HAMMING COUNT - A COMPACTION TESTING TECHNIQUE

Anita Gleason and Wen-Ben Jone

Department of Computer Science
New Mexico Tech
Socorro, NM 87801

ABSTRACT

A new signature compaction method called Hamming count (H-count) is introduced. H-count is similar to a reduced Walsh spectral coefficient test, and encompasses all syndrome testable faults. Hamming count has both a lower masking probability and a simpler circuit design than the index vector test. The proposed method presents an efficient and effective compaction technique.

1. INTRODUCTION

Testing offers a means to verify that a circuit meets designated requirements. Advances in VLSI technology have required changes in test application methods and apparatus. To facilitate circuit testing Built-In Self Testing (BIST) [1], [6-9] has been developed. With BIST, all test sets are generated and reference values stored on the chip. This alleviates the need for any special automatic test equipment to drive circuit input and test the response. By using circuit design techniques to enhance testability such as Level Sensitive Scan Design (LSSD) [3], partitioning of the chip into circuits with fewer than 20 to 25 inputs makes exhaustive testing feasible. Generation of all input combinations can be accomplished using a linear feedback shift register (LFSR) [2], [10] or autonomous LFSR [11]. BIST methods reduce the number of stored reference values by compacting the output responses of the circuit under test (CUT). The use of data compaction (or compression) indicates that some information may be lost due to the compression technique. This loss is referred to as masking. Masking is one criterion used to evaluate compaction methods. Another is the amount of chip area needed for the test hardware [1], [10].

Existing methods of output data compression are divided into three major groups -- (1) parity checking, (2) counting, and (3) LFSR methods [1], [8]. Included in the counting methods are ones count, syndrome test, Walsh coefficient verification [1] and index vector (IV) test [4]. In this paper, we propose another BIST counting technique to accomplish test output compaction. This method is called Hamming count (or H-count).

Although H-count is proposed to be used for BIST, it can also be used as an external circuit test methodology. H-count covers all ones count and syndrome testable circuits. Relation of H-count to Walsh spectral coefficients is discussed and results of H-count comparison to index vector testing are given.

2. H-COUNT - THE PROPOSED METHOD

A compaction method called Hamming count or H-count is introduced. As in syndrome and IV tests, H-count uses exhaustive testing and is based on functional properties of the CUT rather than the circuit structure.

Definition 1: The Hamming count (H-count) of an n-variable Boolean function f, denoted by H(f), is defined to be the n+1-tuple [c0, c1, . . . , cn], where c0 is the number of true minterms of f, and ci, i = 1, . . . , n is the number of true minterms of f in which the variable xi appears uncomplemented.

Example 1: The function f(x, y, z) = zy + zF + zyF + zyF

will produce an output value of one for input combinations 011, 010, 101, and 111. By H-count, we have

C0 = 4
C1 = 2 from terms 111, 101
C2 = 3 from 011, 010, 111, and
C3 = 3 from 011, 101, 111.

Hence, H(f) = [4, 2, 3, 3]. For f(x, y, z) with the fault z stuck-at 0 (denoted z/0), we get

f' (x, y, z) = zy + zF = zyF + zyF + zyF + zyF.

The counter values will be

C0 = 4,
C1 = 2,
C2 = 4, and
C3 = 2.

So H(f') = [4, 2, 4, 2], and the fault z/0 can be detected.

![Figure 1: H-count test approach](image)
The test approach for H-count is shown in Figure 1. Counter
C₀ totals the number of valid input combinations, that is, C₀
performs an ones count. The counters Cᵢ, i = 1,...,n, are incre-
mented whenever input xᵢ and CUT output z both have a logical
value of 1, so

\[ Cᵢ = Cᵢ + 1 \text{ if } xᵢ \cdot z = 1. \]

Figure 2(a) illustrates which counters are incremented by a
given input combination for n = 3 input variables. Counter
C₁ is incremented when there is a circuit output of 1 for minterms
4, 5, 6, 7; C₂ when there is a circuit output of 1 for minterms
2, 3, 6, 7; and C₃ when there is a circuit output of 1 for minterms
1, 3, 5, 7. A similarity between the combinations incremen-
ting a counter and those that determine the value of a
parity bit in Hamming Code [5] can be seen. Because of this
similarity, we have selected the name of Hamming count or H-
count for our test method.

Examining Example 1, we can see that H-count has detected a
fault that would be ones count undetectable since C₀ = 4 for
both functions. The necessity of counter C₀ is illustrated by
Example 2.

### Example 2

Given the function

\[ f(x,y,z) = \bar{x}y + \bar{y}z + \bar{x}y + xz. \]

Suppose some fault α caused \( f(x,y,z) \) to become

\[ f'(x,y,z) = \bar{x}y + \bar{y}z + \bar{x}y + \bar{x}z. \]

The syndromes for \( f \) and \( f' \), \( S(f) \) and \( S(f') \), without normal-
ization are

\[ S(f) = 4 \text{ and } S(f') = 5. \]

Hence, function \( f(x,y,z) \) is syndrome testable for fault α. If
counter C₀ is not included in the H-count method, then the
results, denoted by \( H' \) are

\[ H'(f) = [2, 3, 3] \text{ and } H'(f') = [2, 3, 3]. \]

We have \( H'(f') = H'(f) \), so fault α is undetectable without
C₀.

