Designing of an Algebraic Signature Analyzer for Mixed-Signal Systems

Vadim Geurkov and Lev Kirischian
Email: {vgeurkov, lkirisch}@ee.ryerson.ca
Department of Electrical and Computer Engineering
Ryerson University, Toronto, ON, Canada

***Abstract*—In this paper, we propose a novel approach to designing an algebraic signature analyzer that can be employed for mixed-signal systems testing. Due to its algebraic nature, the analyzer does not contain carry propagating circuitry. This helps to improve its error immunity, as well as performance. The proposed scheme can also be used in arithmetic/algebraic error-control coding and cryptography.**

I. Introduction

Signature analysis has been widely used for digital and mixed-signal systems testing [1]–[4]. In the built-in implementation of this method, a circuit under test (CUT) is fed by test stimuli while the output responses are compacted by a signature analyzer (SA), as illustrated in Figure 1. A signature of a fault free circuit is referred to as a reference signature. The actual signature is compared against the reference signature and a pass / fail decision is made.

It is assumed that the output responses are digital, even though the CUT may contain analogue circuits. In particular, output responses are *n*-bit vectors (or *q*-ary symbols, where $q = 2^n$). Depending on the nature of the CUT (whether it is digital or mixed-signal object), the output responses constitute *point* values or *interval* values [5]. The (closed) interval $[a, b]$ is defined as follows [6]:

$$[a, b] = \{x \in \mathbf{Z} : a \leq x \leq b\}$$

Here $\mathbf{Z}$ is the set of all integers and *a*, *b* are integers with $a \leq b$.

If the CUT is a digital system, the SA implements a circuit that calculates an *algebraic remainder*. The comparison procedure uses a point value of the reference signature. If the CUT is a mixed-signal system, the SA performs an *arithmetic residue* calculation. In this case the comparison procedure involves an interval value of the reference signature (tolerances) and uses a window comparator.

It is essential that the circuit shown in Figure 1 is synchronous, that is the change of test stimuli / output responses and the shift of the signature analyzer are synchronized with the explicit clock. The clock rate is defined by the propagation delay of the CUT.

Design methods for an algebraic signature analyzer have been well developed in error-control coding [7]. A remainder calculating circuit for an arbitrary base (i.e. binary or non-binary) can be readily designed for a digital CUT of any size. In contrast, a residue calculating circuit is much harder to design, specifically for a non-binary base [8]. Furthermore, due to the presence of carry propagating circuitry, the implementation complexity and error vulnerability of the residue calculating circuit is higher compared to the remainder calculating circuit.

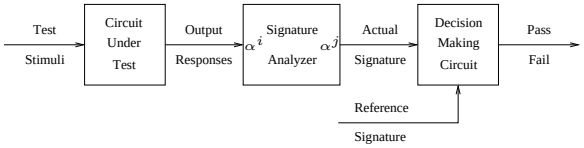

Fig. 1. Built-in signature analysis of a circuit under test

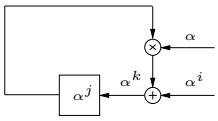

Fig. 2. A symbolic presentation of an algebraic signature analyzer

We propose a novel approach to designing an algebraic signature analyzer for mixed-signal systems testing. Due to an algebraic nature, the analyzer does not contain carry propagating circuitry. This helps to improve its error immunity, as well as performance.

II. Conventional Algebraic Signature Analyzer

An algebraic signature analyzer in a symbolic form is presented in Figure 2 [8]. Here α is a primitive element of a finite field $GF(2^n)$. In particular, α is a root of the generator polynomial $g(x)$ of degree *n* [7].

Without a loss of generality, we will consider a 3-bit signature register ($n = 3$). Therefore, α will be a primitive element of $GF(2^3)$ (e.g. a root of a primitive polynomial $g(x) = x^3 + x + 1$). And a symbolic scheme of Figure 2 will transfer to the logic level scheme of Figure 3 [8].

