates(i, j) then {dominates(i, j) = 1 if column i dominates column j}
          remove-column(j);
    k := max-column(); {select column k with the largest number of 1's}
    select-column(k);
    for i = 1 to number-of-rows do
      if sensitization-table[i][k] = 1 then
        remove-row(i);
    remove-column(k);
  end
end
```

Fig. 8. Procedure for finding a sensitization cover.

```
Procedure FANOUT_INSERTION {To add fanout branches to eliminate even-sensitization}
begin
  Q := φ; {Q is initially empty}
  R := {(l1, l2, ..., lj)}; {R is the set of chains}
  while R ≠ φ do
  begin
    Select a chain (l1, ..., lj) from R;
    Q := Q ⋃ lj; {Add the maximal element lj to Q }
    for each chain of R of the type (li1, li2, ..., lj) do
      R := R − (li1, ..., lj); {Remove chains with maximal element lj}
  end
  for each li ∈ Q do
    Create a fanout branch from li to an extra AND gate with a control input ci;
end
```

Fig. 9. The fanout insertion procedure.

TABLE IV
LOGIC OVERHEAD FOR OUTPUT SELECTION USING PSEUDORANDOM TEST SETS

ISCAS benchmark circuit	Percentage overhead for parity tree	Number of primary outputs	Number of extra AND gates v	Percentage overhead for multiplexed p arity tree		CPU time (sec)
				AND gates and control logic	Total	
c432	4.17	7	2	2.18	6.35	10.03
c499	15.19	32	4	7.29	22.48	16.35
c880	6.88	26	2	5.03	11.91	48.13
c6288	1.29	32	4	0.61	1.90	180.23

describe a different MPT-based technique for zero aliasing which sometimes requires less hardware overhead than output selection.

Fanout Insertion: All SSL faults in a fanout-free circuit are odd-sensitized because each fault must be sensitized to the only available primary output. Hence, fanout is a necessary condition for even sensitization. This suggests that we can ensure sensitization of all faults to an odd number of primary outputs by systematically inserting fanout branches in the circuit to eliminate even sensitization. These fanout branches form extra test observation points that are AND-ed with the control inputs in $\mathcal{M}$. Thus the number of AND gates and control inputs in $\mathcal{M}$ equals the number of such additional test observation points.

Let $L = \{l_1, l_2, \cdots, l_k\}$ be the set of all lines with even-sensitized faults in the circuit. We define a partial ordering on the lines in L as follows: l_j *dominates* l_i, denoted $l_i \leq l_j$, if all paths from l_i to all primary outputs pass through l_j. For example, in the 74180 circuit of Fig. 4, $l_1 \leq l_5$ but $l_1 \not\leq l_8$. (Note that $\leq$ is not the usual level-ordering relation.) We next group the lines in L into chains of the form $\{(l_1, l_2, \cdots, l_j)\}$ such that $l_1 \leq l_2 \cdots \leq l_j$. Let $R = \{(l_1, l_2, \cdots, l_j)\}$ be the set of chains thus formed. For the example circuit of Fig. 4, $L = \{x_1, x_2, \cdots, x_8, l_1, l_2, \cdots, l_7\}$ and $R = \{(x_1, l_1, l_5, l_7), (x_2, l_1, l_5, l_7), (x_3, l_2, l_5, l_7), (x_4, l_2, l_5, l_7), (x_5, l_3, l_5, l_7), (x_5, l_3, l_5, l_7), (x_7, l_4, l_5, l_7), (x_8, l_4, l_5, l_7)\}$.

A procedure *FANOUT_INSERTION* for adding fanout branches is given in Fig. 9, and applied to the 74 180 circuit in Fig. 10. It selects the maximal element of each dominance chain, i.e., it forms the smallest set Q of lines such that every line in the circuit is dominated by at least one line in this set. If $|Q| = v$, we add v fanout branches, v two-input AND gates, and v auxiliary primary outputs to the circuit. Even-sensitization is now eliminated if the modified circuit is tested in $v+1$ steps. In step 0, we set $c_1 = c_2 = \cdots = c_v = 0$ and apply the test set $\mathcal{T}$ for the original circuit. In each subsequent step i $(1 \leq i \leq v)$, we set $c_1 = \cdots = c_{i-1} = c_{i+1} = \cdots = c_v = 0$, $c_i = 1$, and apply $\mathcal{T}$. The control logic can be implemented either by a shift register arrangement as in the output selection method. The test application time and response length are increased by a factor of v. However, as shown in Tables V and VI, v is small for most of the ISCAS circuits, so the increase in the test response length is reasonable.

