Design Verification by Test Vectors and Arithmetic Transform Universal Test Set

Katarzyna Radecka, *Member*, *IEEE*, and Zeljko Zilic, *Member*, *IEEE*

Abstract—In this paper, we investigate methodology for simulation-based verification under a fault model. Since it is currently not feasible to describe a comprehensive explicit model of design errors, we propose an implicit fault model. The model is based on the Arithmetic Transform (AT) spectral representation of faults. The verification of circuits under the small errors in spectral domain is then performed by the Universal Test Set (UTS) approach to test vector generation. The major result in this paper shows that, for errors whose AT has at most t nonzero coefficients, there exist the UTS test vector set of size $O(n_2^{\log t})$. Consequently, verification confidence can be parameterized by the size of the error t, where at most $O(n_2^{\log t})$ verification vectors are simulated to verify the absence of faults belonging to such an implicitly defined fault class. The experimental confirmation of the feasibility of verification using such UTS is presented, together with the relations between the Arithmetic and Walsh-Hadamard spectra that bound the AT error spectrum and show that a class of small error circuits has small error spectrum. The proposed approach has the advantage of compatibility with formal verification and testing methods.

Index Terms—Verification, error modeling, spectral methods, arithmetic transform, Walsh-Hadamard transform, Reed-Muller transform, Universal Test Set.

1 Introduction

Modern microprocessors, embedded and signal processors, as well as communication integrated circuits utilize various arithmetic circuits in their datapaths. The implementations of such datapaths vary in area, delay, and power constraints. Hence, a broad spectrum of hardware realizations can be found, from custom to those that are modified from the standard library elements. Their design, testing, and verification pose a major challenge.

Verification of arithmetic circuits has exposed limits of the early formal verification methods based on Decision Diagrams (DDs). Such solutions initially used Reduced Ordered Binary Decision Diagrams (ROBDDs) that were applied to verify in polynomial time circuits like adders, but are of exponential size for multipliers [5]. Numerous extensions to ROBDDs were proposed that made the verification of arithmetic circuits more efficient. The most relevant are word-level diagrams, like Binary Moment Diagrams, *BMDs [6]. The most inclusive class of such diagrams is that of Word Level Decision Diagrams (WLDDs) [28]. However, the common limitation of any WLDDs is their inability to represent dividers in polynomial size.

WLDDs have been used for equivalence checking, which is a procedure that verifies a function implementation against its specification. This approach is analogous to testing the function behavior for each input combination. However, we will explore the case when nonexhaustive tests can be *provably* performed for a set of well-defined design errors. Then, one could approach the verification problem by devising a set of *test vectors* that target all these faults. Further, the effort in developing verification vectors could be directed toward detecting a source of an error [15], as well as providing stimuli for manufacturing testing. All that is not possible to achieve by formal verifications alone, which are additionally known not to scale well.

In this paper, we present new results on simulation-based verification by error modeling. The existing design error models, e.g., from [8], attempt to explicitly capture design failures at the gate level. Since there is no established design error model yet, unlike in the manufacturing fault testing, we consider an *implicit error model*, in a scheme that provides a bridge between formal and simulation-based verifications. Traditional simulation-based verification methods can only assess the quality of test vectors by actual simulations under explicit fault models. This paper provides the provable bounds on the quality of simulation-based schemes using the characteristics of the faults, instead of only resorting to simulations. For representing errors and test vector generation we use Arithmetic Transform (AT), which is exactly the underlying representation used in WLDDs. We provide the theoretical bounds and present the experimental demonstration of verification using AT as gate-level design errors are bounded in terms of their AT representation size. Finally, we show that such test vector generation can be also applied successfully to explicit design faults [8], considered recently for verification by error modeling.

In this paper, we assume the combinational model of circuits under verification; we do not address the FSM

- *K. Radecka is with the Department of Electrical and Computer Engineering, Concordia University, 1455 de Maisonneuve West, S-H-961, Montreal, Quebec H3G 1M8, Canada. E-mail: kasiar@ece.concordia.ca.*
- *Z. Zelic is with the Microelectronic and Computer Systems Laboratory, Department of Electrical and Computer Engineering, McGill University, 3480 University Street, Montreal, Quebec H3A 2A7, Canada. E-mail: zeljko@ece.mcgill.ca.*

Manuscript received 23 Apr. 2002; revised 30 Jan. 2003; accepted 29 Oct. 2003.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 116391.

verification. Note, however, that this and other combinational circuit verification procedures can successfully tackle sequential circuits. The key observation is that internal registers can, similarly to scan-based testing, serve as pseudoprimary I/Os. Hence, values of registers within the circuit can be observed and set at any individual simulation run.

The paper is organized as follows: In Section 2, we present the simulation-based verification scenario under error model assumption. Then, in Section 3, we describe a transform-based scheme that will allow us to use the universal testing and diagnosis approach. The main result is given by Theorem 1 in Section 4.2. The theoretical substantiation of the proposed approach, based on the spectrum comparison between AT and Walsh-Hadamard Transform, is elaborated in Section 5. Experiments on various arithmetic circuits and benchmarks are included to demonstrate the use of our approach for explicit design faults. The appendices contain key proofs.

2 Verification by Simulations

Implementation verification can be carried out either by formal or simulation-based approaches. While there are many advances in formal verification, simulation-based methods are still predominant in practice. It is now widely accepted that neither formal nor simulation methods can alone successfully verify wide classes of circuits [7]. Ideally, we would want to treat these two approaches as complementary methods. However, as they rely on vastly different data structures, it is not simple to merge them into one verification flow. We propose a new simulation-based verification scheme that addresses the problems of incompatibility of data structures through the utilization of Arithmetic Transform, which is the same representation used in formal verification of datapaths by word-level decision diagrams [28].

Traditionally, simulation-based verification methods fall within one of the following categories: *code coverage*, *functional testing*, *mutation testing*, and *error modeling* techniques. The code coverage methods closely follow the hardware description language (HDL) representation of a circuit and attempt to exercise all statements, execution paths, and individual expressions. Verification vectors are generated to guarantee satisfactory coverage for any of the above cases. The functional testing is an older and somewhat more vaguely defined coverage method where test vectors exercise "typical functions." For example, in a system accessing a RAM memory block, typical functions tested would be those of a read and a write operation. Test suites are produced either manually or semi-automatically. Among other simulation-based verification schemes, a recent trend lies in employing the traditional software testing methods. Mutation testing is one such method that randomly, yet systematically, changes the source code text by using a model of faulty code elements [30]. The underlying assumptions behind mutation testing are that the design is almost correct and that vectors that can detect simple errors detect complex ones as well. Behavioral level descriptions are predominantly in a form of hardware description languages like VHDL or Verilog. Therefore, any potential error can be classified and treated as a software error. Hence, methods like mutation testing could be very useful in verification at that stage [30].

2.1 Verification by Error Modeling

Verification of digital circuits via *error modeling* [2], [3], similarly to mutation and manufacturing testing, uses the techniques that inject faults and seek the test vectors. However, unlike mutation testing, this method deals with the circuit at the gate level, which differs substantially from the behavioral circuit description.

