A Fuzzy Multiple Signature Compaction Scheme for BIST *

Yuejian Wu Andrii Ivanov
Department of Electrical Engineering
University of British Columbia
Vancouver, B.C., V6T 1Z4, Canada
e-mail: jian@ee.ubc.ca ivanov@ee.ubc.ca

Abstract — Compared to single signature analysis, checking multiple signatures yields smaller aliasing, easier fault coverage computation, shorter average test-time, and increased fault diagnosability. In conventional multiple signature (CMS) schemes, for a CUT to be declared good, at each check point, the signature obtained must match a specific reference. This strict one-to-one correspondence makes the CMS scheme complex to implement and expensive in terms of silicon area. In this paper, we propose a Fuzzy Multiple Signature compaction scheme in which the requirement for the one-to-one correspondence is removed. In the FMS scheme, for a CUT to be declared good, it suffices that the signature obtained at each check point correspond to any of a set of references.

1 Introduction
In Built-In Self-Test (BIST), both test pattern generation and output data evaluation are performed on chip [8,12,20]. The test output data evaluation generally consists of data compaction, whereby the CUT's output sequences are compacted into a signature of a few bits, and of comparison of the signature with a predetermined fault-free signature at the end of test to determine whether the CUT is good. However, due to the information loss through data compaction, some error responses may be mapped to the fault-free signature, thus causing some faulty circuits to escape detection [2]. This problem is called aliasing. Usually, fault coverage before compaction can be known by fault simulation. But, fault coverage after data compaction is usually only estimated since exact fault simulation is not computationally feasible for large CUTs. Although many probabilistic techniques have been developed, such techniques are difficult to use confidently when dealing with a specific CUT because of the statistical uncertainty.

At least two kinds of signature analysis have been proposed: single signature (SS) analysis and multiple signature (MS) analysis [2]. In the SS scheme, only a single final signature is checked at the end of a test session. In the MS scheme, however, in addition to the final signature, some intermediate signatures are checked as well. The MS scheme is known to significantly reduce aliasing [2,3,9]. Recent researches have shown that, besides small aliasing, checking multiple signatures has many other advantages over the SS scheme, such as easier fault coverage computation [13], shorter average test time [14], and increased fault diagnosability [16].

A major problem with the MS schemes is complex implementation and large associated hardware overhead. In [17], a multiple signature compaction scheme for BIST is proposed. By sharing some of the circuitry required by standard single signature BIST schemes, the scheme achieves very small aliasing at the expense of small silicon area. Unfortunately, its area requirements increase with test length. This renders it less attractive as CUTs get larger and require long test sets. In this paper, we propose a new output data compaction scheme for BIST which is simple to implement and requires little hardware overhead, even compared with SS schemes, and yet has all the advantages of conventional MS schemes.

2 Conventional Signature Analysis
2.1 Single Signature Analysis
In BIST with signature analysis [2], a linear feedback shift register (LFSR) is used to compact the CUT output sequence into a final signature (final LFSR state). Denoting the output sequence length by \( l \) and the number of the LFSR stages by \( k \), under the assumption of the equally likely error model [2], the aliasing probability \( P_{al} \) is:

\[
P_{al} = \frac{2^{l-k} - 1}{2^l - 1}.
\]

Assuming \( l >> k \) yields \( P_{al} \approx 2^{-k} \). This asymptotic result is well-known, and is also true for unequally likely error models [11].

2.2 Multiple Signature Analysis
Assuming the equally likely error model, it has been
shown that if two signatures are checked, i.e., an intermediate signature after the first \( l_1 \) bits of the sequence have been shifted into the signature analyzer, and a second signature at the end of the sequence of length \( l \), then the aliasing probability is [2]:

\[
P_{al}(2) = \left(\frac{2^{l_1-k} - 1}{2^{l_1} - 1}\right)\left(\frac{2^{l-k} - 1}{2^l - 1}\right).
\]

Assuming \( l_1 >> k \) yields \( P_{al}(2) \approx 2^{-2k} \). It is easy to prove that \( P_{al}(n) \approx 2^{-n\times k} \) if we check \( n \) signatures at the positions \( l_1, l_2, ..., l_n \), respectively, and assume \( k << l_1 < l_2 < ... < l_n = l \), where \( l_i \) is the position of the \( i \)th signature. We refer these positions as check points.