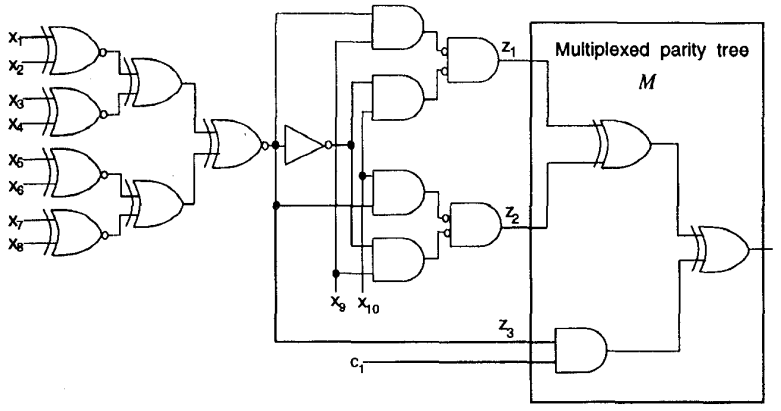


Fig. 10. Application of *FANOUT_INSERTION* to the 74 180 circuit.

TABLE V
LOGIC OVERHEAD FOR FANOUT INSERTION FOR REDUCED TEST SETS GENERATED USING (a) COMPACTEST, AND (b) ATALANTA

ISCAS benchmark circuit	Percentage overhead for parity tree	Number of extra AND gates v	Percentage overhead for multiplexed parity tree			CPU time (sec)
			AND gates & control logic	Exclusive-or gates	Total	
c432	4.17	4	6.97	6.25	13.22	1.13
c499	15.19	38	100	25.00	125.00	1.57
c880	6.88	0	0	6.88	6.88	2.03
c1355	5.82	1	0.40	6.01	6.41	5.22
c1908	3.21	1	0.28	3.34	3.62	9.57
c2670	14.21	11	3.78	14.92	18.70	31.43
c3540	1.43	1	0.14	1.49	1.63	97.00
c5315	5.57	70	15.35	7.21	22.56	144.57
c6288	1.29	7	0.55	1.46	2.01	159.52
c7552	3.48	1	0.07	3.51	3.58	249.25

(a)

ISCAS benchmark circuit	Percentage overhead for parity tree	Number of extra AND gates v	Percentage overhead for multiplexed parity tree			CPU time (sec)
			AND gates & control logic	Exclusive-or gates	Total	
c432	4.17	3	5.23	6.25	11.48	1.14
c499	15.19	6	15.79	17.15	32.94	1.49
c880	6.88	0	0	6.88	6.88	2.09
c1355	5.82	1	0.40	6.01	6.41	5.19
c1908	3.21	1	0.28	3.34	3.62	9.60
c2670	14.21	1	0.37	14.31	14.68	37.83
c3540	1.43	1	0.14	1.49	1.63	102.33
c5315	5.57	34	7.67	7.12	14.79	141.37
c6288	1.29	1	0.08	1.33	1.41	163.01
c7552	3.48	1	0.07	3.51	3.58	290.17

(b)

TABLE VI
LOGIC OVERHEAD FOR FANOUT INSERTION USING PSEUDORANDOM TEST SETS

ISCAS benchmark circuit	Percentage overhead for parity tree	Number of extra AND gates v	Percentage overhead for multiplexed parity tree			CPU time (sec)
			AND gates & control logic	Exclusive-or gates	Total	
c432	4.17	3	5.23	6.25	11.48	10.01
c499	15.19	6	15.79	17.15	32.94	16.32
c880	6.88	10	15.13	8.51	32.64	47.99
c6288	1.29	1	0.08	1.33	1.41	179.21

We next show that *FANOUT_INSERTION* ensures zero aliasing in the space compaction process. Every fault that is odd-sensitized in the original circuit generates an error that is propagated to the output of the space compactor when $c_j = 0$, for $1 \leq j \leq v$. Consider a fault $f = l$ s-a-d that is even-sensitized in the original circuit. Suppose $l \leq l_i$, where l_i s-a-d_i is also even-sensitized and $l_i \in Q$. (If l is not dominated by any other line, then $l_i = l$.) If $t \in T$ is a test for f, then it sensitizes f to an odd number of primary outputs in the modified circuit when $c_i = 1$ and $c_j = 0$ for $j \neq i$. Also, the addition of the fanout branches does not create any even-sensitized faults. Suppose a fanout branch is inserted at l_j. Consider a line l_i such that there is a path from l_i to a primary output via l_j. If a fault on l_i is odd-sensitized when there is no fanout branch then it remains odd-sensitized in the modified circuit since c_j can be set to 0.