Common to all verification methods is the need to find a source of an error such that it can be easily removed. Therefore, fault detection is not only critical, but so is diagnosis and correction. Formal verification methods such as equivalence checking are known to be of no assistance in diagnosis, leading to substantial difficulties in finding the error source once the implementation has been proven incorrect [15]. Similarly, most simulation-based verification schemes in practice employ some sort of random test generation, which does not give any diagnosis information, in addition to not providing any confidence in the verification coverage. In our approach, we provide the test vector generation that is provably sufficient for coverage, diagnosis and error correction for a given class of errors.

2.1.1 Design Error Models

Errors injected at a netlist level can be seen as netlist module replacements and are somewhat related to manufacturing testing. Although there are not widely accepted standards for such design errors, in-depth studies resulted in classifications proposed in [2], [3], and [8]. A study of microprocessors designed in academia during a period of few years [9] shows that a large majority of gate-level design failures consists of erroneous replacements of a gate or wire in a network with another gate or wire, respectively. In particular, it was shown that, by applying modern design flows, 98.9 percent of all design errors in the DLX processor and 94.2 percent of errors in PUMA floating-point units fall within this class. Additionally, the authors in [1] reported that 97.8 percent of design errors that occur during the manual interventions belong to an error model of gate and wire replacements from [2].

According to error models proposed in [2], [3], and [8], the common design faults belong to one of the following categories: Bus Order Error (BOE), Bus Source Error (BSE), Bus Driver Error (BDE), Bus Single Stuck Line (SSL) Error, and Module Substitution Error (MSE). The first four classes correspond to errors in ordering (BOE), driving (BSE, BDE), and a logic value (SSL) on a bus (defined as one or more signal wires). The MSE refers to erroneously substituting a module by another module with the same number of inputs and outputs. This class includes gate replacement errors as well as extra/missing inverters. In this paper, we follow closely the error classification from [2] and [3]. We additionally arrange the above faults into the classes of *gate replacements* and *wire replacements*, respectively.

TABLE 1
Valuation Functions for Common Word Encodings

Word	Valuation $V(x)$		
	Unsigned	Sign Extended	2's Complement
Integer	$\sum_{i=0}^{n-1} x_i 2^i$	$(1-2x_{n-1})\sum_{i=0}^{n-2} x_i 2^i$	$\sum_{i=0}^{n-2} x_i 2^i - x_{n-1}2^{n-1}$
Fractional	$\sum_{i=1}^{n-1} x_i 2^{-i}$	$(1-2x_0)\sum_{i=1}^{n-1} x_i 2^{-i}$	$-x_0 + \sum_{i=1}^{n-1} x_i 2^{-i}$

3 Arithmetic Transform and Implicit Error Model

Arithmetic Transform (AT), also known as integer-valued Reed-Muller (RM) polynomial [13], extends the RM expansion to *pseudo-Boolean functions*, which have non-Boolean (e.g., integer valued) outputs, while the inputs remain Boolean. RM Transform of a Boolean function *f* is obtained by applying Davio expansion around each input variable *x*, *y*, etc., as follows:

$$f = f|_{x=0} + x(f|_{x=1} - f|_{x=0}). \quad (1)$$

In the case of RM expansion, the arithmetic is performed over finite field GF2, i.e., modulo 2; consequently, "+" and "-" denote the XOR operation. In our case, AT is obtained by using instead the word-level (e.g., integer) addition in (1).

Arithmetic Transform of a pseudo-Boolean function *f* is calculated by applying the expansion from (1) to each variable, leading to a polynomial:

$$f = \sum_{i_0=0}^{1}\sum_{i_1=0}^{1}\cdots\sum_{i_{n-1}=0}^{1} c_{i_0 i_1 \cdots i_{n-1}} x_0^{i_0} x_1^{i_1} \cdots x_{n-1}^{i_{n-1}}. \quad (2)$$

AT expresses a function using the set of linearly independent functions defined as:

$$x_0^{i_0} x_1^{i_1} \cdots x_{n-1}^{i_{n-1}}, \text{ where } x_j^{i_j} = \begin{cases} x_j, & i_j = 1, \\ 1, & i_j = 0, \end{cases}, j = 0, \ldots, n-1.$$

Coefficients $c_{i_0 i_1 \cdots i_{n-1}}$ are calculated as the inner product of function f and the basis vector $x_0^{i_0} x_1^{i_1} \cdots x_{n-1}^{i_{n-1}}$ over real numbers (however, the arithmetic is most often restricted to integers).

Integer coefficients $c_{i_0 i_1 \cdots i_{n-1}}$ are called the *arithmetic spectrum*. Each spectral coefficient multiplies an orthogonal basis function. Basis functions are often related to discrete Fourier Transform, such as those in Walsh-Hadamard Transform (WHT). In the case of AT, the basis consists of monomials and the transform is simply treated as a polynomial.

In this paper, we consider the application of AT to verification of datapath circuits. To quickly obtain arithmetic spectrum for such circuits, we use an auxiliary valuation function $V(x)$ equal to a value that a binary-represented number takes. Table 1 contains valuations of frequently used integer and fractional data types.

As an example, we consider the Arithmetic Transform of an adder. Here, the numerical value of the sum of two n-bit unsigned numbers x and y is determined as:

$$V(x+y) = \sum_{i=0}^{n-1} (x_i + y_i)2^i.$$

Comparing with (2), we notice that this is a polynomial with integer coefficients representing a multiple-output function of arguments x_i and y_i, where $i = 0, \ldots, n-1$. As a result, AT of the addition operation has $2n$ nonzero coefficients. Similarly, the subtraction is obtained by replacing the arithmetic "+" with a "-" sign. For a sign-extended encoding, the operation is:

$$V(x-y) = \sum_{i=0}^{n-2} (x_i - y_i)2^i - 2x_{n-1}\sum_{i=0}^{n-2} x_i 2^i + 2y_{n-1}\sum_{i=0}^{n-2} y_i 2^i.$$

The other basic elements of datapath, e.g., multipliers, can also be represented in a straightforward way by using $O(n^2)$ spectral coefficients:

$$V(x * y) = \sum_{i=0}^{n-1} x_i 2^i \times \sum_{i=0}^{n-1} y_i 2^i.$$

This expression leads to n^2 spectral coefficients after the sums are multiplied out. In practice, this number can be reduced to $2n$ by keeping the polynomial in the above factored form. We note that multipliers are kept in factored form in representation by *BMDs [6] and related WLDDs.

The extension to the more complex arithmetic expressions is straightforward. A multiple-output Boolean function is described by a single polynomial, i.e., its arithmetic spectrum. For example, the arithmetic spectrum of $x + yz$ is a simple expression:

$$V(x+yz) = \sum_{i=0}^{n-1} x_i 2^i + \sum_{i=0}^{n-1} y_i 2^i \times \sum_{i=0}^{n-1} z_i 2^i.$$

For example, any FIR filter with *m* coefficients $a_1 a_2 \cdots a_m$ and *mn*-bit integer inputs $x_1 x_2 \cdots x_m$ can be represented using *mn* spectral coefficients:

$$V(a_1x_1 + a_2x_2 + \cdots + a_mx_m) = \\ \sum_{i=0}^{n-1} (a_1x_{1i} + a_2x_{2i} + \cdots + a_mx_{mi})2^i.$$

The application of AT to alternative data types results in the spectrum of size comparable to that for the unsigned integer encoding.