Assume the \( n \) actual signatures at the check points \( l_1, l_2, ..., l_n \) are \( s_1, s_2, ..., s_n \), i.e., the \( i \)th signature \( s_i \) is checked after \( l_i \) test patterns have been applied to the CUT. Once the check points are fixed, by logic simulation, one can easily determine the \( n \) corresponding fault-free signatures or references, \( r_1, r_2, ..., r_n \). When testing the CUT, the \( i \)th signature \( s_i \) is compared with its corresponding reference \( r_i \) at the \( i \)th check point. Thus, in this scheme, the signatures and references must correspond on a one-to-one basis for a CUT to be declared good. Such a one-to-one correspondence between specific references and signatures at specific check points makes the conventional multiple signature (CMS) schemes complicated and expensive to implement. Fig. 1 illustrates the implementation of such a CMS scheme. As shown in addition to a ROM for storing the \( n \) references, extra circuitry is required for generating ROM addresses and comparing ROM output with the signatures from the LFSR. Usually, the extra circuitry takes more silicon area than that of the ROM itself. This is because implementing the address generator requires a form of counter, which is area-consuming compared to combinational circuits and ROM cells. Also, comparing the references with the signatures requires either a parallel or a serial comparator with some kind of parallel to serial converter to convert each \( k \)-bit reference into a serial \( k \)-bit sequence [3][17][19]. A parallel comparator or a parallel to serial converter also takes considerable silicon area if \( k \) is not very small. Furthermore, connections among these functional blocks can also take much area.

### 3 Fuzzy Multiple Signature Analysis

As mentioned in Section 2.2, the complexity of checking multiple signatures is mainly due to the requirement of the one-to-one correspondence between the reference and signature. By removing this strict requirement we can get a much simpler data compactor. This is the basic idea of the proposed scheme, referred to as a Fuzzy Multiple Signature (FMS) scheme since it consists of checking multiple signatures, however does not impose the one-to-one correspondence between each reference and signature.

#### 3.1 Basis

Like the CMS scheme, the FMS scheme also checks \( n \) signatures at check points, \( l_1, l_2, ..., l_n \). But, unlike the CMS scheme where a signature \( s_i \) is compared with a specific reference \( r_i \) at check point \( l_i \) in the FMS scheme, each \( s_i \) is compared with the whole set of references \( r_1, r_2, ..., r_n \). A signature \( s_i \) is considered good if it matches any of the references in the reference set. Therefore, in the FMS scheme, for a CUT to be declared good, it suffices that the signature obtained from the LFSR at each check point corresponds to any of the references \( r_1, r_2, ..., r_n \). Thus, in the FMS scheme, the requirement for the one-to-one reference – signature correspondence is removed. This fuzziness introduced due to not knowing the reference – signature correspondence may result in a small increase of aliasing compared to the CMS scheme for given \( k \) and \( n \). But, this can be easily compensated for by the reduced complexity of the FMS scheme compared to the CMS scheme. Otherwise, the FMS scheme has all the advantages that the CMS scheme has over SS schemes.

#### 3.2 Implementation

Since the one-to-one correspondence between references and signatures no longer exists, implementing the FMS scheme is very simple. As shown in Fig. 2, the FMS scheme consists of a Signature Observer (SO) and a LFSR. The LFSR collects signatures. The SO checks each signature, and generates Pass/Fail signals. At each check point, if the signature generated by the LFSR matches any of the references, the SO outputs a Pass signal, say logic 0. Otherwise, the SO outputs a Fail signal, say logic 1. The Fail signal can be fed to a test controller to terminate the testing and declare the CUT as faulty. If the SO outputs Pass signals at all of the predetermined check points, the CUT is declared good.