For most circuits, the size of Q, the set of maximal lines in the chains, is small, and therefore the hardware overhead to ensure zero aliasing is modest. We have applied *FANOUT_INSERTION* to the ISCAS combinational benchmark circuits using the reduced test sets of Table I. The results for test sets generated by COMPACTEST, including the total CPU time for the identification of even-sensitized faults and fanout insertion, are given in Table V(a). The area overhead is estimated by the weighted gate count (gate count multiplied by the average fanin) assuming a 5-gate realization of a flip-flop [20]. The overhead for the multiplexed parity tree consists of a $(k + v)$-input parity tree, v 2-input AND gates, and the logic required to generate the control signals $c_1, c_2, \cdots, c_v$. The overhead figures for a k-input parity tree are also listed. For four of the benchmark circuits—c1908, c3540, c6288, and c7552—the total overhead is very low (less than 5%). For two other circuits, c880 and c1355, the total overhead is less than 10%.

Although the overhead is very high for the c499 circuit due to its relatively high fraction of even-sensitized faults, it can be reduced by a different choice of the test set. To demonstrate this, we applied *FANOUT_INSERTION* to the ISCAS circuits with test patterns generated by ATALANTA. The results, shown in Table V(b), indicate that the overhead does indeed depend on the choice of the test set. In all cases, the larger test sets generated by ATALANTA reduced the area overhead. In the c499 case, ATALANTA produced more tests, but the overhead for fanout insertion dropped from over 100% to less than 33%.

We also applied *FANOUT_INSERTION* to some ISCAS circuits for pseudorandom test sets generated using FSIM. The

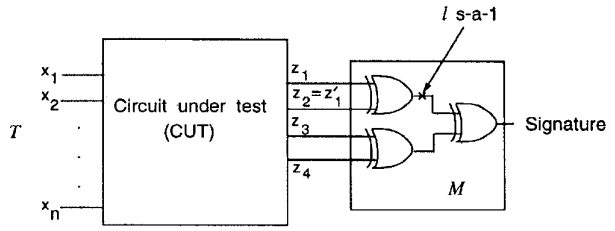


Fig. 11. An undetectable fault in $\mathcal{M}$ due to the circuit structure.

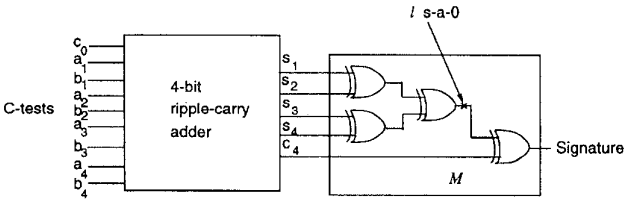


Fig. 12. An undetectable fault in $\mathcal{M}$ due to the test set.

TABLE VII
THE TEST PATTERNS AND FAULT-FREE RESPONSES FOR A 4-b RIPPLE-CARRY ADDER

c_0	a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4	s_1	s_2	s_3	s_4	c_4
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	1	1	1	1	1	0
0	1	0	1	0	1	0	1	0	1	1	1	1	1
1	0	1	0	1	0	1	0	1	0	0	0	0	1
1	1	0	1	0	1	0	1	0	0	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	0	0	0	1	0	1	0
1	0	0	1	1	0	0	1	1	1	0	1	0	1

results, shown in Table VI, show that the area overhead for the c499 circuit is considerably reduced if a pseudorandom test set is used. On the other hand, the area overhead for c880, which is less than 7% for the reduced test set of Table I, is now increased more than four times.