An operational cycle (a shift) of this SA can be described by the expression:

$$\alpha^j \alpha + \alpha^i = a_j^+ = \alpha^k$$

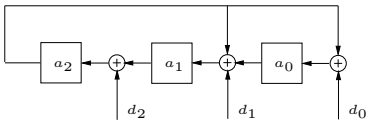

Fig. 3. A logic level presentation of the algebraic 3-input signature analyzer

TABLE I. THREE REPRESENTATIONS FOR THE ELEMENTS OF $GF(2^3)$ GENERATED BY $g(x) = x^3 + x + 1$. HERE $g(\alpha) = 0$.

Power representation	Polynomial representation	Vector representation
0	0	0 0 0
α^0	α^0	0 0 1
α^1	α^1	0 1 0
α^2	α^2	1 0 0
α^3	$\alpha^1 + \alpha^0$	0 1 1
α^4	$\alpha^2 + \alpha^1$	1 1 0
α^5	$\alpha^2 + \alpha^1 + \alpha^0$	1 1 1
α^6	$\alpha^2 + \alpha^0$	1 0 1

Here a_j^+ is the next state of the signature register. Since α is a field element, algebraic operations in the left part of this formula will result in a new field element. Therefore, $a_j^+ = \alpha^k$.

If the preliminary cleared analyzer receives, for example, the following sequence of 3-bit output responses from a digital CUT, $\alpha^5, \alpha^6, \alpha^4, \alpha^2, \alpha^1, \alpha^0$, then after the 6-th shift its content will become:

$$(((((0 \cdot \alpha + \alpha^5)\alpha + \alpha^6)\alpha + \alpha^4)\alpha + \alpha^2)\alpha + \alpha^1)\alpha + \alpha^0 = \alpha$$

The power representation of the field element, α, corresponds to the vector representation, 010, which is the actual signature of the CUT.

The relationship between different representations for the elements of $GF(2^3)$ is given in Table I.

The output responses of a mixed-signal CUT are distorted even in a fault-free circuit. Small permissable variations in the responses cause a significant deviation of the final signature. For example, if in the above sequence of output responses the least significant bit in the first response changes from 1 to 0 (i.e. the vector 111 changes to 110, or power α^5 changes to α^4), then the actual signature will change from 010 to 101 (or from α to α^6 in power form).

Apparently, the conventional SA represented in Figures 2 and 3 can not be employed for mixed-signal circuits testing.

III. NOVEL APPROACH

Small (permissible) deviations in the data feeding a conventional *n*-bit signature analyzer cause the corresponding signatures to scatter through the complete set of 2^n possible signatures. Under these conditions, the decision making circuit in Figure 1 must be able to compare the actual signature with the large set of reference signatures. This increases the complexity of the analyzer. In contrast, an arithmetic residue calculating analyzer (also referred to as an accumulator) does not search through the entire set of reference signatures. Since this set is *contiguous*, the analyzer employs an "interval" (or window) comparator to make a decision. This simplifies the circuitry.

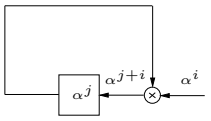

Fig. 4. A symbolic form of an algebraic SA for a mixed-signal CUT

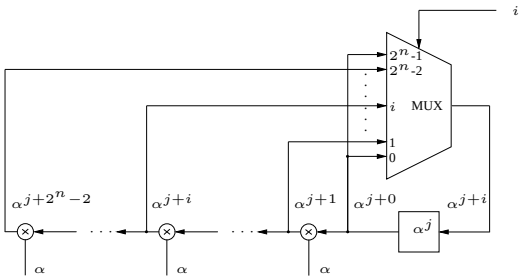

Fig. 5. A more detailed symbolic form of the SA

In the rest of this paper, we will show how to design an algebraic SA, which generates a contiguous set of algebraic reference signatures. In order to solve this task, we will need to order the set of signatures.

