Using Algebraic Signatures to Compress Built-In Self Test on a Chip

Jaya Jeswani*, John Rose, SJ*, Thomas Schwarz, SJ†
*Xavier Institute of Engineering, Mahim Causeway, Mahim, Mumbai, India
jayajeswani21@gmail.com, johnrose@xavierengg.com
†Marquette University, Milwaukee, WI, thomas.schwarz@marquette.edu

***Abstract*—Chip functionality testing can greatly benefit from a Built In Self-Test (BIST). The Self-Test Using MISR and Parallel Shift Register Sequence Generator (STUMPS) architecture uses a compression technique to generate a set of test patterns, to submit them to the circuit undergoing testing, and to compare the output with that of a "gold" (known to be good circuit) by loading and comparing the contents of a Multiple Input Shift Register (MISR). We propose to use algebraic signatures as the comparison signature implemented by the MISR. As we will see, the MISR is still basically a Linear Feedback Shift Register (LFSR), but can now be made to guarantee to discover one or up to k output discrepancies, where k is a very small number that determines the length of the MISR register. The construction of the algebraic signature register is generic and only the comparison value needs to be programmed.**

***Index Terms*—BIST, MISR, Algebraic Signatures**

I. Introduction

Since using outside equipment for testing VLSI circuits is prohibitively expensive, modern circuits are usually tested using a Built-In Self Test (BIST). The Self-Test Using MISR and Parallel Shift Register Sequence Generator (STUMPS) architecture consists of a pseudo-random number generator that generates the test pattern, usually implemented as a Linear Feed-back Shift Register (LFSR), a module to calculate a signature of the output sequence, and testing logic [1], [4].

A number of techniques have been proposed to analyze the output of the Circuit Under Test (CUT). The naïve method of comparing each individual CUT output with a desired value is usually impractical, as it involves a large ROM table of required outputs. Instead, we usually compress both horizontally and vertically. (Actually, the term compaction is preferred since compression suggests no loss of information [11].) We sometimes compact horizontally by using a concentrator-circuit to reduce n outputs into m with $m < n$. We compact vertically by calculating a value from the various outputs of the CUT generated by the various test patterns. Common techniques are counting (the number of zeroes, ones, or transitions), accumulators (where all outputs are treated as binary magnitudes and added), parity checks, and most importantly signature analysis. In signature analysis, the (possibly compacted output) of the CUT is gated at each cycle into an LFSR, which maintains a *signature*. All methods of (vertical) compaction accrue loss of information that can result in a false positive. This is often referred to as *fault masking* or *signature aliasing* [2], [12].

We propose to use algebraic signatures [10] to control false positives. Algebraic signatures are defined over small, binary Galois fields $\mathbb{GF}(2^l)$, whose elements are bit strings of length l. While current signature schemes already use Galois field operations when compacting the different output values, an algebraic signature consists of k different Galois field elements. They have the property that if the stream of CUT output values to be compacted contains less than k errors, then no fault masking is possible. We expect a faulty CUT to be either so bad that the result of a test appears to be essentially random or only shows erroneous output for a few test patterns. The guarantee offered by the use of an algebraic signature is very useful in the latter case, but using them does not put us in worse shape in the first case. The calculation of the algebraic signature is done by k internal LFSR-based Multiple Input Signature Registers (MISRs), one of which can be (but does not have to be) a simple bit-wise parity calculator.

In the remainder of the paper, we discuss in Section II the basic BIST STUMPS architecture and then define and implement algebraic signatures in Section III. In Section IV we prove the key-property of algebraic signatures in order to keep this contribution self-contained. Section V calculates the masking probability in the important case of checking the contents of a large ROM. We then conclude.

II. BIST Architecture review

Testing Integrated Chips (ICs) designed by VLSI (Very Large Scale Integration) technology is a complex and time consuming task as circuits are huge. Built-In Self Test (BIST) replaces the use of outside test equipment and results in overall cost savings. We present the basic BIST layout (after [12]) in Fig. 1. To initiate the self-test, the BIST test controller initializes the Test Pattern Generator (TPG). Traditional BIST uses pseudo-random patterns to generate a series of test patterns. But this approach cannot be very accurate. In order to improve fault tolerance, deterministic test pattern generators such as hybrid BIST [5], reseeding [6], combinatorial Automatic Test Pattern Generators (ATPG) [7], sequential ATPG [3], and compressed ATPG [13] among other were introduced.