In general, output selection is more practical than fanout insertion since the latter requires some redesign to add fanout branches; it also requires additional test observation points. However, for many circuits (see Table V), only one extra fanout branch is needed so the hardware overhead is quite low. For a few circuits such as c6288 and c3540, the overhead for fanout insertion is less than that for output selection.

V. TESTING THE COMPACTION CIRCUIT

In this section, we examine the following question: Given a test set $\mathcal{T}$ for the circuit under test (CUT), how many faults in the multiplexed parity tree $\mathcal{M}$ are detected by $\mathcal{T}$? We make the assumption of single fault occurrence, i.e., SSL faults in $\mathcal{M}$ are not associated with faults in the CUT. We also restrict ourselves to faults on the inputs of the exclusive-or gates in $\mathcal{M}$. The s-a-0 faults on the AND gates of $\mathcal{M}$ are equivalent to s-a-0 faults on the exclusive-or gates.

Consider a fault $f = l$ s-a-d in $\mathcal{M}$. If there exists an input pattern in $\mathcal{T}$ that places a d' on l, then the fault effect, D or $\overline{D}$, is propagated to the output of the parity tree. This implies that f's signature differs from the fault-free signature, and the fault is detected. Let l be the parity of r primary outputs, i.e.,

$$l = z_{i_1} \oplus z_{i_2} \oplus \cdots \oplus z_{i_r}$$

If there exists $t_i \in \mathcal{T}$ that makes an odd number of the z_{i_j}'s 1 ($1 \leq j \leq r$), then the fault l s-a-0 is detected. Similarly, if there exists $t_j \in \mathcal{T}$ that makes an even number of the z_{i_j}'s 1, then the fault l s-a-1 is detected. Fig. 11 presents an example of an undetectable fault (l s-a-1) in the parity tree. Because of the relation between z_1 and z_2 ($z_2 = z_1'$), it is not possible to make $l = 0$, and hence the fault cannot be detected. In this example, untestability is a result of the structure of the circuit. In many cases, untestability is due to the choice of the test set. For example, if a 4-b ripple-carry adder is tested using 8 test patterns (Table VII), and the outputs are combined as in Fig. 12, the fault l s-a-0 is untestable, since $l = s_1 \oplus s_2 \oplus s_3 \oplus s_4$ is 0 for every test pattern.

However, since the detection of l s-a-d requires the existence of *at least one* test pattern that places a value d' on l, it might be expected that most faults in $\mathcal{M}$ are detected. Table VIII demonstrates that this is indeed the case for the ISCAS 85 circuits when reduced test sets generated by ATALANTA ($\mathcal{T}_1$) and COMPACTEST ($\mathcal{T}_2$) are applied, as well as when pseudorandom test patterns generated by FSIM ($\mathcal{T}_3$) are applied. Except once (for c5315 with $\mathcal{T}_2$), 100% coverage is obtained for the faults in $\mathcal{M}$. We also list the number of test patterns that are required to detect these faults. In all cases, a small number of test patterns is adequate to detect the faults in $\mathcal{M}$. Note that FSIM is a parallel pattern fault simulator, and in our experiments we set the word size to 32. This explains why the entries in the last three columns of the table (except for c5315 with $\mathcal{T}_2$ where all patterns are applied) are multiples of 32.

We next address the problem of testing the parity tree when the primitive elements are not exclusive-or gates, but are elementary gates—AND, OR, NAND, and NOR. In addition to detecting SSL faults on the inputs and the output of an exclusive-or module, we also have to detect all the "internal" faults. Any realization of a 2-input exclusive-or function requires four tests, i.e., all four input combinations have to be applied (Lemma 5 of [9]). This is equivalent to testing a parity tree composed of exclusive-or gates under a general fault model [12].

We enumerated the fault-free responses of four ISCAS circuits—c432, c499, c880, and c6288 for the reduced tests generated by ATALANTA and COMPACTEST. We then determined the exclusive-or modules that do not have all input combinations applied at their inputs. Thus we were able to calculate the fault coverage without performing fault simulation. The results are listed in Table IX. The fault coverage is less than 100% for the c499 and the c880 circuits.