A signature can be represented in the vector or power forms. We will use the power exponent as the criterion for ordering the signature set. The distance between two vectors (signatures) will be evaluated as the arithmetic difference between the corresponding exponents. For example, the distance between the signatures 010 and 101 will be 5, because the exponents of powers α^6 and α differ by 5. We can interpret these exponents as output responses of a mixed-signal CUT, since they possess arithmetic properties. At the same time, the corresponding vectors (signatures) possess algebraic properties. Therefore, an arithmetic data is mapped into an algebraic data. Figure 4 represents the circuit which performs this mapping and computes an algebraic signature.

The logic level implementation of the circuit in Figure 4 is more complicated compared to the circuit in Figure 2. Prior to designing the circuit, we have to make a few observations.

The first observation is that

$$\alpha^j \alpha^i = ((\cdots(\alpha^j \underbrace{\alpha)\alpha)\cdots\alpha}_{i}$$

Let us denote an output response from a mixed-signal CUT as *i*. The second observation is that the response i can be considered as an exponent of the power, i.e. α^i. Essentially, this means that the arithmetic values i are mapped into algebraic values α^i.

Based on these observations, we can design a signature analyzer in the way shown in Figure 5. Here α is a primitive element of a finite field $GF(2^n)$; *n* coincides with the bit-length of the output responses. The lower and upper inputs of the multiplexer in Figure 5 are connected together, since $\alpha^{2^n-1} = \alpha^0$ in $GF(2^n)$.

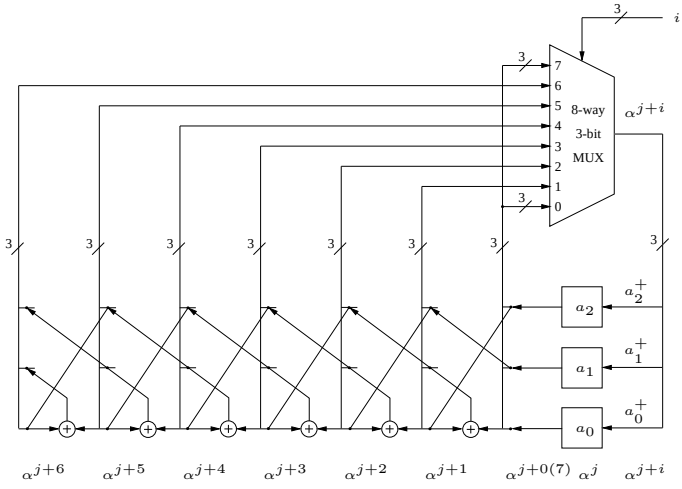

Fig. 6. A register transfer level implementation of the SA

Considering the case when the analyzer is fed by 3-bit data, its more detailed implementation will have the form of Figure 6.

Here the buses consist of 3 lines, as indicated by the appropriate number. The initial content of the SA before the shift is α^j, or $a_2x^2 + a_1x + a_0$ in the polynomial form. The notations a_v and a_v^+, where index v can be one of the 0, 1, 2, indicate the present and next states, respectively.

A multiplier by α in $GF(2^3)$ is realized bearing in mind that $g(x) = x^3 + x + 1$, α corresponds to *x*, and

$$(a_2x^2 + a_1x + a_0)x \bmod g(x) =$$

$$(a_2x^3 + a_1x^2 + a_0x) \bmod g(x) =$$

$$a_2(x + 1) + a_1x^2 + a_0x =$$

$$a_1x^2 + (a_2 + a_0)x + a_2$$

This operation is shown by cross-lines in Figure 6. The multiplexer inputs "0" and "7" are tied together, because $\alpha^7 = \alpha^0$ in the field $GF(2^3)$.