3.1 Calculation of Arithmetic Transform

Although we will not require the exact identification of the faulty AT, we use its derivation in the analysis to follow. AT of an arbitrary pseudo-Boolean function can be formed through multiplying the vector of function values by the transform matrix T_n, defined recursively as:

$$T_n = \begin{bmatrix} T_{n-1} & 0 \\ -T_{n-1} & T_{n-1} \end{bmatrix}, \qquad T_0 = 1. \quad (3)$$

This matrix has 2^{2n} entries—the transform, obtained by multiplying T_n with the vector of values, consequently requires $O(2^{2n})$ operations. There exist much faster ways of determining AT, including the interpolation algorithm that is quadratic in the number of interpolated points [32].

Computationally efficient is Fast Arithmetic Transform implementation which, in $O(n^{2n-1})$ time and $O(2^n)$ space, employs recursively the expansion from (1). This approach can be used in conjunction with decision diagrams for reducing the execution time and producing graph descriptions such as *BMDs. The arithmetic spectrum can also be generated as a polynomial interpolated from the values that a function takes [32].

Yet another way of generating the arithmetic spectrum is derived by considering pseudo-Boolean functions $f: 2^n \mapsto [0..2^m - 1]$ by means of the Boolean lattice $B = 2^n$. To form the lattice structure, the *partial order* relation $\prec$ is defined on Boolean vectors (points in the lattice 2^n) as follows: For vectors x and y, we say that $y \prec x$ if the coordinates of x that are "0" are the subset of 0-coordinates of y, for example, $\prec$ 1010. Incomparable vectors exist, such as 1010 and 0110. Vectors with i ones belong to the same ith layer in the lattice. For n-variable functions, the ith layer contains $\binom{n}{i}$ vectors.

Arithmetic Transform can also be obtained by traversing the lattice in the increasing order of points. Following the quadratic time interpolation algorithm in [32], at each point x, the spectral coefficient c_x is calculated by subtracting all the preceding coefficients from the function value at x, starting from the bottom value $f_\perp = f_{00\cdots 0}$:

$$c_x = f_x - \sum_{y \prec x} c_y, \qquad c_\perp = f_\perp. \tag{4}$$

Consider, for example, transforming the adder function $a + b$, for the 2-bit unsigned inputs: $a = a_1a_0$ and $b = b_1b_0$. Each point in the lattice corresponds to an assignment of inputs in the order of variables: $a_1a_0b_1b_0$. The spectral coefficients are generated by applying (4) to function values in an increasing lattice order, i.e., $c_{0000} = f_{0000} = 0 + 0 = 0$, $c_{0001} = f_{0001} - c_{0000} = 0 + 1 - 0 = 1$,

$$c_{0010} = f_{0010} - c_{0000} = 0 + 2 - 0 = 2,$$

$c_{0100} = 1$, and $c_{1000} = 2$. All other coefficients are 0, as inscribed on Fig. 1, where nonzero coefficients are highlighted. The resulting polynomial is:

$$AT(f) = (a_0 + b_0) + (a_1 + b_1) * 2.$$

By extending this construction to n-variable functions, it is easy to see that, for unsigned arithmetic functions, all nonzero coefficients of AT are in layer 1 (for adders) or layer 2 (multipliers) of the lattice.

In the proposed verification scenario, an arithmetic spectrum is used as a specification of an arithmetic operation. The form of an AT polynomial for a given arithmetic operation is known and depends only on the data type used. The knowledge of the shape of a canonical representation (in our case, a polynomial) is typically used in formal verification to check whether the circuit is correct. Contrary to this, we use the shape of *error* polynomials to verify the absence of an error by a minimum amount of vector simulations.

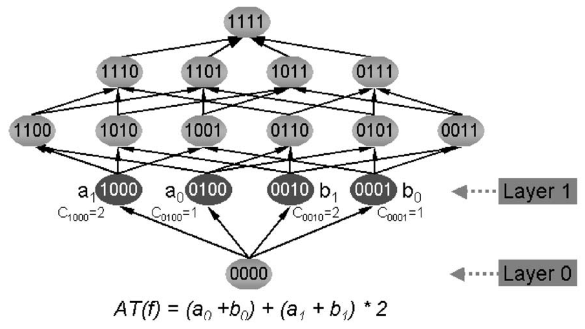

Fig. 1. Lattice Structure 2^4—Transform of Adder.

3.2 Implicit Error Model

In verification by error modeling, a set of test vectors is required to check that a circuit contains no error from the model. A circuit can be treated as a *black box*, by which a description of a design error can be obtained by subtracting responses of an erroneous circuit from corresponding responses of its (correct) specification. This is an *implicit error model* given by the Arithmetic Transform of a difference between the correct and faulty circuits. We next derive efficient fault detection methods for errors whose AT has at most t nonzero coefficients.

Since, in our implicit error model, a fault is treated as a quantity added to the circuit output, the behavior $\tilde{f}$ of the faulty circuit is represented as a sum of the correct output and the error function e, i.e., $\tilde{f} = f + e$. As AT is linear, the relation:

$$AT(\tilde{f}) = AT(f + e) = AT(f) + AT(e)$$

is satisfied. The size of the error is the number of nonzero coefficients of the error e, i.e., $AT(e)$.

Although the polynomial e for each such error can be obtained by simulating the faulty and correct circuits and subtracting their outputs, we emphasize that this verification scheme and the analysis to follow do not require the complete explicit identification of an error. Consequently, AT transformation algorithm does not need to be invoked. We rather relate the implicit class of small AT spectrum errors to the explicit model from Section 2.1.1 and then derive the conditions for detecting such faults.

3.2.1 Arithmetic Transform of Basic Design Errors

We now express the design error classes from [8] in terms of their spectral representations. Most of these categories can be described as errors with few spectral AT coefficients. The basic error types identified in [8] are the bus errors: *Bus Order Error* (BOE), *Bus Source Error* (BSE), *Bus Driver Error* (BDE), *Bus Single Stuck Line* (SSL), and *Error and Module Substitution Error* (MSE). For a single bus in isolation, the error spectrum is compact for most of the above classes, as shown next.