The TPG then drives the Circuit under Test (CUT). The Output Response Analyzer (ORA) analyzes the output to reach a verdict on whether the CUT passes or fails the test. The naïve method is to compare the outputs with the golden values (i.e. the outputs of a good circuit) stored in a ROM. However, this

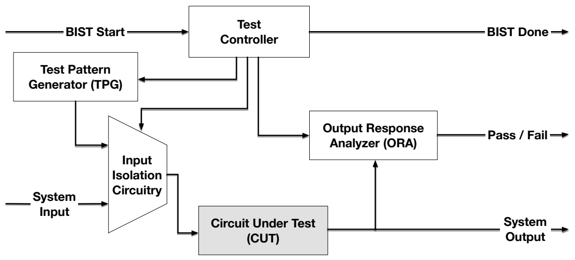

Fig. 1. Basic BIST architecture after [12].

method contributes exceedingly to the overhead of the BIST circuitry. The techniques to *compact* to reach the verdict are according to Stroud [12]:

- Comparison based ORA: The CUT output stream is compared with data in a ROM. The major drawback of this method is the need for a sizable ROM.
- Counting techniques: The ORA counts the number of zeroes (or ones) in the output stream, usually for each output bit separately.
- Accumulators: The CUT output is treated as a binary number, the elements of the output stream are added up, and the sum is compared to the result of a "gold standard" (functioning) circuit.
- Parity checks: The bit-wise parity of the CUT outputs is calculated and compared to that produced by a gold CUT.
- Signature based schemes: A generalization of accumulators, the CUT outputs are algebraically manipulated to generate a signature, which then is compared to the signature resulting from testing a gold CUT.

These schemes can be augmented with other mechanisms, such as concentrators, which are small circuits that process several bits of CUT output into a single bit. For instance, we can calculate the exclusive-OR of several output bits.

Signature based schemes frequently use a Multiple Input Signature Register (MISR) with an internal feedback Linear Feedback Shift Register (LFSR). We give a small example in Fig. 2. At each clock cycle, the four bit input becomes available and is directed to an exclusive-OR gate, to which the output of the previous flip-flop is the other input. The output of the left-most flip-flop is directed to the exclusive-OR gate in front of the rightmost flip-flop. This output is also added to the input to the second flip-flop. This MISR design shifts the previous contents to the right. If there is an overflow in the rightmost flip-flop, then it exclusively-ORs the one to the leftmost and second leftmost flip-flop. As is well known, we can represent the MISR operations in terms of multiplying by t in the ring of polynomials over $\{0,1\}$, i.e. in $\{0,1\}[t]$ modulo a generator polynomial, which would be t^4+t+1 in our case. This polynomial indicates that the rightmost output should be connected to the first and the second flip-flop on the left.

It turns out that a LFSR with maximum cycle when all inputs are zero with the exception of the first one are given

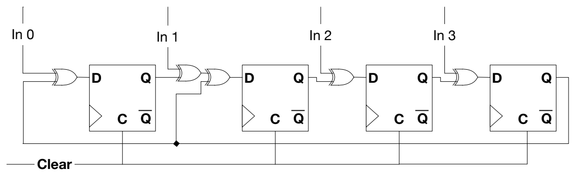

Fig. 2. Four-bit MISR.

by an irreducible polynomial. In this case, the LSFR produces what we call the 1-component algebraic signature of the input stream based on Galois field calculations. BIST architectures can be improved by lowering power consumption and gate count, e.g. [8].

III. Calculating Algebraic Signatures in Hardware

Algebraic signatures are hashes (signatures) with algebraic properties [10].

A. Galois field $\mathbb{GF}(2^n)$ implementation

Algebraic signatures are defined over a Galois field $\mathbb{GF}(2^n)$. The elements of said Galois field are bit vectors $(a_{n-1}, a_{n-2}, \dots, a_2, a_1, a_0)$, which are mathematically identified with a polynomial $a_{n-1}t^{n-1} + a_{n-2}t^{n-2} + \ldots + a_2t^2 + a_1t + a_0 \in \{0,1\}[t]$ with binary coefficients of degree n. We add and subtract two such polynomial by taking the exclusive-or of the coefficients, which is just the bit-wise exclusive-or for the corresponding bit vectors. However, we need the representation as polynomials in order to define multiplication, which is via a so-called "generator" polynomial ϕ of degree $n+1$. A generator polynomial is any irreducible polynomial of degree $n+1$. While switching generator polynomials results in different signatures, mathematically, the underlying fields are isomorphic. The product of two polynomials of degree less than n in $\{0,1\}[t]$ is their product modulo ϕ. In addition, we want ϕ to be such that for $\alpha = 000\ldots010 \approx t$ we have