In order to demonstrate how to use this analyzer, we will assume that it receives only two values from a CUT, in particular *j* and *i*. Since the CUT is of a mixed-signal nature, there is an unavoidable (and thereby permitted) deviation of these values by ± 1 (the greater tolerances can also be considered). The analyzer will map the received data into $\alpha^{j\pm1}$ and $\alpha^{i\pm1}$, respectively. If we assume that the initial content of the SA is 001 (versus 000 for a conventional SA), then after the first shift the content becomes $\alpha^0\alpha^{j\pm1} = \alpha^{j\pm1}$. After the second shift, it changes to $\alpha^{j\pm1}\alpha^{i\pm1} = \alpha^{j+i\pm2}$. This expression is derived using the rules of interval arithmetic [6]. It states that for the fault-free CUT the actual result must match one of the values from the interval $[\alpha^{j+i-2}, \alpha^{j+i+2}]$, that is one of the following:

$$\alpha^{j+i-2}, \alpha^{j+i-1}, \alpha^{j+i}, \alpha^{j+i+1}, \alpha^{j+i+2}$$

To further simplify the SA operation, we will assume that instead of α^0 (i.e. 001) the initial SA content is $\alpha^{-(j+i)}$. We will refer to this value as the *seed* value. Then, by the same reasoning, the SA content after two shifts will match one of the following powers:

$$\alpha^{-2}, \alpha^{-1}, \alpha^0, \alpha^1, \alpha^2$$

Due to the closure property of the field $GF(2^3)$, this power set is equivalent to:

$$\alpha^5, \alpha^6, \alpha^0, \alpha^1, \alpha^2 \quad (1)$$

Consequently, the decision making circuit in Figure 2 will work as follows. If the actual signature does not match any values from the set (1), the CUT is considered to be faulty. Since these values are ordered (and surround the power α^0), the decision making circuit can employ a window comparator, thereby reducing the hardware complexity of the SA.

As in any signature analyzer, some errors in the CUT output responses may escape detection. The aliasing rate can be estimated as described in [9] and will coincide with the aliasing rate of the conventional analyzer. If the bit-length of the signature analyzer is *n*, the number of output responses fed into it is *m*, and $mn \gg m + n$ (which normally holds in practice), then the the aliasing rate can be estimated as $\approx 2^{-n}$ [9]. If for a given bit-length, *n*, of the output responses, the aliasing rate, 2^{-n}, is not sufficiently small, then the bit-length of the analyzer can be made greater than *n*.

For example, if $n = 10$ and $m = 300$, then $mn = 3,000$ and $m + n = 310$. Since $3,000 \gg 310$, the aliasing rate can be estimated as $2^{-n} = 2^{-10} = 0.0009766$. If it is desirable to improve this rate and make it even lower, the bit-length of the analyzer can be extended. For example, for a length 12 analyzer (and the same *m*), the rate will become $2^{-12} = 0.0002441$. Note that we have not changed the bit-length of the output responses; it equals to 10. We have only extended the length of the analyzer (from 10 to 12).

Example Let us consider a 3-bit CUT, which is fed by two input stimuli. Under the fault-free operation, the CUT produces the output responses $j = 101 \pm 1$ and $i = 110 \pm 1$. Therefore, the seed value will be $\alpha^{-(j+i)} = \alpha^{-(5+6)} = \alpha^{-11} = \alpha^3$, or 011 in the vector form. If the CUT is fault-free, then after 2 shifts the SA content must match one of the elements in the set (1). For example, if the actual responses are 101+1=110 (or α^6) and 110+1=111 (or α^7) (i.e. the variations are within the tolerance bounds), the signature will be $\alpha^3\alpha^6\alpha^7 = \alpha^2$ which belongs to the set (1). And the decision making circuit will generate a *pass* signal. The validity of such a decision is determined by the aliasing rate.

Let us assume that a fault in the CUT has made the following changes in the output responses: $110 \to 011$ ($\alpha^6 \to \alpha^3$) and $111 \to 100$ ($\alpha^7 \to \alpha^4$). Then the actual signature will become $\alpha^3\alpha^3\alpha^4 = \alpha^3$. This element does not belong to the set (1), so the fault is detected.

There are two distinct ways of designing the decision making circuit depending on the optimization criteria (time or hardware overhead).

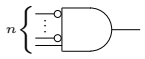

Fig. 7. An *n*-bit comparator

Hardware overhead If performance is paramount and time overhead is not permitted, the following approach can be employed. Let *m* be the number of output responses. All of the 2*m*+1 α-multiplier outputs (see Figure 5) that belong to the set (1), are connected to the first inputs of the 2*m*+1 comparators of a similar type. The second inputs of these comparators are shared and fed by the vector $0\ldots01$. If the CUT is fault-free, one of the comparators will produce a logic "1" signal. The logic OR of the comparator outputs will constitute a *pass/fail* signal.