Bus Order Error. This group includes a common design error of incorrectly ordering bits in a bus. For example, if the signals x_l and x_k of the bus x, $x = x_0x_1 \ldots x_{n-1}$ have been interchanged, the transform of this bus considered in isolation is:

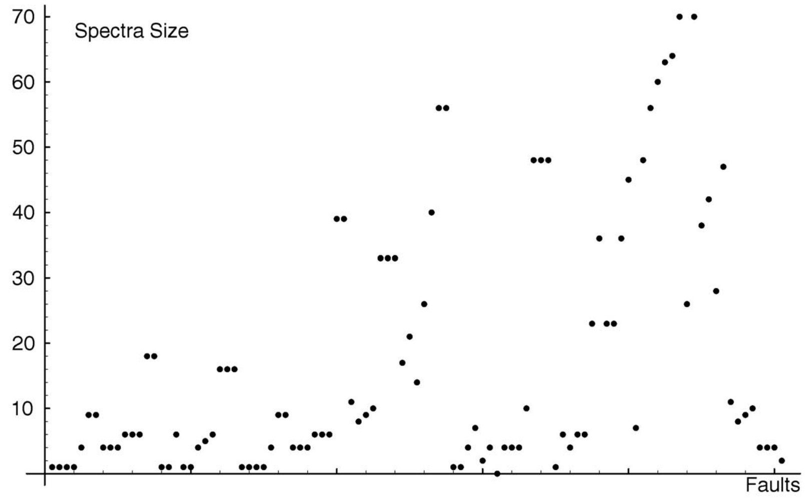

Fig. 2. Spectrum size distribution of stuck-at faults in 4×4 multiplier.

$$AT(\tilde{f}) = \sum_{i=0}^{n-1} (x_i * 2^i) + x_k * 2^l - x_k * 2^k + x_l * 2^k - x_l * 2^l.$$

If the correct circuit transform was $AT(f)$, the transform of the faulty one is:

$$AT(\tilde{f}) = AT(f) + x_k * 2^l - x_k * 2^k + x_l * 2^k - x_l * 2^l.$$

The error polynomial has four nonzero spectral coefficients. In general, any permutation of signals of an n-bit bus will have the error transform with at most $2n$ spectral coefficients.

Bus Source Error. This class represents errors which cause the replacement of the intended source x_k with the source r_k. Arithmetic Transform of the error is $AT(e) = r_k * 2^l - x_k * 2^k$.

Bus Driver Error. This kind of error corresponds to a bus being driven by two sources. It manifests itself in a way dependent on the implementation technology. For example, if a bus line is realizing a "wired-OR," then, by connecting an additional source r_k to a line x_k, the resulting signal is $x_k \vee r_k$. Using integer arithmetic, the logical *OR* is obtained as $x_k \vee r_k = x_k + r_k - x_k * r_k$. This identity leads to the following AT of the additive error $AT(e) = (r_k - x_k * r_k) * 2^k$.

Module Substitution Error. A module is substituted by another module with the same number of inputs and outputs. Depending on the circuits replaced and their position in a logic network, various transforms can be obtained. In a simple example, where an *AND* gate producing $x_k = i_1 \wedge i_2$ is replaced by an *OR* gate generating $x_k = i_1 \vee i_2$, the error is represented as $AT(e) = (i_1 + i_2 - 2 * i_1 i_2) * 2^k$ as the two transforms are $AT(i_1 \wedge i_2) = i_1 * i_2$ and $AT(i_1 \vee i_2) = i_1 + i_2 - i_1 * i_2$. By considering single gates, the *gate replacement* error model is included in this class.

Bus Single Stuck Line. The error in which an n-bit bus is stuck at a constant value (0 or 1) is manifested as the additive error for which Arithmetic Transform has $O(n)$ nonzero spectral coefficients. For this SSL error, the transform is:

$$AT(\tilde{f}) = \sum_{i=0}^{n-1} x_i * 2^i - \sum_{k=0}^{n-1} x_k * 2^k = AT(f) - \sum_{k=0}^{n-1} x_k * 2^k.$$

Hence, the error transform is equal to

$$AT(e) = -\sum_{k=0}^{n-1} x_k * 2^k.$$

In the case of a bus stuck-at-1, the error transform has $n + 1$ nonzero coefficients:

$$AT(e) = \sum_{k=0}^{n-1} (2^k - x_k * 2^k).$$

Any combination of the SSL errors would have the error transform that is linear in the number of lines that are stuck.

Single Stuck-At Faults. In contrast to the above, the single stuck-at faults are directly used in the testing of a circuit. The effects of these faults cannot be described by a single formula. For example, the distribution of spectra of all single stuck-at faults in a 4×4 Carry Save Adder (CSA) multiplier is shown in Fig. 2. The x-axis corresponds to the faulty node; nodes are in the reverse topological order.

This figure indicates that there are a number of faults that result in a substantial error spectrum. However, regardless of the spectra size of the stuck-at faults, we demonstrate through experiments that these faults are easily detectable by the vector sets for small spectral errors.

4 Detecting Small AT Errors

The implicit error model that we study consists of all faults for which the number of nonzero AT coefficients is smaller than some assumed bound. The size of a test vector set is based on the number of spectral coefficients of an error (spectral error size). If there are only few coefficients, then errors are small and a test vector set detecting them is small, as we prove in this section. The assumption of a "small" error is partly motivated by the examples in Section 3.2.1 of the common design error classes from [8], where we have shown that most of these errors are represented by polynomials of small number of terms. Next, we investigate the existence and use of a Universal Test Set for errors parameterized by the size of the error AT.

4.1 Universal Test Set

The concept of a Universal Test Set (UTS) primarily refers to a test set that can summarily detect a complete well-defined class of errors. In practice, the application of the UTS notion can take several forms. Depending on the circumstances, UTS is used to either detect all faults (usually stuck-at) in a specific class of circuit implementations [18] or test for all possible faults in an unknown circuit implementation [14], [19]. In our case, we develop UTS that detects the class of all faults whose AT spectrum size is smaller than a given bound.

The natural extension of the above concept is that of the Universal Diagnosis Set (UDS), which is the set of test vectors that can uniquely identify all faults of bounded AT size. Similarly, one defines the Universal Correction Set (UCS) as a set of vectors that allow correcting of the faulty circuit.

4.2 AT-Based Universal Test Set

To derive the test set, recall that AT of a pseudo-Boolean function can be obtained by multiplying the function values by the transform matrix T_n, (3). For $n = 3$, matrix T_3 is:

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix}. \quad (5)$$

The columns of matrix T_n correspond to combinations of function inputs which can be considered as test vectors. In our case, this constitutes UDS for detecting design errors. Hence, obtaining a test set amounts to choosing a number of columns from T_n. The primary goal is to have such a selection that is suitable for reconstructing the function under the presence of an error by a minimal number of test vectors.

We notice that the polynomial representing a given function can be found by inverting the matrix T_n. The structure of this matrix is identical to that of the RM error correcting code check matrix [26], albeit with integer (considered as a subset of real numbers) arithmetic, rather than that over finite field GF2. We show that the redundancy incorporated in the matrix allows us to find a minimal test, diagnosis, and correction set for the case of a bounded spectrum error.

To derive conditions for finding UDS, we use an auxiliary *error check matrix* H_r that consists of the rows of T_n^{-1} corresponding to the vectors in the top $r+1$ lattice layers. This matrix has 2^n columns; the number of rows is equal to the number of points in these to $r+1$ lattice layers.

Lemma 3 in Appendix A states that the matrix H_r has at least $2^{r+1}-1$ independent columns. We need this information to prove the error correcting capability of UDS. The following theorem is used to derive the test set for the case of errors of bounded spectrum.