$$\{\alpha^i | 0 \leq i \leq 2^n - 1\} = \mathbb{GF}(2^n)^*.$$

In other words, ϕ is such that all non-zero elements of $\mathbb{GF}(2^n)$ can be represented as a power of α. The mathematical theory of finite fields is very rich [9]. All finite fields with the same number of elements are mathematically isomorphic, i.e. have the same structure. For any prime power, there exists one (and mathematically only one) Galois field with this number of elements. There is always a ϕ with the stated property.

B. Definition of Algebraic Signatures

An algebraic signature of a finite sequence $B = (\beta_i)_{i=0,1,\dots,m}$ with respect to a non-zero element $\gamma \in \mathbb{GF}(2^n)$ is defined as

$$\mathrm{sig}_\gamma(B) = \sum_{\nu=0}^{m} \beta_i \gamma^i.$$

The N-fold composite algebraic signature is a vector

$$\mathrm{sig}_{\gamma,N} = \big(\mathrm{sig}_{\gamma^0}(B), \mathrm{sig}_{\gamma^1}(B), \mathrm{sig}_{\gamma^2}(B), \dots \mathrm{sig}_{\gamma^{N-1}}(B)\big)$$

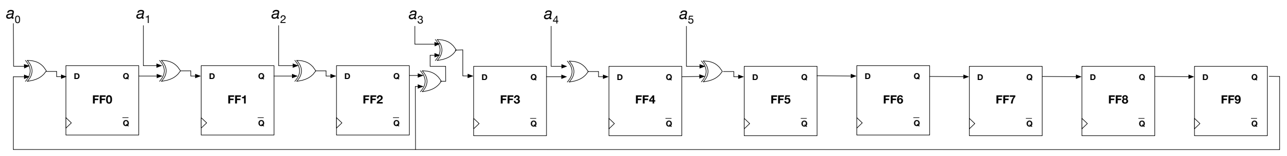

Fig. 3. Implementation of a 10-bit MISR for the α-signature of a 6-bit input stream.

Vandermonde's determinant formula implies that the N-fold composite algebraic signature changes for sure if we change up to N elements in the sequence [10].

We use $\alpha = 00\ldots010$ as the defining element in the signature calculation, i.e., we calculate $\mathrm{sig}_\alpha(B)$. Multiplication by α corresponds to multiplication by t for polynomials, which corresponds to shifting left by one for a bit-vector. The result needs to be taken modulo ϕ. If the coefficient of t^n is one after multiplication, taking modulo ϕ is implemented by exclusive-or-ing with ϕ, otherwise the result of the shift remains unchanged. In Python used as pseudo-code, we have

```
def mult_alpha(mu):
   mu = mu << 1
   if mu & (1 << n):
      mu ^= phi
   return mu
```

We present a schematics of the multiplier in Fig. 3.

The calculation of the α signature can be done using Horner's method

$$\mathrm{sig}_\alpha(B) = (\ldots(((\beta_0\alpha + \beta_1)\alpha + \beta_2)\alpha + \beta_3)\ldots)$$

which in pseudo-code is

```
def sig_alpha(B):
   sig = B[0]
   for i in range(1, m+1):
      sig = mult_alpha(sig) ^ B[i]
   return sig
```

C. Example

We present the implementation of a 10-bit MISR that calculates the algebraic signature of a 6-bit input stream $A = (a_0, a_1, a_2, a_3, a_4, a_5)$ in Fig. 3. The contents of the MISR are maintained in the flip-flops FF0 to FF9. Originally, the contents of the registers are all zero.

We need to implement multiplication in $\mathbb{GF}(2^{10})$, which is done by selecting a primitive polynomial ϕ of degree 10 with coefficients in $\{0,1\}$. Since we will XOR with ϕ, it behoves us to select a primitive polynomial with the least possible number of non-zero coefficients. It turns out that polynomials with two non-zero coefficients are not irreducible, but often trinomials will do. We can use the table by Zierler and Brillhart [15] to find that $\phi(t) = t^{10} + t^3 + 1$ is a primitive polynomial of degree 10 over $\{0,1\}$. A small Python script that generates all powers of t modulo $\phi(t)$ confirms this fact.