The above procedure is based on the fact that the fault-free CUT produces one of the signatures from the set (1). If the actual signature is α^0, the comparator connected directly to the signature register produces a logic "1", thus indicating that the CUT is fault free. If the actual signature is α^6, then the product $\alpha^6\alpha$, generated at the output of the first α-multiplier equals to 1, which is detected by the next comparator. The same reasoning applies to the rest of signatures from the set (1). The logic diagram of the *n*-bit comparator is shown in Figure 7. It simply represents an *n*-bit AND gate.

Time overhead If time overhead is allowed, the hardware complexity can be further reduced. In terms of implementation, it is more convenient to use the following seed value: $\alpha^{-(j+i+m+1)}$, where *m* is the number of output responses. For the above example, $\alpha^{-(11+3)} = \alpha^0$, and the set (1) will change to:

$$\alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6 \qquad (2)$$

After the last shift, the SA continues to shift its content 2*m*+1 more times, while the input i is forced to 1. This ensures that the SA content is multiplied by α with each shift. For the above example, 2*m*+1=5. If within this time, the match with an element of the set (2) has been determined, the CUT is considered to be fault-free. Otherwise, it is faulty.

If the CUT is fault free and its output responses have not exceeded their tolerances, then while cycling through the states during the extra 2*m*+1 shifts, the output of the multiplexer in Figure 5 will go through the power α^0 or vector $0\ldots01$. The match with the vector $0\ldots01$ is detected by the comparator of Figure 7 connected to the multiplexor's output. The comparator output is actually producing a *pass/fail* signal.

IV. Conclusion

We examined an algebraic signature analysis method that can be employed for mixed-signal circuits testing. We demonstrated how to design the appropriate device. This device does not produce arithmetic carries and is therefore less prone to errors. The absence of carry propagating circuits also contributes to the higher performance of the device.

The proposed scheme can also be used in arithmetic and algebraic error-control coding, as well as cryptography.

We did not analyze a noise that can corrupt the output responses of mixed-signal signal circuits. It was assumed that the noise has a zero mean and its effect is neutralized in a long run of test experiments.

References

[1] R. Frohwerk, "Signature analysis: a new digital field service method," *Hewlett Packard J.*, vol. 28, no. 9, pp. 2–8, 1977.

[2] G. Starr, Q. Jie, B. Dutton, C. Stroud, F. Dai, and V. Nelson, "Automated generation of built-in self-test and measurement circuitry for mixed-signal circuits and systems," in *Proc. 15th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2009, pp. 11–19.

[3] C. Stroud, J. Morton, T. Islam, and H. Alassaly, "A mixed-signal built-in self-test approach for analog circuits," in *Proc. Southwest Symposium on Mixed-Signal Design*, 2003, pp. 196–201.

[4] N. Nagi, A. Chatterjee, Y. Heebyung, and J. Abraham, "Signature analysis for analog and mixed-signal circuit test response compaction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 6, pp. 540–546, 1998.

[5] G. Alefeld and G. Mayer, "Interval analysis: Theory and applications," *Journal of Computational and Applied Mathematics*, vol. 121, no. 1-2, pp. 421–464, 2000.

[6] R. Moore, R. Kearfott, and M. Cloud, *Introduction to Interval Analysis*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2009.

[7] W. Peterson and E. Weldon, *Error Correcting Codes*. Cambridge, MA: The MIT Press, 1972.

[8] V. Geurkov, "Optimal choice of arithmetic compactors for mixed-signal systems," in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2012, pp. 182–186.

[9] V. Geurkov, V. Kirischian, L. Kirischian, and R. Sedaghat, "Concurrent testing of analog-to-digital converters," *I-managers J. on Electronics Engineering*, vol. 1, no. 1, pp. 8–14, 2010.