Theorem 1. *Consider an error superimposed to an n-variable function. Any error whose AT has at most t spectral coefficients can be uniquely identified by examining*

$$V = \sum_{i=0}^{\lceil \log_2(t+1) \rceil - 1} \binom{n}{i}$$

points (vectors) in $\lceil \log_2(t+1) \rceil - 1$ *upper layers of the lattice.*

Proof. By selecting all the points from the upper $\lceil \log_2(t+1) \rceil - 1$ lattice layers, we obtain a reduced matrix $H_{\lceil \log_2(t+1) \rceil - 1}$ of size $V \times 2^n$. It is sufficient to check that each $2t$ columns of this matrix are independent to detect and correct any error polynomial with up to t terms. According to Lemma 3 in Appendix A, the minimal number of independent rows is: $2^{\lceil \log_2(t+1) \rceil - 1 + 1} - 1 \geq 2t$. Therefore, any polynomial with up to t terms is uniquely identified. □

Similar to Fig. 1, where bottom lattice layers (0 and 1) suffice to encode arithmetic circuits by AT, the top $O(\log_2 t)$ layers are needed to detect and correct a t term error AT. For test vector generation, matrices T_n and H_r do not need to be explicitly constructed as they were needed only for proving Theorem 1.

A proof of the similar theorem for detecting faults by binary RM Transform is found in [11] and [26]. A nonbinary input generalization had been proven in [12] for a multiple-valued RM Transform and the finite field arithmetic. Note that the RM Transform is of exponential size even for adders, while the corresponding AT is of polynomial size, hence our result is more practical. Also, Theorem 1 generalizes results in [11] to nonbinary outputs and word-level arithmetic.

Note that Theorem 1 provides an upper bound on the number of points to be simulated for unique identification of the class of t-term AT polynomial errors. For fault diagnosis and correction purposes, Theorem 1 guarantees that an error superimposed on a correct circuit will be exactly identified by a minimum number of test vectors. In actual circuits, faults that involve many more spectral coefficients will be detected if the only goal is error detection. A stronger statement related to error detection alone is given next.

Corollary 1. *Any error whose AT has at most t spectral coefficients can be uniquely identified by examining vectors in* $\lceil \log_2(t+1) \rceil - 2$ *layers of the lattice.*

Proof. The number of independent rows is t, which is sufficient for detecting whether the linear system is inconsistent, but is insufficient for correction. □

4.3 AT-Based Test and Correction Procedure for Implicit Errors

The properties just proven suffice in deriving the verification test procedure for detecting a t-term error. It is sufficient, by Corollary 1, to simulate a circuit under verification with $O(n_2^{\log t})$ vectors, which constitute the UTS for t term faults. The procedure is given in Algorithm 1.

```
Inputs: n-variable circuit C, specification f, size of the error t
Outputs: Circuit is correct up to a t-term AT polynomial (pass or fail)
      Vectors = Select_UTS_by_lattice_layers(n, t);
      foreach v ∈ Vectors
            if Simulate(C, v) ≠ f(v) return fail ;
   return pass;
```

The number of required simulation runs is hence polylogarithmic in t, which itself is a parameter in range $(0, 2^n)$. The verification procedure is then parameterized by the value that t takes. Unsurprisingly, in the extreme case of verifying for up to 2^n term errors, the complexity equals that of the verification by exhaustive simulation, i.e., $O(2^n)$.

In the same way, Theorem 1 provides us with both a UDS and UCS vector set for error detection and correction of a faulty function $\tilde{f}$. The simulation of the UCS set alone allows us to exactly identify the error e. Then, in order to correct the faulty circuit, it is sufficient to construct a circuit implementing the error function e and subtract it from the faulty circuit $\tilde{f}$, Fig. 3.

The error size t is critical in establishing the confidence in the overall procedure. It also impacts the amount of computational resources. Hence, the need for determining this parameter, in general, depends on the required level of confidence in verification result, as well as other parameters like the availability of simulation resources. The desired confidence can vary throughout the design process. Sometimes, quick regression checks may assume small errors, while, at some other steps, a more thorough design check might be useful. The remainder of this paper provides insight in selecting the error size parameter t, based on the actual design errors and the consideration of the error circuit structure. First, in Section 5, we provide arguments for reasoning that, when the injected error is small as a circuit, the error spectrum will be bounded. Second, in Section 6, we consider the actual fault detecting capability for explicit faults through experiments.

5 Bounding Error Spectrum through Walsh-Hadamard Transform

To extend the study in Section 3.2.1, of how the implicit errors relate to superimposed error circuits e (Fig. 3), we now consider errors represented as classes of circuits, rather than polynomials, superimposed on the correct design implementation. Such erroneous "additions" to the fault-free circuit are likely to be "small"; otherwise, they would easily be detected in the design process. A more formal definition of small circuits should avoid dependency on the design implementation (e.g., Sum of Products versus Product of Sums) and gates used (e.g., OR versus XOR). In theory, the class of "constant depth circuits" is often used to represent small circuits [21]. Such circuits do not increase gate depth as the number of inputs increases. Results in [21] show that the Walsh-Hadamard (WHT) spectrum of the class of constant depth circuits is small and concentrated in low order coefficients. In this section, we show that the spectra of AT and the version of WHT are in direct proportional relation; hence, the assumption of small AT spectrum can be extended to constant depth circuits.

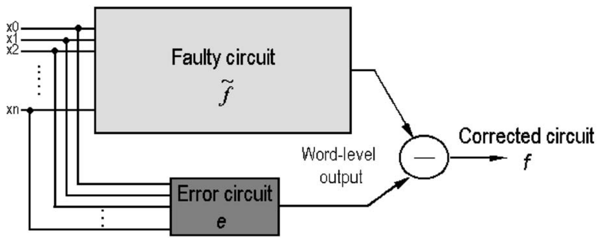

Fig. 3. Design error correction by Arithmetic Transform UCS.

Walsh-Hadamard Transform is defined by the transform matrix similar to (3):

$$T_n^{\varphi} = \begin{bmatrix} T_{n-1}^{\varphi} & T_{n-1}^{\varphi} \\ T_{n-1}^{\varphi} & -T_{n-1}^{\varphi} \end{bmatrix}, \qquad T_0^{\varphi} = 1. \tag{6}$$

The negation of rows (or columns) of matrix T_n^{φ} leads to the alternative orthogonal transform that belongs to the same family of Walsh transforms [16]. We refer to this transform as *negated Walsh Transform (nWT)*. Such negation preserves the sums of absolute values of spectral coefficients and their squares (energy density spectrum). To facilitate the comparison, we define T_n^{ϕ} by inverting half of the rows of the transform matrix T_n^{φ}:

$$T_n^{\phi} = \begin{bmatrix} T_{n-1}^{\phi} & T_{n-1}^{\phi} \\ -T_{n-1}^{\phi} & T_{n-1}^{\phi} \end{bmatrix}, \qquad T_0^{\phi} = 1. \tag{7}$$

Let us consider the case of $n = 3$. The following transform matrix is obtained:

$$T_3^{\phi} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix}.$$

Matrix T_3^{φ} has the entries in rows 2, 3, 5, and 8 inverted. To derive a relation between the nWT and arithmetic spectra, we evaluate the difference between negated Walsh and Arithmetic Transforms, referred to as A_n. For $n = 3$, we have:

$$A_3 = T_3^{\phi} - T_3^{AT} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & -1 & 0 & 1 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In general, the recursive structure of this matrix is described by the following lemma:

Lemma 1. *The recursive form of matrix A_n is:*

$$T_n^{\phi} - T_n^{AT} = A_n = \begin{bmatrix} A_{n-1} & T_{n-1}^{\phi} \\ -A_{n-1} & A_{n-1} \end{bmatrix}, \qquad A1 = 0.$$

Proof (by induction). It is sufficient to prove that $A_n + T_n^{AT} = T_n^{\phi}$.