With each clock cycle, new inputs A become available. Each clock cycle also moves the contents of the previous flip-flop into the successive one. The inputs and the output of the previous flip-flop are eXclusive-OR-ed (XOR-ed). If flip-flop FF9 in Fig. 3 contains a one value, then we also need to XOR with $\phi(t)$. This is easily done. The leading coefficient and the one stored in FF9 are XOR-ed, giving us zero. We can therefore safely ignore this value. The constant 1 and t^3 are XOR-ed to the contents of the shift register. This is simply achieved by XOR-ing the output of FF9 with the a_0 and with a_3 together with the output of FF2 respectively.

Notice that the design would not change if the input would be a 10-bit value.

D. Zero component signatures

The calculation of the $\alpha^0 = 1$ signature is of course much easier, since $\mathrm{sig}_1(A)$ is just the exclusive-or of the components of A. Consequentially, this part of the algebraic signature MISR only needs to have as many flip-flops as there are bits in A. Fig. 4 gives the simple design.

E. Second component signature

For the second component, at each clock cycle we shift the contents of the MISR by two. The overflow after shifting is now two-bits long. We again use ten flip-flops to store the contents of the MISR, but now need to process overflow if FF8 *or* FF9 are one. If FF8 is one and FF9 is zero, then we need to add $\phi(t)$ to the MISR contents, if FF8 is zero and FF9 is zero, then we need to add $t \cdot \phi(t)$ to the MISR contents, and if both FF8 and FF9 are one, then we need to add $\phi(t) + t \cdot phi(t)$ to the MISR contents. Of course, if both of them are zero, then there is no overflow and we need no additional processing.

For example, assume that the MISR contains currently the values $m_0 = 1$, $m_1 = 0$, $m_2 = 0$, $m_3 = 1$, $m_4 = 0$, $m_5 = 0$, $m_6 = 1$, $m_7 = 1$, $m_8 = 1$, and $m_9 = 1$. In vector form, *starting* with the constant coefficient, the MISR contents are (1001001111). Shifting by two positions to the right gives (001001001111), which is remedied by adding (XOR-ing) the primitive polynomial ϕ, corresponding to 10010000001 and by adding $t \cdot \phi(t)$, corresponding to 010010000001 The result is (11111001100), but the two rightmost bits are always guaranteed to be zero, so we do not need to store them, leaving us with the ten bits (111110011).

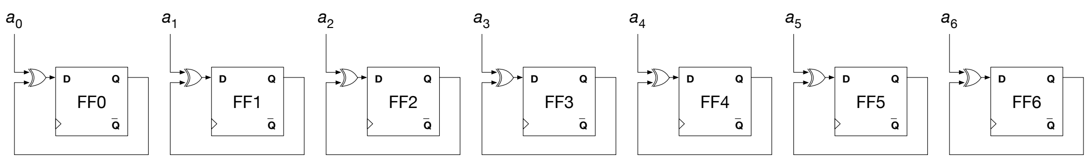

Fig. 4. Implementation of a 6-bit register for the $\alpha^0 = 1$ signature of a 6-bit input stream.

In the hardware implementation, we need to XOR the current contents of FF8 with the inputs to FF0 and FF3 and to XOR the current contents of FF9 with the inputs to FF1 and FF4. Note that our trinomial has the nice property that we do not need to XOR two outputs to a single flip-flop, but this would be no hindrance.

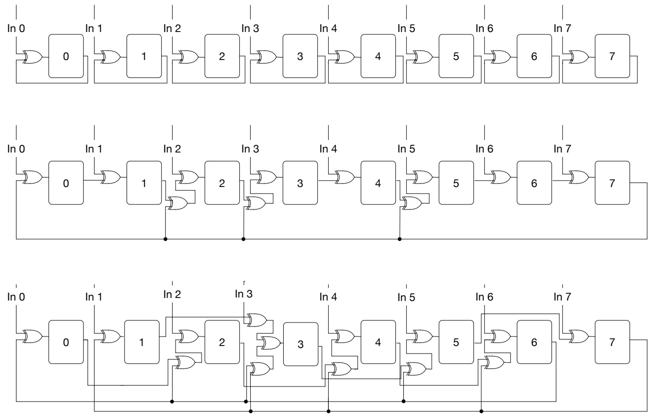

Fig. 5. 8-bit MISRs for the α^0-, α- and α^2 signature on top of $\mathbb{GF}(2^8)$. The rectangular boxes denote flip-flops.