We next consider ways in which the fault coverage of $\mathcal{M}$ can be made 100%. One solution to this problem is to modify $\mathcal{M}$ by permuting its inputs. The parity signature $z = z_1 \oplus z_2 \oplus \cdots \oplus z_k$ can be realized in a number of ways; for example, for $k = 4$, either $z = (z_1 \oplus z_2) \oplus (z_3 \oplus z_4)$, or $z = [(z_1 \oplus z_2) \oplus z_3] \oplus z_4$. However, this ordering (or the parity tree structure) is usually determined by factors such as ease of routing, interconnect length, and wire delay. Therefore, other methods for enhancing the testability of the parity tree must

TABLE VIII
NUMBER OF FAULTS IN $\mathcal{M}$ DETECTED BY TEST PATTERNS FOR THE CUT GENERATED BY COMPACTEST ($\mathcal{T}_1$), REDUCED TEST SETS GENERATED BY ATALANTA ($\mathcal{T}_2$), AND PSEUDORANDOM TEST PATTERNS GENERATED BY FSIM ($\mathcal{T}_3$)

ISCAS benchmark circuit	No. of faults in $\mathcal{M}$	No. of test patterns applied			No. of faults detected by the the test patterns			Percentage of faults detected by test patterns			No. of patterns required for fault detection		
		$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$
c432	12	48	54	512	12	12	12	100	100	100	32	32	32
c499	62	59	56	1024	62	62	62	100	100	100	32	32	32
c880	50	34	62	3616	50	50	50	100	100	100	32	32	128
c1355	62	95	87	5024	62	62	62	100	100	100	32	32	32
c1908	48	129	128	6016	48	48	48	100	100	100	32	32	32
c2670	278	75	127	100000	278	278	278	100	100	100	32	64	128
c3540	42	113	172	12000	42	42	42	100	100	100	32	32	32
c5315	244	59	137	6016	244	234	244	100	95.90	100	32	137	128
c6288	62	23	40	64	62	62	62	100	100	100	32	32	32
c7552	214	88	236	20000	214	214	214	100	100	100	64	64	128

TABLE IX
NUMBER OF FAULTS IN $\mathcal{M}$ DETECTED BY REDUCED TEST SETS GENERATED USING COMPACTEST ($\mathcal{T}_1$) AND ATALANTA ($\mathcal{T}_2$) FOR (a) AND-OR REALIZATION OF AN EXCLUSIVE-OR GATE, AND (b) 4-NAND REALIZATION OF AN EXCLUSIVE-OR GATE

ISCAS benchmark circuit	No. of faults in parity tree	No. of faults detected		Pe rcentage of fa ults detected	
		$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_1$	$\mathcal{T}_2$
c432	120	120	120	100	100
c499	620	600	572	96.67	92.26
c880	500	498	498	99.60	99.60
c6288	620	620	620	100	100

(a)

ISCAS benchmark circuit	No. of faults in parity tree	No. of faults detected		Pe rcentage of fa ults detected	
		$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_1$	$\mathcal{T}_2$
c432	132	132	132	100	100
c499	682	654	612	95.89	89.74
c880	550	548	548	99.64	99.64
c6288	682	682	682	100	100

(b)

be used. One such method is to add patterns to the given test set for the circuit.

The parity tree $\mathcal{M}$ can be tested under the general fault model with only four tests [12]. However, the inputs to $\mathcal{M}$ are the outputs of the circuit under test, hence it may not be possible to apply these tests. It appears that the more the logical dependence between the primary outputs, the harder it is to apply the tests to $\mathcal{M}$. The justification procedure of the D-algorithm can be used to find a suitable assignment to the primary inputs of the CUT, if one exists. Satisfiability-based methods can also be used, but they require that an algebraic representation be extracted from the CUT.

Yet another way of ensuring full testing of $\mathcal{M}$ is suggested by a recent theorem on fault detection in parity checkers [10], which states that a k-input parity tree can be completely tested for all stuck-line faults by the following $k+1$ patterns: $\{00\cdots0,\ 10\cdots0,\ 010\cdots0,\ \cdots,\ 0\cdots01\}$. These test patterns can be easily generated by an on-chip cyclic shift register and applied to the parity tree as shown in Fig. 13. However, the shift register and multiplexer can have a significant impact on the hardware overhead. For example, the associated overhead for c499 is excessive (125%) while for c432, it is a significant

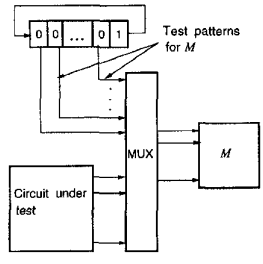


Fig. 13. On-chip test generation for $\mathcal{M}$.