Induction Base: For $k = 3$, we have just shown that $A_3 + T_3^{AT} = T_3^{\phi}$.

Induction Step: Assume that the claim is true for $n = k$. Then, for $k+1$, the definition is:

$$A_{k+1} + T_{k+1}^{AT} = \begin{bmatrix} A_k & T_k^{\phi} \\ -A_k & A_k \end{bmatrix} + \begin{bmatrix} T_k^{AT} & 0 \\ -T_k^{AT} & T_k^{AT} \end{bmatrix}$$
$$= \begin{bmatrix} T_k^{\phi} & T_k^{\phi} \\ -T_k^{\phi} & T_k^{\phi} \end{bmatrix} = T_{k+1}^{\phi}.$$

□

5.1 Spectrum Comparison

Using the recursively defined difference between the two transforms, we now compare the sums of the spectral coefficients, $\sum_{i=0}^{2^n-1} c_i^{\phi}$ and $\sum_{i=0}^{2^n-1} c_i^{AT}$. The first sum is calculated as:

$$\sum_{i=0}^{2^n-1} c_i^{\phi} = \sum_{i=0}^{2^n-1} \left(\sum_{j=0}^{2^n-1} T_{ij}^{\phi} f_j \right).$$

By changing the order of summation, the inner sum is calculated by adding columns of the nWT Transform matrix:

$$S(T^{\phi}) = \sum_{i=0}^{2^n-1} c_i^{\phi} = \sum_{j=0}^{2^n-1} f_j \left(\sum_{i=0}^{2^n-1} T_{ij}^{\phi} \right).$$

The same transformation is used to obtain $\sum_{i=0}^{2^n-1} c_i^{AT}$ and

$$\sum_{i=0}^{2^n-1} (c_i^{\phi} - c_i^{AT}) \qquad \text{and} \qquad \sum_{i=0}^{2^n-1} (c_i^{\phi} + c_i^{AT}).$$

The sums are obtained by matrices $A_n = T_n^{\phi} - T_n^{AT}$, $B_n = T_n^{\phi} + T_n^{AT}$, and the following theorem.

Theorem 2. *The sum $S(T^{\phi})$ of all spectral coefficients of the nWT obtained using T_n^{ϕ} is always 2^n times larger than the sum of all Arithmetic Transform coefficients.*

The proof is presented in Appendix B, together with an auxiliary characterization of matrices A_n and B_n. Theorem 2 shows that there is a direct proportional relation between AT and WHT spectra sums. This result indicates that the sum of spectral coefficients of constant depth circuits remains bounded in the AT domain. More specifically, an error circuit having a small WHT spectrum also has a small AT spectrum and, hence, could be tested by our AT-based method using a small number of lattice layers.

5.2 Spectra Distribution and Partial Spectra Comparison

In order to complete the spectrum comparison, it is important not only to know the total sums of spectral coefficients, but also their distribution. For example, in [21], the authors present the characterization of small depth circuits by their concentration in low-order spectral coefficients. To investigate the distributions of two spectra, we relate their partial spectra, i.e., subsets of spectral coefficients.

Relations similar to that in Theorem 2 can be proven for partial spectra. We say that the *upper half* of the spectrum is obtained by restricting a variable (say x_0) to 1 and denote such partial spectra as $\sum c^{\phi}\big|_{x_0}$ and $\sum c^{AT}\big|_{x_0}$. By considering the upper halves of spectral coefficients, and applying the same comparison as in Appendix B, we obtain:

$$\sum_{i=2^{n-1}}^{2^n-1} (c_i^{\phi} - c_i^{AT}) = (2^{n-1} - 1) * (f_{2^n-1} - f_{2^{n-1}-1})$$
$$\sum_{i=2^{n-1}}^{2^n-1} (c_i^{\phi} + c_i^{AT}) = (2^{n-1} + 1) * (f_{2^n-1} - f_{2^{n-1}-1}),$$

leading to:

$$\sum c_i^{\phi}|_{x_0} = \frac{2^n}{2} \sum c_i^{AT}|_{x_0}.$$

Note that the multiplicative factor $2^n - 1$ is halved relative to the one used in the total spectra comparison in Theorem 2. When the same restriction is applied to several input variables, this multiplicative factor reduces exponentially in the number of variables assigned. Extending the argument above easily proves the following theorem.

Theorem 3. *The upper part of the AT spectrum corresponding to the assignment of i variables to 1 is 2^{n-i} times smaller than that of the nWT spectrum:*

$$\sum c_i^{\phi}|_{x_0 x_1 \cdots x_{i-1}} = 2^{n-i} \sum c_i^{AT}|_{x_0 x_1 \cdots x_{i-1}}.$$

In the border case, i.e., when all variables are assigned to one, the last coefficient $c_{11\ldots1}$ is the same for both of these transforms, as can be verified by inspecting the transform matrices.

The partial spectra comparisons show that the AT spectrum of a function has higher content of high-order coefficients, while the sum of all spectral coefficients is in direct proportional relation to the nWT. In other words, functions that have small AT spectrum also have small WHT spectrum that is more concentrated in low-order coefficients. This property is characteristic for constant depth circuits, so the AT of a superimposed small error circuit will be small as well.

6 Experimental Results

So far we have presented theoretical results justifying the assumption of a small AT error in verification through simulations. While Section 2.1.1 proves that it is sufficient to simulate $O(n^{\log t})$ vectors to detect t-term AT errors, the experiments are required to find values for t that result in high coverage for concrete design errors. We omit deliberation diagnosis and error correction procedures, which are equivalent to the decoding of the integer-valued RM error-correcting codes. Please note that the considered implicit small AT error model represents a large number of the *explicit* gate and wire errors. In practice, where the simulations used for verification purpose take much of the overall time, considering the implicit error model can

TABLE 2
Coverage of s-a-v and Selected Gate Replacement Faults with Lattice Layers

Circuit	Single s-a-v			Single gate replacement		
	2nd	3rd	≤ 4	AND	OR	XOR
i1	86.9	86.9	86.9	50.0	66.7	85.7
alu2	91.1	97.7	98.5	97.8	96.6	100
alu4	90.1	96.1	98.9	97.8	98.2	100
9symml	49.9	50.1	100	100	100	74.0
Cordic	74.5	87.0	98.0	100	98.0	44.4
my_adder	100	100	100	97.6	100	100
Mux	90.0	100	100	100	100	100
Parity	96.7	96.7	100	100	100	NA
cm138a	67.5	77.5	97.5	100	100	100
Decod	76.7	74.4	100	100	100	100
f51m	100	100	100	100	100	100
z4ml	84.6	100	100	100	100	100

save many simulation cycles. Further, we can parameterize the simulation effort by the size of the error polynomial and, finally, such a simulation effort can be easily parallelized.