The SO is basically a \( k \)-input, 1-output decoder which decodes a set of fault-free signatures from the LFSR. E.g., to check \( n \) \( k \)-bit signatures, the SO can be easily implemented with a \( k \)-input, 1-output, \( m \)-cube PLA, where \( m \leq n \), or other non-PLA-type log logic structures [15][19].
3.3 FMS Aliasing Performance

Assume the compaction of a $l$-bit random sequence into a $k$-bit signature, and $r_1$ to be the only valid reference. The total number of sequences (including the fault-free one) that map to $r_1$ is $2^{l-k}$. If we assume that distinct references $r_1$ and $r_2$ are both acceptable, then the total number of sequences that map to either $r_1$ or $r_2$ is $2 \times 2^{l-k}$ since the sequences that map to one will not map to the other. Thus, if we assume that $m$ distinct references, $r_1, r_2, ..., r_m$ are acceptable, the total number of $l$-bit sequences that map to any one of the $m$ references is $m \times 2^{l-k}$. Excluding the fault-free sequence, the deception volume [1] for such case is thus $m \times 2^{l-k} - 1$. Since there are $2^l - 1$ possible error sequences, assuming all are equally likely, the aliasing probability when $m$ distinct references are acceptable at a single check point is:

$$P_{al} = \frac{m2^{l-k} - 1}{2^l - 1}. \quad (3)$$

Assuming $l >> k$ yields $P_{al} \approx m2^{-k}$.

The FMS scheme checks $n$ signatures at check points, $t_1, t_2, ..., t_n$, against a set of $m$ references, with $m \leq n$ because there can be at most $n$ distinct references if we check $n$ signatures. However, some references may happen to be, or be made identical [18], thus making $m < n$.

Using the arguments presented in Section 2.1 for the aliasing probability of CMS schemes, the following aliasing probability results for the FMS scheme:

$$P_{FMS} = (\frac{m2^{l-k} - 1}{2^l - 1})^n. \quad (4)$$

Assuming $l >> k$ yields $P_{FMS} \approx [m2^{-k}]^n$. Clearly, for fixed $k$ and $n$, the best-case aliasing occurs for $m = 1$ [18]. The worst case occurs when $m = n$, for which $P_{FMS} \approx [n2^{-k}]^n$.

The following analysis assumes the worst-case scenario (i.e., $m = n$). To study the aliasing performance of the FMS scheme, we define a FMS scheme equivalent length, $L_{eFMS}$, as a figure of merit. For a given aliasing probability in the FMS scheme, we define $L_{eFMS}$ to be the length of a LFSR that yields the same aliasing probability in a SS scheme. Ideally, $L_{eFMS}$ should be as large as possible to minimize aliasing. Assume $L_{eFMS}$ to be a continuous variable. Since $P_{SS} \approx 2^{-k}$ and $P_{FMS} \approx [n2^{-k}]^n$, then

$$2^{-L_{eFMS}} = [n2^{-k}]^n. \quad (5)$$

Solving for $L_{eFMS}$ yields:

$$L_{eFMS} = n(k - \log_2(n)). \quad (6)$$

In comparison, the equivalent length of a CMS scheme is $L_{eCMS} = nk$; while that of a SS scheme is $L_{eSS} = k$. Thus, we have $L_{eFMS} = L_{eCMS} - n\log_2(n)$ for given $n$ and $k$.

**Example 1**: Assuming $k = 16$ and $n = 4$, $L_{eSS} = 16$, $L_{eCMS} = 64$, and $L_{eFMS} = 64 - 4 \times 2 = 56$. Thus, $P_{CMS} = 2^{-64}$, $P_{FMS} = 2^{-56}$, and $P_{SS} = 2^{-16}$. Here, the aliasing probability of the FMS scheme is 2 orders of magnitude greater than that of the CMS scheme, but still 12 orders of magnitude smaller than that of the SS scheme.

With the CMS scheme, the equivalent length increases linearly with both $k$ and $n$. With the FMS scheme, however, the equivalent length increases linearly with $k$ but not with $n$. For fixed $k$, as $n$ increases, $L_{eFMS}$ peaks and then decreases. When $n = 2^k$, $L_{eFMS} = 0$. Fig. 3 shows an example of the $L_{eFMS}$ as a function of $n$ for $k = 8$, $k = 9$, $k = 12$ and $k = 16$.