F. A more complicated example

The Galois field $\mathbb{GF}(2^8)$ has no trinomial generator, but we can use $\phi(t) = t^8 + t^5 + t^3 + t^2 + 1$ as a primitive generator polynomial. We are implementing 8-bit MISRs for the zero component (α^0-signature), the first component (α-signature) and the second component (α^2-signature. The zero-component signature is the exclusive-or of the input lines. In consequence, we store each coordinate in its own flip-flop and whenever new input is available, we replace its contents with the exclusive-or of the old contents and the input. Fig. top shows the implementation minus the timing hardware.

The MISR for the α-signature is also straightforward. The signature is accumulated in eight flip-flops. Whenever we process new input, we shift the contents of the flip-flop to the right after an exclusive-or with the new input. Fig. 5 middle shows the result. Since we follow the convention that we shift to the right, the lowest coordinate is kept in the leftmost flip-flop. Whenever we have a one in the right-most flip-flop (corresponding to input line 7), we add (via an exclusive-or) the generator polynomial minus the leading component. Since the generator polynomial is $\phi(t) = t^8 + t^5 + t^3 + t^2 + 1$, we lay an exclusive-or of the output of the last flip-flop, namely 7, to the flip-flops 1, 2, 3, and 5. Since we use the Mathematical convention of starting polynomials with the highest power of t, the order is reversed. We need one exclusive-or gate for all input lines and one for each monomial in the generator polynomial with the exception of the constant and the leading monomial. This is the number of ones in the binary representation minus two. In our case, we have $8 + 3$ exclusive-or gates.

The MISR for the α^2-signature is slightly more complicated. As explained before, we shift the contents of the eight flip-flops by two positions to the right. If the second-last flip-flop has a current state of one, then we add $t^5 + t^3 + t^2 + 1$ to the current state, if the last flip-flop has current state of one, then we add $t \cdot (t^5 + t^3 + t^2 + 1) = t^6 + t^4 + t^3 + t$ to the contents. We can express this algebraically. We denote with f_i^{in} the input to flip-flop i and with f_i^{out} its output. We call in_i the input bit i. Then we have

$$\begin{aligned}
f_0^{\text{in}} &= f_6^{\text{out}} \oplus \text{in}_0 \\
f_1^{\text{in}} &= f_7^{\text{out}} \oplus \text{in}_1 \\
f_2^{\text{in}} &= f_0^{\text{out}} \oplus f_6^{\text{out}} \oplus \text{in}_2 \\
f_3^{\text{in}} &= f_1^{\text{out}} \oplus f_6^{\text{out}} \oplus f_7^{\text{out}} \oplus \text{in}_3 \\
f_4^{\text{in}} &= f_2^{\text{out}} \oplus f_7^{\text{out}} \oplus \text{in}_4 \\
f_5^{\text{in}} &= f_3^{\text{out}} \oplus f_6^{\text{out}} \oplus \text{in}_5 \\
f_6^{\text{in}} &= f_4^{\text{out}} \oplus f_7^{\text{out}} \oplus \text{in}_6 \\
f_7^{\text{in}} &= f_5^{\text{out}} \oplus \text{in}_7.
\end{aligned}$$

The number of times that we guide f_6^{out} and f_7^{out} to another flip-flop is each equal to the number of monomials in the generator polynomial minus one. We connect the output of all other flip-flops to another flip-flop as many times as there are flip-flops minus two. We also connect each input-line to its flip-flop. In our example, we need $2\times 4+8-2 = 14$ exclusive-or gates. In general, if we have an b-bit MISR and r non-zero monomials in the generator polynomial excluding the leading one, then the number of exclusive-or gates is $2\times(r-1)+b$. We show an implementation of the second component signature at the bottom of Fig. 5.