To find the error size t sufficient for representing *explicit* gate and wire replacement errors, we have to resort to experiments. Stuck-at faults are briefly addressed for comparison purposes. Our experimental setup consists of arithmetic and MCNC circuits synthesized into generic Synopsys gate library—GTECH. We built fault simulation and our redundant fault identification on the UC Berkeley SIS program. Experiments were run on an Apple PowerMac G2 with two 1.25GHz PowerPC processors and 512MB of main memory, under MAC OS X v10.2 operating system.

The first set of experiments compares gate replacement faults with stuck-at-value faults. Table 2 reports the coverage of single stuck-at faults and replacement faults with AND, OR, and XOR gates for MCNC benchmarks. For fair comparison and to expose the scope of redundant faults, the experiments were performed on the fault list with no redundant errors removed. Due to the redundancies that cause exhaustive simulations, only small circuits were tried. To present the effectiveness of our UDS vectors, we gradually increase the number of lattice layers used for stuck-at fault detection. The results reveal that four layers, i.e., $\log t = 4$, are sufficient for fault coverage over 95 percent. Hence, all remaining experiments will involve simulation of at most $O(n^4)$ vectors, for a given number of circuit inputs n.

Experiments also indicate that there are many more redundant single gate and wire replacement faults than in the case of single stuck-at faults. Hence, efficient methods for identifying redundant gate and wire replacement faults are crucial in obtaining a meaningful coverage. We have shown in [24] and [25] that the problem differs substantially from the case of stuck-at faults (e.g., [17]) and that the effective redundant fault identification can be obtained by combining several methods. Since redundant faults are caused by don't care (DC) conditions in a circuit, we apply Compatible Observability Don't Care (CODC) approximations [27] for quick identification of most of the redundant faults. Then, we construct a satisfiability formulation [20] for the remaining faults that the CODC approximation finds likely to be redundant. We also produced the exact SAT-only method for redundant wire and gate fault identification [24], [25].

The results for arithmetic circuits are summarized in Table 3. Dividers were chosen because their Decision Diagrams (even WLDDs) are of exponential size [28]. Reported are two sets of results: First, we replace each gate with all input-compatible gates from the library and, second, we replace each wire by another irredundant wire in netlist. We conclude that high coverage is obtained for both gate and wire replacement faults when redundancy identification is employed. In columns "Faults," we reported the total number of irredundant faults simulated. Columns "Sims" contain the total number of simulation runs. For each fault, we simulate vectors from UTS in the

TABLE 3
Arithmetic Circuit Coverage with UTS (Four Top Lattice Layers)

Circuit	Size	Gate Replacements				Wire Replacements			
		Cov.	Faults	Sims	Vec.	Cov.	Faults	Sims	Vec.
ALU	24	99.3	191	516505	80	100	1570	912585	205
	32	99.5	348	1015996	84	100	2643	1618214	244
	48	99.7	736	1424568	87	100	3231	2103455	251
CLA Divid.	11x6	100	153	8720	45	100	1012	1274	68
	17x8	100	247	18834	51	95.9	2232	1683	76
	33x16	100	579	37792	75	94.1	4121	10514	97
Array Divid.	11x6	100	96	1654	17	100	825	753	7
	17x8	100	149	2147	18	100	1043	958	7
	33x16	100	293	4738	20	100	1964	1168	8

TABLE 4
Fault Coverage, Running Times for Redundancy Identifications, and Simulation Statistics for Gate Replacement Faults on MCNC Benchmarks

Circuit	DC BDD		Redund. Id			Vectors		
	size	Time	Red [s]	Total [s]	Cov. [%]	Faults	Sims	Vec.
i1	180	0.05	0	0	94.4	18	3093	10
alu2	2187	0.40	0.14	3.27	96.2	26	812	15
alu4	2148	0.94	4.51	35.7	95.9	49	6075	30
9symm	1252	0.95	0.04	0.16	97.5	6	190	2
cordic	401	0.21	0.02	0.04	92.8	28	37417	19
C499	64547	5.80	0	0.59	100	162	92466	84
C432	173829	6.07	0.15	0.31	100	167	11750	40
C1355	176390	153.7	1.99	2.26	100	68	13428	22
C1908	443558	218.4	1.71	2.11	91.2	98	650879	29
C2670	4401323	217.8	1.76	4.16	98.5	330	10088	36
C6288	∞	∞	35	55.9	100	705	66003	271
C880	30501	5.31	0.1	0.47	97.6	188	6005874	106

TABLE 5
Fault Coverage, Running Times for Redundancy Identifications, and Simulation Statistics for Wire Replacement Faults on MCNC Benchmarks

Circuit	DC BDD		Redund. Id.		Vectors		
	Size	Time[s]	Time [s]	Cov [%]	Faults	Sims	Vec.
alu2	2239	0.391	0.137	100	602	814	5
alu4	2849	0.883	0.226	100	737	2958	15
9symm	2884	0.218	0.355	100	196	1302	19
cordic	926	0.137	0.202	100	219	36128	28
C499	73120	4.581	2.484	96.2	946	123231	87
C432	271463	6.258	2.177	100	927	7180	43
C1355	190077	123.88	7.359	97.6	1407	204206	139
C1908	120999	185.40	10.181	90.4	818	2722549	96
C2670	3513027	213.56	6.23	99.9	436	45023	31
C6288	∞	∞	19.867	94.9	1852	145402	67
C880	36406	4.419	0.56	97.0	283	7042301	89
My_adder	4253	0.073	0.012	100	14	3605	14

decreasing lattice order until first fault detection. Columns "Vec." report vectors that would suffice for the given coverage if simulated alone. They are obtained by counting the number of distinct vectors that for first time detect a fault. Note that this number could be reduced through standard vector compaction methods.

The coverage with the four top layers for MCNC circuits with all gate and wire replacement faults is recorded in Table 4 and Table 5, respectively. Experiments again show that high fault coverage is obtained by using the small error spectrum assumption in UTS test pattern generation. These two tables also report the time and space needed to construct BDDs (needed in one of our redundancy identifications). This data is useful for comparing the cost of our method with equivalence checking by BDDs. Columns "Redund. Id" indicate the times required for and coverage with our exact SAT-based redundancy identification. Finally, the last three columns report the number of all faults simulated, total number of simulation runs for coverage by exact identification, as well as the total number of vectors that suffice for the given coverage, as in Table 3. Time spent in simulations depends on the circuit simulation speed, but note that this task is amenable to massive parallelization as both the circuits and the vectors are known in advance. The reported number of simulations is that needed with a naive serial simulator and any fault simulator would significantly reduce the actual number of simulation runs.

6.1 Improvements—Neighborhood Subspace Points

Although only up to four lattice layers (plus vector 11 ...1) were needed for good coverage, we considered the alternative schemes that use a subset of the considered lattice points. The UDS approach of generating test vectors was enhanced by a test set reduction scheme in which the tests are still independent of the circuit implementation. We use additional information from the AT specification, rather than the circuit structure.

While the original test vectors exhaustively cover the four top lattice layers, we also considered exhaustively covering only the *neighbor* variables among these lattice layers. For example, in adders, each expression $(a_i + b_i)2^i$ of the adder AT specification joins together the neighbor variables in a polynomial term that is multiplied by the same constant 2^i. Additionally, a *carry out* bit propagates from the *i*th to the (*i* + 1)th stage. We then deem the neighboring inputs to be a_i, b_i, a_{i+1}, and b_{i+1}, Fig. 4, and insure that only such four bits are simulated exhaustively among the four top lattice layers.