![Aliasing performance of the FMS scheme](image-url)

**Fig. 3. Aliasing performance of the FMS scheme**

4 Comparative Evaluation of the FMS Scheme

In this section, we compare the aliasing performance and hardware requirements of the FSM scheme with those of the SS and CMS schemes, respectively.

4.1 FMS vs. SS

To achieve an aliasing probability of $2^{-k}$, the SS scheme requires a $k$-bit LFSR. To achieve the same aliasing probability, the FMS scheme only requires a $(k/n + \log_2(n))$-bit LFSR plus the area for the signature observer (SO). For the following more detailed area comparisons, a PLA implementation of the SO is assumed. Since each PLA-input variable corresponds to two lines in the AND plane of a PLA, and since the drivers in the PLA take an area of about 8 cubes [7], the normalized area of a $k$-input, $s$-output, $n$-cube PLA is $(n + 8) \times (k \times 2 + s)$ units. We base our area estimate comparisons on the actual layout of a PLA and a 16-bit LFSR, using the CadenceTM...
automatic place and route tool, and 3 um double-metal CMOS technology. The LFSR was built with static D flip-flops, and measured about 1.38 × 10^6 um^2. Actual layout revealed that a 12-input, 4-output, 64-cube PLA takes about the same area as a 16-bit LFSR. According to the above analysis, this PLA requires an area of (64 + 8) × (12 × 2 + 4) = 1016 units. Therefore, we assume that a PLA of 2016 units corresponds to the area of a 16-bit LFSR.

We illustrate the comparison of the FMS and the SS schemes by the following example.

**Example 2:** If k = 9 and n = 32, P_{FMS} = (32 × 2^9)32 = 2^128. In this case, the required hardware is a 9-stage LFSR, a 9-input, 1-output, 32-cube PLA to implement the SO. The PLA requires (32 + 8) × (9 × 2 + 1) = 760 units of area, which is about 37.7% of the size of a 16-bit LFSR, or about the size of a 6-bit LFSR. Thus, the total area overhead for the FMS scheme to achieve an area overhead of the CMS scheme becomes higher. This is because the aliasing probability of the FMS scheme is not a linear function of n when k is fixed (see Fig.3).

When k = 12, the FMS scheme requires less area overhead than the CMS scheme as P_{al} > 2^{-490}. If k = 16, the FMS scheme requires less overhead as P_{al} > 2^{-1000}. For a fixed k, when P_{al} is smaller than a certain value, the area overhead of the FMS scheme becomes higher. This example shows that the FMS scheme requires less overhead as the number of test points increases.

![Fig. 4. Area overhead of the FMS and CMS schemes.](image)

### 5 Observation and Control

#### 5.1 Test Result Observation

For the observation of the test result, we consider two different cases. In the first case, if there is a Pass/Fail pin available, a "zero" detector may be used, as shown in Fig.5, to detect the Pass/Fail signal from the SO. Prior to testing a CUT, the "zero" detector is preset to "1". Thus, setting the Pass/Fail pin to 0. When a signature is checked, the controller temporarily sets the clk signal to 1. This sensitizes the "zero" detector to the SO output. Once a Fail signal (a "1" at the SO output) is detected, the detector outputs and keeps a Fail signal, "1", at the Pass/Fail pin. The "zero" detector can be shared by all self-testable blocks on a chip. When shared, the detector outputs and keeps a Fail signal if any of the self-testable blocks is found faulty. This Fail signal can be used to terminate testing thereby saving test time.

![Fig. 5. Observation of the FMS Test Result.](image)
The controller required for the FMS scheme is the same as for the CMS scheme. In the following, we show that the hardware overhead of the FMS scheme's controller can be made as small as that required for the control of a SS scheme.

For the control of a SS scheme, an on-chip counter or LFSR (which may be simultaneously used for pseudo random pattern generation) is required to count the applied test vectors [6][8]. When the final count is reached, a decoder detects this final count and generates a signal to stop the test and perform final signature evaluation.