IV. Error Detection with Algebraic Signatures

Assume an k-component α-signature using a Galois field $\mathbb{GF}(2^l)$ that therefore consists of lr bits. We assume that 2^l is larger than the number n of test pattern generated. Assume that the output sequence $\left(s_i\right)_{0\leq i<n}$ contains up to k values different from the "golden" values $\left(g_i\right)_{0\leq i<n}$. Let i_1, i_2, …, i_n be the indices where there is a difference. This gives the following system of equations, when we recall that the difference operation is the same as the addition operation in $\mathbb{GF}(2^l)$.

$$\begin{gathered}
(s_{i_1}+g_{i_1})+(s_{i_2}+g_{i_2})+\ldots+(s_{i_k}+g_{i_k})=0\\
\alpha^{i_1}(s_{i_1}+g_{i_1})+\alpha^{i_2}(s_{i_2}+g_{i_2})+\ldots\\
\cdots+\alpha^{i_1}(s_{i_k}+g_{i_k})=0\\
\alpha^{2i_1}(s_{i_1}+g_{i_1})+\alpha^{2i_2}(s_{i_2}+g_{i_2})+\ldots\\
\ldots+\alpha^{2i_1}(s_{i_k}+g_{i_k})=0\\
\vdots=0\\
\alpha^{(k-1)i_1}(s_{i_1}+g_{i_1})+\alpha^{(k-1i_2}(s_{i_2}+g_{i_2})+\ldots\\
\ldots+\alpha^{(k-1i_1}(s_{i_k}+g_{i_k})=0
\end{gathered}$$

The coefficient matrix of this system is a Vandermonde matrix and therefore invertible. Hence, the differences $s_{i_\nu}+g_{i_\nu}$ have to be zero. Therefore, in fact, the output sequences cannot be different from the gold values if the k-component algebraic signature is not different from its gold value.

V. Analysis of Output Response Analyzers on ROM contents

We evaluate the various ORA possibilities for a small ROM consisting of 1024 = 2^{10} words of 8 bit each. We compare MISRs build on Galois fields with 2^8, 2^{10}, 2^{11} and 2^{12} elements and generator polynomials in hexadecimal form of `0x12d`, `0x803`, `0x1003`, and `0x2009` respectively. We compared the error detection capability of four configurations, a single α-signature, a double signature with Exclusive-Or and the α-signature, an alternative double signature with the α- and α^2-signatures, and a triple signature. If we use our methods on a ROM with l lines and size b, then the normal MISR would use b bits, giving a compression rate of $1 : l$. If we use triple algebraic signatures and MISRs with c bits, then the compression rate goes down to $3c : lb$.

We simulated many runs of one million error sets with a given number b of bit errors. We did not assume that all b errors were in different words, only that the total number of flipped bits is b. For each error set, we calculated the various signatures and determined whether the signatures had changed and the errors would have been detected. Table I gives the number of times (per million) that a chip passes the test while being faulty, together with 95% confidence intervals.

All signatures detect if exactly one bit of the ROM contents is in error. The double signatures built on top of $\mathbb{GF}(2^8)$ (abbreviated as GF8 in Table I) sometimes fail to detect a double bit flip, this is completely attributable to having the same error occur in words of distance 511 or a multiple thereof apart. Similarily, the double signatures on top of $\mathbb{GF}(2^{10})$ fail to detect two flipped bits for sure. This is because the antilog table rolls over after $2^{10}-1$ increments and there are 2^{10} ROM words in error. While the mathematical properties are the same for a double signature consisting of the exclusive-or and the α-signature as for a double signature consisting of the α- and the α^2-signature, their behavior is not. We also observe that the capability of error detection depends on the even-ness of b.

For implementation, we would use Field-Programmable Gate Arrays (FPGA). We use a device `xc5vlx20t-2ff323` belonging to the Virtex5 family of XILINX. Each EXclusive OR (EXOR) requires four gates and each Flip-Flop (FF) requires eight gates, [14]. For example, the circuit in Fig. 2 has five EXOR gates and four FFs and therefore needs 52 gates. Using these numbers, we obtain the gates counts for our various ORA configuration presented in Table II. These gate counts do not contain the overhead imposed by timing and synchronization.

Selecting the optimal configuration supposes an approximate knowledge of the probability of programming errors. If we deem that two flipped bits has too low a probability to be taken into account, then a single 8b-MISR implementing a zero-component is sufficient. If this is not the case, then under most circumstances, the double configurations would be sufficient, under the proviso that the number of non-zero elements in the Galois field is equal or larger than the number of words in the ROM. We would thus pick the GF11 double or the GF11 double-alternative configuration, preferring the latter because the small increase in the number of gates (272 versus 288) is dwarfed by the overall better error detection capabilities. However, a strong case can also be made for the larger GF12-based ORAs.