11%. The overhead tends to decrease for larger circuits; for c6288 it is negligible (0.2%) and it is only 4% for c7552.

VI. CONCLUSION

We have presented an efficient zero-aliasing space compaction approach for multiple-output circuits based on multiplexed parity trees. We exploit the fact that typical ATPG programs sensitize SSL faults to an odd number of primary outputs, a property we call odd sensitization. We have demonstrated that test generation can often be used to achieve zero-aliasing one-step compaction with parity trees. An alternative approach to achieving zero aliasing is multistep (sequential) compaction, which is based on multiplexed parity trees. We have developed two techniques—output selection and fanout insertion—that achieve multistep transparency. We have shown that the associated hardware overhead is moderate for the ISCAS combinational benchmark circuits. An added benefit of this approach is that very high fault coverage is obtained for faults in the compaction circuit.

The results presented in this paper raise a number of open questions. We have seen that for some circuits such as the 74 180 parity checker and the 74 181 ALU, zero aliasing with parity trees can only be achieved via multistep compaction. However, for others such as c432, zero aliasing can also be achieved via one-step compaction by altering the test set. For the latter circuits, can we find efficient algorithmic techniques for constructing zero-aliasing test sets by adding test patterns to a reduced test set? Second, the logic overhead for both one-step and multistep compaction methods depends on the choice of the test set. Can zero-aliasing test sets be designed

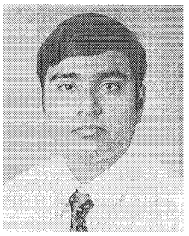
to reduce overhead, especially for circuits such as c499? We are currently investigating a two-pronged approach to the zero-aliasing problem that systematically combines test generation with the design of the multiplexed parity tree to reduce the area overhead.

ACKNOWLEDGMENT

We are grateful to Professor I. Pomeranz and Professor S. Reddy of the University of Iowa for providing us with COMPACTEST. We also thank Professor D. Ha of Virginia Tech. for making FSIM and ATALANTA available.

REFERENCES

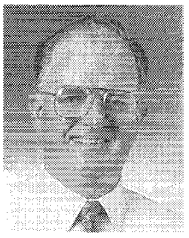
[1] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in Test for VLSI: Pseudorandom Techniques.* New York: Wiley, 1987.
[2] D. Brand, "Verification of large synthesized designs," in *Proc. 1993 Int. Conf. Computer-Aided Design,* 1993, pp. 534–537.
[3] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target simulator in Fortran," in *Proc. 1985 Int. Symp. Circuits Syst.,* 1985, pp. 695–698.
[4] J. Broseghini and D. H. Lenhert, "An ALU-based programmable MISR/pseudorandom generator for a MC68HC11 family self-test," in *Proc. 1993 Int. Test Conf.,* 1993, pp. 349–358.
[5] K. Chakrabarty and J. P. Hayes, "Efficient test response compression for multiple-output circuits," in *Proc. 1994 Int. Test Conf.,* 1994, pp. 501–510.
[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* Cambridge, MA: MIT Press, 1990.
[7] H. Fujiwara and A. Yamamoto, "Parity-scan design to reduce the cost of test application," *IEEE Trans. Computer-Aided Design,* vol. 12, pp. 1604–1611, Oct. 1993.
[8] M. C. Hansen and J. P. Hayes, "High-level test generation using physically-induced faults," in *Proc. 1995 VLSI Test Symp.*, pp. 20–28.
[9] J. P. Hayes, "On realizations of Boolean functions requiring a minimal or near-minimal number of tests," *IEEE Trans. Comput.,* vol. 20, pp. 1506–1513, Dec. 1971.
[10] W.-B. Jone and C.-J. Wu, "Multiple fault detection in parity checkers," *IEEE Trans. Comput.,* vol. 43, pp. 1096–1099, Sept. 1994.