Thus the inclusion of counter C₀ is necessary to assure that
every fault detectable by syndrome testing is also detectable
using H-count. With the inclusion of counter C₀ the following
Lemmas hold.

**Lemma 2:** All syndrome testable functions are H-count
testable.

**Lemma 2':** All ones count testable functions are H-count
testable.

**Lemma 3:** All faults which affect an odd number of minterms
are H-count detectable.

But if there is a fault that affects exactly an even number of
minterms, it may be undetectable under H-count.

### 3. Comparisons to Other Counting Techniques

The H-count table of Figure 2(a) can be compared to that
shown in Figure 2(b) for a three variable Walsh coefficient test
using the definition of Walsh functions given in [1]. The com-
parison shows that the counter incrementing functions for
Cᵢ, i = 1,...,3 are analogous to those for primary functions
Wᵢ, i = 1,...,3, and C₀ to function W₀. Thus, our proposed
method is similar to a reduced Walsh coefficient test with no
decrementing.

Examining Example 1, we can see that H-count has detected a
fault that would be ones count undetectable since C₀ = 4 for
both functions. The necessity of counter C₀ is illustrated by
Example 2.
Example 4: For the function
\[ f (x, y, z) = \bar{y} z + x \bar{z} \bar{x} + x z + \bar{x} y, \]
we have \( H(f) = [4, 2, 3, 3] \) and \( IV(f) = [0, 1, 2, 1] \). In the presence of fault \( x \bar{z} \) the function becomes
\[ f'(x, y, z) = y = \bar{y} z + \bar{x} z + x z + \bar{x} y, \]
with \( H(f') = [4, 2, 3, 2] \) and \( IV(f') = [0, 1, 2, 1] \).

The above discussion and counterexample give proof to Lemma 4, and its converse Lemma 5.

Lemma 4: All IV detectable faults are not a subset of \( H \)-count detectable faults.

Lemma 5: All \( H \)-count detectable faults are not a subset of IV detectable faults.

A computer simulation comparing IV and \( H \)-count testing was made. The simulation exhaustively selected a number of minterms to be eliminated and an equal number of minterms as a replacement. The IV's of the dropped and added minterms were compared. If the vectors were the same, the fault was considered to be masked. \( H \)-count was evaluated in the same manner.

The results of the simulation are presented in Table 1. The first two columns of Table 1 represent information about the simulation. Here \( n \) represents the number of input variables, and \( r \) the number of minterms selected for elimination. The last column of the table gives the number of combinations possible for each particular value of \( n \) and \( r \). The percentage of undetected faults is given for both methods. From the results, we can see that \( H \)-count compared favorably with IV testing in the percentage of faults detected. The percentage of faults undetected by \( H \)-count was much lower than that of IV method.

Hardware considerations of both methods can be estimated from Figures 1 and 2. We can see that the test approach for \( H \)-count does not need a Bit Counter. Thus the design of \( H \)-count is simpler to implement. Table 2 gives the ratio of counter bits required by each method. Both methods use the same number of counters, but \( H \)-count requires approximately twice as many counter bits than IV test. However, in most cases, the increase in fault detection using \( H \)-count is enough to warrant this sacrifice of chip area.

4. CONCLUDING REMARKS

A new BIST method called \( H \)-count was introduced. \( H \)-count includes all syndrome testable functions. Plus, \( H \)-count detects faults which are missed by syndrome testing as in Example 1. It has been shown that \( H \)-count detects faults which were missed by index vector testing, and a counterexample was given to prove that all IV testable functions are not a subset of \( H \)-count testable functions. Simulation results confirm that \( H \)-count masks much fewer faults than IV test. Also, the design of \( H \)-count is simpler to implement than that of IV test, since the need for a Bit Counter module is eliminated.

\( H \)-count is directly related to testing using a reduced set of spectral coefficients based on the Walsh functions \( W \) through \( W_2 \). Thus, \( H \)-count could provide an alternative to a complete check of all \( 2^n \) spectral coefficients.

The simulation data in Table 1 provides a small view of the potential of the proposed \( H \)-count compaction test method. Simulation results for combinations with a larger number of input variables and/or minterms require an extensive amount of computing time. Mathematical analysis of the \( H \)-count masking probability to be smaller than that of other counting methods. Reduction of \( H \)-count hardware overhead by minimizing the number of required counters is under consideration and will appear in a future paper.

REFERENCES


### Table 1

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Ratio of Total Number of Counter Bits Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 4 6 8 10 12 14 16 18 20</td>
<td>H-count ( / ) IV Ratio</td>
</tr>
<tr>
<td>4 6 8 10 12 14 16 18 20</td>
<td>1.75 1.90 2.04 1.87 1.95 1.78 1.66 1.64 1.63</td>
</tr>
</tbody>
</table>

### Table 2

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Ratio of Total Number of Counter Bits Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 4 6 8 10 12 14 16 18 20</td>
<td>H-count ( / ) IV Ratio</td>
</tr>
<tr>
<td>4 6 8 10 12 14 16 18 20</td>
<td>1.75 1.90 2.04 1.87 1.95 1.78 1.66 1.64 1.63</td>
</tr>
</tbody>
</table>