For the control of the FMS scheme, we consider three possible cases depending on the scheduling of the check points. In the first case, the check points are arbitrary, i.e., the test length between check points \( t_i \) and \( t_{i+1} \) can be arbitrarily selected. To control the FMS scheme, like the SS scheme, an on-chip counter is required to count the applied test vectors. When a check point is reached, a decoder detects the corresponding count and outputs the \( \text{chk} \) signal to enable the evaluation of a signature. Unlike a SS scheme's controller where the decoder detects only the final count, the decoder in this case must decode all the counts corresponding to the check points, and is thus considerably larger. This is the worst case in terms of hardware overhead among the possible implementations of the FMS scheme's controller.

In the second case, the scheduling of the check points follows a regular equidistant pattern, i.e., the test length between check points \( t_i \) and \( t_{i+1} \) is constant for all \( i \). A convenient constant is \( 2^q \), where \( q \) is an integer. To control the FMS scheme in this case, we may simply split the counter required in the SS scheme into two smaller ones, say \( C_1 \) and \( C_2 \). \( C_1 \) is a \( q \)-bit counter that counts the test length between two adjacent check points, i.e., \( 2^q \). \( C_2 \) is a \( \log_2(n) \)-bit counter that counts the number of signatures to be checked. Assume the final count to be 0 for both \( C_1 \) and \( C_2 \). Every time \( C_1 \) decrements to 0, a decoder decodes the 0 and generates the \( \text{chk} \) signal to enable the evaluation of a signature. If the signature is incorrect testing can be terminated and the \( \text{CUT} \) declared faulty. Otherwise, \( C_2 \) is decremented by 1, and testing continues. When both \( C_1 \) and \( C_2 \) reach the 0 state, the test is complete. The hardware requirement of the controller in this case is almost the same as that for the SS scheme since the total length of \( C_1 \) and \( C_2 \) is the same as that of the counter used in the SS scheme, and the total complexity of the two decoders to decode the 0s from \( C_1 \) and \( C_2 \) is the same as that of the decoder in the SS scheme. This is the best case in terms of hardware overhead among the possible implementations of the FMS scheme's controller.

The third case lies between the first two, i.e., the check points are selected such that the test length between two adjacent check points, \( t_i \) and \( t_{i+1} \), is variable but constrained to values of \( 2^q \), where \( q_i \) is an integer. In this case, we may still use counters \( C_1 \) and \( C_2 \) to count the applied test vectors. \( C_1 \) must be of length \( q_i \), where \( q_i = \min(q) \). Two decoders are required. One decodes the 0 state from \( C_1 \), while the other decodes the counts of \( 2^q \) from \( C_2 \), \( i = 1, \ldots, n-1 \). Every time \( C_1 \) reaches 0, and \( C_2 \) reaches \( 2^q \), a signature is checked. The hardware overhead in this case is between the first two cases, since the decoder for \( C_1 \) detects only one count, but the decoder for \( C_2 \) has to detect \( n \) counts corresponding to the check points.

The above discussion assumes a counter for controlling the FMS scheme. A LFSR random pattern generator can be shared as a counter to count the applied test vectors, e.g., as in the BIST controller of the Intel 80386 [8]. The FMS control implementations described above applies equally well to the case where the conventional counter is replaced by a LFSR-type counter. The split of the LFSR into two smaller LFSR-based counters, \( C_1 \) and \( C_2 \), will not affect the randomness of the input pattern generator if concatenable polynomials are used [4].

### 6 Fault Simulation Experiments

To study the fault coverage performance of the FMS scheme, exact fault simulation of the ISCAS'85 benchmark circuits [5] were performed with output data compaction. In the experiments, we used a multiple input nonfeedback shift register as a space compactor to convert the multiple bit output data from a CUT into a single bit data to the LFSR signature analyzer [1]. In the experiments, single stuck-at faults were assumed. In the case of the SS scheme, an 8-bit primitive LFSR was used as the data compactor. For the FMS scheme, 8 signatures were checked at equidistant check points, and each was generated by the 8-bit LFSR, i.e., \( k = 8 \) and \( n = 8 \). In terms of fault simulation time reduction, the equidistant check point scheduling is the worst of the control strategies described in section 5.2. But the area requirement for the control of the FMS scheme in this case is the same as that required for a SS scheme. The total silicon area of the FMS scheme in the experiments is thus about equivalent to that of a 10.16-bit LFSR, as opposed to an 8-bit LFSR for the SS scheme.