Finally, if the probability of bit-flips in the programming of the ROM is large or the costs of a faulty device that passes the self-test is high, then we would need to invest in an additional signature component.

VI. Conclusion

We have proposed a method to lower the probability of error masking in Output Response Analyzers in case of few errors and evaluated it in terms of probability of detecting b flipped bits and gate count. Our ORA are guaranteed to find 1, 2, 3 and by extension larger numbers of flipped bits in the test output. They also do well for larger numbers of errors.

An advantage of our proposal is its generic nature. While test pattern generators need to be tailored to the circuit, our MISR organization is independent of the CUT. Only its dimension and of course the "golden value" vary between circuits. Given that design costs are an important part of total chip costs, this is an indisputable advantage.

TABLE I
PROBABILITY OF FAILING TO DETECT AN ERRONEOUS ROM. THE NUMBERS GIVE THE AVERAGE NUMBER OF FAILURES TO DETECT AN ERROR PATTERN BY SIGNATURES PER ONE MILLION CASES AND THE 95% CONFIDENCE INTERVAL GIVEN BY ITS DEVIATION FROM THE CENTER POINT.

	Errors per Million								
ROM Err	2	3	4	5	6	7	8	9	10
GF8 single	3812.76 ±12.35	3910.56 ±11.06	3902.31 ±12.70	3903.51 ±13.08	3896.99 ±10.54	3909.72 ±11.87	3907.38 ±13.99	3900.35 ±13.08	3907.07 ±11.82
GF8 double	371.12 ±4.17	0 ± 0	165.64 ±2.49	0 ± 0	88.45 ±9.08	0 ± 0	58.04 ±1.39	0 ±0	43.44 ±1.30
GF8 double alt.	369.88 ±3.101	71.81 ±1.34	21.83 ±0.88	15.66 ±0.74	14.85 ±0.79	15.42 ±0.74	15.19 ±0.78	15.54 ±0.72	14.96 ±0.76
GF8 triple	0 ± 0	8.13±0.51	0 ± 0	0.41±0.13	0 ± 0	0.2 ± 0.09	0 ± 0	0.19±0.08	0.17±0.08
GF10 single	849.26 ±5.47	431.38 ±4.16	504.91 ± 4.34	485.48 ±4.31	492.38 ±4.52	488.82 ±4.77	485.76 ±4.56	488.94 ±3.95	490.88 ±4.27
GF10 double	0.22 ±0.10	0 ± 0	20.79± 0.89	0 ± 0	11.13 ±0.63	0 ± 0	7.12 ±0.49	0 ± 0	5.55 ±0.44
GF10 double alt.	0.24±0.08	2.18±0.30	0.55±0.14	0.23±0.08	0.23±0.10	0.17±0.07	0.3 ± 0.10	0.32±0.12	0.24±0.10
GF10 triple	0.28±0.11	0 ± 0	0.32±0.13	0 ± 0	0 ± 0	0 ± 0	0.02±0.03	0 ± 0	0 ± 0
GF11 single	853.57 ±5.7	248.2 ±3.48	243.86 ±3.11	245.22 ±2.64	245.13 ±3.13	244.07 ± 2.83	245.98 ±2.87	246.05 ±3.04	243.22 ±3.28
GF11 double	0 ± 0	0 ± 0	10.3±0.64	0 ± 0	5.51±0.51	0 ± 0	3.73±0.37	0 ± 0	2.65±0.32
GF11 double alt.	0 ± 0	4.31±0.39	0.48±0.13	0.09±0.06	0.08±0.05	0.08±0.05	0.07±0.05	0.06±0.05	0.05±0.04
GF11 triple	0 ± 0	0 ± 0	0.46±0.13	0 ± 0	0.02±0.03	0 ± 0	0 ± 0	0 ± 0	0 ± 0
GF12 single	852.11 ±5.48	120.71 ±2.03	122.45 ±2.19	122.31 ±2.07	123.77 ±1.92	123.66 ±1.96	120.6 ±2.20	122.22 ±2.45	121.34 ±2.31
GF12 double	0 ± 0	0 ± 0	5.34 ± 0.4	0 ± 0	2.78±0.29	0 ± 0	1.87±0.29	0 ± 0	1.4 ± 0.22
GF12 double alt.	0 ± 0	1.78±0.27	0.24±0.01	0.05±0.04	0 ± 0	0 ± 0	0.04±0.04	0 ± 0	0 ± 0
GF12 triple	0 ± 0	0 ± 0	0.32±0.12	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0

TABLE II
GATE COUNTS FOR THE VARIOUS ORA IMPLEMENTATIONS CONSIDERED.