[11] M. Karpovsky and P. Nagvajara, "Optimal time and space compression of test responses for VLSI devices," in *Proc. 1987 Int. Test Conf.,* 1987, pp. 523–529.
[12] W. H. Kautz, "Testing for faults in combinational logic arrays," in *Proc. 8th Ann. Switching and Automata Theory Symp.,* 1967, pp. 161–174.
[13] H. K. Lee and D. S. Ha, "An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation," in *Proc. 1991 Int. Test Conf.,* 1991, pp. 946–955.
[14] Y. K. Li and J. P. Robinson, "Space compaction methods with output data modification," *IEEE Trans. Computer-Aided Design,* vol. 6, pp. 290–294, Mar. 1987.
[15] B. T. Murray, "Hierarchical testing using precomputed tests for modules," Ph.D. dissertation, Univ. Michigan, 1994.
[16] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A method to generate compact test sets for combinational circuits," in *Proc. 1991 Int. Test Conf.,* 1991, pp. 194–203.
[17] I. Pomeranz, S. M. Reddy, and R. Tangirala, "On achieving zero aliasing for modeled faults," in *Proc. 1992 Eur. Design Automation Conf.,* Mar. 1992, pp. 291–299.
[18] S. M. Reddy, K. K. Saluja, and M. G. Karpovsky, "A data compression technique for built-in self-test," *IEEE Trans. Computers,* vol. 37, pp. 1151–1156, Sept. 1988.
[19] B. Tsuji, A. Ivanov, and Y. Zorian, "Selecting programmable space compactors for BIST using genetic algorithms," in *Proc. 1994 Asian Test Symp.*, 1994, pp. 233–241.
[20] *The TTL Data Book.* Dallas: Texas Instruments, 1988, vol. 2.
[21] Y. You and J. P. Hayes, "Implementation of VLSI self-testing by regularization," *IEEE Trans. Computer-Aided Design,* vol. 8, pp. 1261–1271, Jan. 1989.
[22] Y. Zorian and A. Ivanov, "Programmable space compaction for BIST," in *Proc. 1993 Int. Symp. Fault-Tolerant Computing,* 1993, pp. 340–349.



Krishnendu Chakrabarty (S'92–M'96) received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in computer science and engineering.

While at the University of Michigan, he was a research assistant with the Advanced Computer Architecture Laboratory of the Department of Electrical Engineering and Computer Science. Currently, he is an Assistant Professor of electrical and computer engineering at Boston University. His research interests are in computer-aided design of VLSI circuits and systems, testing, design verification, and fault-tolerant computing.

Dr. Chakrabarty is a member of ACM and Sigma Xi.



John P. Hayes (S'67–M'70–SM'81–F'85) received the B.E. degree from the National University of Ireland, Dublin, in 1965, and the M.S. and Ph.D. degrees from the University of Illinois, Urbana-Champaign, in 1967 and 1970, respectively, all in electrical engineering.

While at the University of Illinois, he participated in the design of the ILLIAC III computer. In 1970 he joined the Operations Research Group at the Shell Benelux Computing Center in The Hague, where he worked on mathematical programming. From 1972 to 1982 he was a faculty member of the Departments of Electrical Engineering–Systems and Computer Science of the University of Southern California, Los Angeles. He joined the University of Michigan in 1982. He was the founding director of the University of Michigan's Advanced Computer Architecture Laboratory. Currently, he is a Professor of electrical engineering and computer science at the University of Michigan, Ann Arbor, where he teaches and does research in the areas of computer architecture; computer-aided design, verification, and testing; VLSI design; and fault-tolerant embedded systems. He is the author of numerous technical papers and five books, including *Computer Architecture and Organization* (2nd ed., McGraw-Hill, 1988), *Hierarchical Modeling for VLSI Circuit Testing* (Kluwer, 1990; coauthored with D. Bhattacharya), and *Introduction to Digital Logic Design* (Addison-Wesley, 1993).

Dr. Hayes was Technical Program Chairman of the 1977 International Conference on Fault-Tolerant Computing, Los Angeles, and the 1991 International Computer Architecture Symposium, Toronto. He has served as editor of various technical journals, including the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and the *Journal of Electronic Testing*. He is a member of ACM and Sigma Xi.