Table 1 shows the exact fault coverage before signature analysis (the column referred as "NC"), and the coverage with the SS and the FMS schemes, respectively. From Table 1, in all cases, the coverage of the FMS scheme is
either the same or very close to that of the no-compaction case, and higher than that of the SS scheme.

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>No. Faults</th>
<th>Fault Coverage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C439</td>
<td>562</td>
<td>99.244</td>
</tr>
<tr>
<td>C499</td>
<td>732</td>
<td>99.244</td>
</tr>
<tr>
<td>C5315</td>
<td>1171</td>
<td>99.244</td>
</tr>
<tr>
<td>C5315</td>
<td>1820</td>
<td>99.244</td>
</tr>
<tr>
<td>C5906</td>
<td>2015</td>
<td>94.652</td>
</tr>
<tr>
<td>C2670</td>
<td>2781</td>
<td>82.659</td>
</tr>
<tr>
<td>C3540</td>
<td>4035</td>
<td>93.451</td>
</tr>
<tr>
<td>C5915</td>
<td>5982</td>
<td>96.230</td>
</tr>
<tr>
<td>C7552</td>
<td>8408</td>
<td>93.010</td>
</tr>
</tbody>
</table>

Table 1. Fault coverage enhancements (average of 4 trials).

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>No. Faults</th>
<th>Fault Simulation Time (sec.)</th>
<th>Time Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>C439</td>
<td>562</td>
<td>2.25</td>
<td>48.90</td>
</tr>
<tr>
<td>C499</td>
<td>732</td>
<td>3.70</td>
<td>72.64</td>
</tr>
<tr>
<td>C5315</td>
<td>1171</td>
<td>14.30</td>
<td>191.37</td>
</tr>
<tr>
<td>C5315</td>
<td>1820</td>
<td>29.30</td>
<td>386.36</td>
</tr>
<tr>
<td>C5906</td>
<td>2015</td>
<td>65.66</td>
<td>493.26</td>
</tr>
<tr>
<td>C2670</td>
<td>2781</td>
<td>209.56</td>
<td>1244.56</td>
</tr>
<tr>
<td>C3540</td>
<td>4035</td>
<td>268.21</td>
<td>1823.64</td>
</tr>
<tr>
<td>C5915</td>
<td>5982</td>
<td>419.86</td>
<td>4992.12</td>
</tr>
<tr>
<td>C7552</td>
<td>8408</td>
<td>1065.72</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Table 2. Fault simulation time reductions (average of 4 trials).

Table 2 shows the CPU time to perform the above fault simulations. From Table 2, the CPU time required to perform the fault simulations with the CMS scheme is always one order of magnitude longer. In Table 2, the column Time Savings shows the CPU time saved from using the CMS scheme rather than the SS scheme. Here, the number of test vectors simulated is small, and the check point scheduling is the worst case in reducing fault simulation time. If longer test length, larger circuits, and optimal scheduling of the check points were considered, the fault simulation time saved from using the CMS scheme rather than the SS scheme would be even much more significant.

6 Conclusions

In conventional multiple signature (CMS) schemes, for a CUT to be declared good, the signatures and the references must correspond on a one-to-one basis. That is, at each check point, the signature obtained must match the specific reference for that check point. This requirement of the one-to-one signature – reference correspondence makes the CMS scheme complex and expensive to implement in terms of silicon area. In this paper, we proposed the Fuzzy Multiple Signature (FMS) scheme in which the aforementioned requirement is removed. The FMS scheme is very simple, thus much easier and less expensive to implement in a BIST environment. A model for predicting the FSM scheme’s aliasing was developed. Compared with the CMS schemes and single signature (SS) schemes, the FMS scheme requires less silicon area to achieve a given aliasing probability. We reported experimental results on fault coverage enhancement, fault simulation time savings, and silicon area overhead. We have also shown that the complexity for the control of the FMS scheme can be made similar to that of a commonly used SS BIST scheme. In the paper, a LFSR-based data compaction was assumed. However, the FMS scheme can be applied to Multiple Input Shift Register (MISR), Cellular Automata (CA), or count-based data compactors.

7 References