MISR type	bits	EXORs	FFs	Gates
GF8 single	8	14	8	120
GF8 double	16	14	16	184
GF8 double alt	16	25	16	228
GF8 triple	24	33	16	260
GF10 single	10	12	10	128
GF10 double	20	23	20	252
GF10 double alt	20	24	20	256
GF10 triple	30	62	30	488
GF11 single	11	13	11	140
GF11 double	22	24	22	272
GF11 double alt	22	28	22	288
GF11 triple	33	39	33	420
GF12 single	12	14	12	152
GF12 double	24	26	24	296
GF12 double alt	24	30	24	312
GF12 triple	36	42	36	456

We did not evaluate error detection capabilities in dependence of the generator polynomial of the Galois field, but now suspect that there is a dependence. That smaller number of errors sometimes have higher probability of remaining undetected is a phenomenon worth further investigation. Because of time and space constraints, we did not evaluate our proposal against "normal" circuits. One reason is of course that the behavior should depend heavily on the circuit under test and that therefore one has to evaluate many different circuits to start drawing generic conclusions.

REFERENCES

[1] P. H. Bardell, J. Savir, and W. H. McAnney, *Built-in Test for VLSI*. Wiley, 1987.

[2] D. K. Bhavsar and R. W. Heckelman, "Self-testing by polynomial division," *Journal of Digital Systems*, vol. 6, no. 2-3, pp. 139–160, 1982.

[3] H. Cho, G. D. Hachtel, and F. Somenzi, "Fast sequential ATPG based on implicit state enumeration," in *Test Conference, 1991, Proceedings., International*. IEEE, 1991, p. 67.

[4] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for large industrial designs: real issues and case studies," in *Proceedings of the IEEE International Test Conference*, 1999, pp. 358–367.

[5] G. Jervan, P. Eles, Z. Peng, R. Ubar, and M. Jenihhin, "Hybrid BIST time minimization for core-based systems with STUMPS architecture," in *Proceedings, 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems,*. IEEE, 2003, pp. 225–232.

[6] E. Kalligeros, X. Kavousianos, and D. Nikolos, "Multiphase BIST: a new reseeding technique for high test-data compression," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 10, pp. 1429–1446, 2004.

[7] Y. C. Kim, V. D. Agrawal, and K. K. Saluja, "Combinational test generation for acyclic sequential circuits using a balanced ATPG model," in *Fourteenth International Conference on VLSI Design, 2001*. IEEE, 2001, pp. 143–148.

[8] N. R. Kiran, G. Harish, A. Karthik, and S. Yellampalli, "Low power and hardware cost STUMPS BIST," in *9th International Symposium on VLSI Design and Test (VDAT)*, vol. 1. IEEE, 2015, pp. 1–4.

[9] R. Lidl and H. Niederreiter, *Finite fields*. Cambridge University Press, 2008.

[10] W. Litwin and T. Schwarz, "Algebraic signatures for scalable distributed data structures," in *Proceedings, 20th International Conference on Data Engineering*. IEEE, 2004, pp. 412–423.

[11] E. J. McCluskey, "Built-in self-test techniques," *IEEE Design & Test of Computers*, vol. 2, no. 2, pp. 21–28, 1985.

[12] C. E. Stroud, *A Designer's Guide to Built-in Self-test*. Kluwer Academic Publishers, 2002, vol. 19.

[13] P. Wohl, J. A. Waicukauski, S. Patel, and G. Maston, "Effective diagnostics through interval unloads in a BIST environment," in *Proceedings of the 39th Annual Design Automation Conference (DAC)*, 2002.

[14] XILINX, "Gate count capacity metrics for FGGAs; XAPP059, version 1.1, 1997," [Online], `www.xilinx.com/support/documentation/application_notes/xapp059.pdf`

[15] N. Zierler and J. Brillhart, "On primitive trinomials (mod 2)," *Information and Control*, vol. 13, no. 6, pp. 541–554, 1968.