In [23], it was shown by experiments that almost no coverage for arithmetic circuits is sacrificed compared to original UDS. While testing with all the vectors belonging to the top four layers requires $O(n^4)$ points, the subset that is exhaustive only for all the neighbor variable combinations contains only $O(n^2)$ vectors. The savings are equivalent to using two layers less in the test set.

7 Conclusions and Future Work

We proposed a vector-based verification of datapath circuits using Arithmetic Transform and the concept of implicit error modeling. We have shown that this approach can be applied to derive effective test sets for several classes of design errors and has inherent diagnosis and error correction capabilities. By employing AT, the whole approach is compatible with commonly used formal verification representations. Further, the verification process can be combined with the test pattern generation by reusing verification vectors for detecting manufacturing faults or vice versa. Additionally, as the deep submicron faults might need a fault model that is closer to gate replacements than to stuck-at model [1], dealing with gate

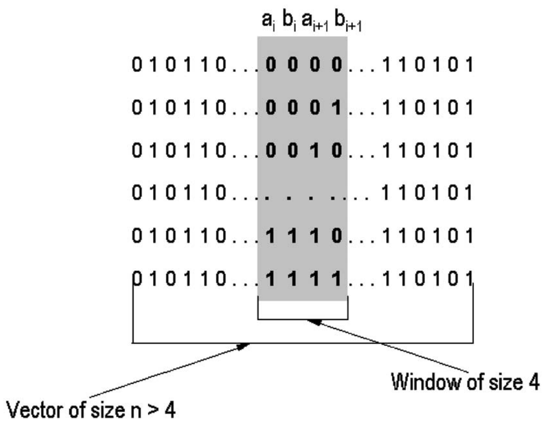

Fig. 4. Exhaustive coverage of vector subspace of size four.

and wire replacements might become a necessity in manufacturing fault testing.

The compact circuit representations and the capability of relating the common errors to the bounds on the vector set size are achieved by using the arithmetic spectrum. This provides confidence in restricting an otherwise exhaustive test set, without sacrificing the fault detection capability. The improvements to the basic concept include the use of the high-level information on the input variable dependences, through neighbor window variables, as outlined in Section 6.1. More work in that direction could practically reduce the complexity of the approach. Finally, a good diagnostics method based on the spectral reconstruction could be developed as well.

Appendix A

Check Matrix H_R—Auxiliary Lemma and Preliminaries

We prove in this appendix the error correcting properties of Arithmetic Transform and matrix H_r, introduced in Section 4.2. The check matrix H_r is obtained from the inverse AT transform matrix by selecting the evaluation points belonging to the $r+1$ top lattice layers, where the inverse is:

$$T_n^{-1} = \begin{bmatrix} T_{n-1}^{-1} & 0 \\ T_{n-1}^{-1} & T_{n-1}^{-1} \end{bmatrix}.$$

By multiplying $T * T^{-1} = I$, one can verify the correctness of the definition of T^{-1}.

We first consider an example of $n = 3$ and two top layers, i.e., the points taken are 111, 011, 101, and 110. The check matrix H_1 is then obtained from T_3^{-1} as:

$$H_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

By considering columns 4, 6, 7, and 8, we observe that the maximal number of independent columns, i.e., *matrix rank*, is 4. Notice that not every four columns are independent, such as columns 5, 6, 7, and 8. However, any three columns are independent, thus the *minimum number of independent columns* is three. The structure of the matrix is such [26] that the minimum number can be found by inspecting the last (tailing) three columns, i.e., 5, 6, and 7.

By taking one more, $r + 2\text{th}$, lattice layer, the corresponding rank of H_{r+1} is always equal to the number of rows, whereas the minimal number of independent columns is claimed to be $2^{r+2} - 1$. This statement can be inspected by considering the $2^{r+2} - 1$ columns, corresponding to the points considered. For example, adding one more layer to H_1 results in a matrix:

$$H_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Here, the minimal number of independent columns is seven. In general, this number increases every time a new layer is added and is equal to the largest subspace contained in the tailing columns. The fact that the diagonal entries of the matrix H_r are one ensures that the tailing columns are sufficient to consider as the columns ahead cannot be linearly dependent on any tailing column. By inspecting H_1 and H_2, one can notice that the number of tailing columns considered is 2^{k+1}, for k layers, and that the first of these 2^{k+1} tailing columns is dependent on the rest. Hence, when a layer is added to H_k, a new 2^{k+1} vectors are added to the independent set of columns. We are now ready to prove the lemma independent of the number of variables n.

Lemma 2. *Matrix H_r has at least $2^{r+1} - 1$ independent columns.*

Proof. By induction.

Base step: we showed that, for $k = 2$, the minimal number of independent columns is $2^{k+1} - 1$.

Induction step: for $k + 1$ layers, the minimal number of independent columns is equal to those for k layers, to which new 2^{k+1} columns are added. Then, the number of independent columns is:

$$2^{k+a} - 1 + 2^{k+1} = 2^{k+2} - 1 = 2^{(k+1)+1} - 1.$$

□

Appendix B

Auxiliary Results in Relating AT and WHT

Lemma 3. *The following relations hold:*

1. *The sum of column entries of $A_n = T_n^{\phi} - T_n^{AT}$ is $2^n - 1$ for the $(2^n - 1)\text{th}$ column, and 0 otherwise.*
2. *The sum of column entries of $B_n = T_n^{\phi} + T_n^{AT}$ is $2^n + 1$ for the $(2^n - 1)\text{th}$ column, and 0 otherwise.*

Proof.

1. The sum of columns in A_n is generated by observing (as from matrix A_3, without loss of generality) that the number of positive and negative entries are the same in all columns,

except in the last one, where there are $2^n - 1$ ones in the column.

2. The sum of columns in B_n is obtained by using Case 1 and the identity:

$$B_n = A_n + 2T_n^{AT}.$$

Then, the sums of the elements in the columns can be calculated by summing the matrices A_n and B_n matrices independently. Observe from T_3^{AT} that the sum of columns is 1 for the last column and 0 otherwise. The sum of the entries in the last column of B_n is $2^n - 1 + 2 * 1 = 2^n + 1$. □

Theorem 2. *The sum $S(T^\phi)$ of all spectral coefficients of Walsh-Hadamard Transform T_n^ϕ is always 2^n times larger than the sum of all Arithmetic Transform coefficients.*

Proof. Using Lemma 3, just proven, the sum $S(A_n)$, where $A_n = T_n^\phi - T_n^{AT}$, reduces to:

$$S(A_n) = \sum_{j=0}^{2^n-1} f_j \sum_{i=0}^{2^n-1} (T_{ij}^\phi - T_{ij}^{AT}) = \sum_{j=0}^{2^n-2} f_j \sum_{i=0}^{2^n-1} (T_{ij}^\phi - T_{ij}^{AT}) +$$
$$f_{2^n-1} \sum_{i=0}^{2^n-1} (T_{ij}^\phi - T_{ij}^{AT}) = 0 + (2^n - 1) * f_{2^n-1}.$$

Likewise, $S(B_n) = (2^n + 1) * f_{2^n-1}$ and the following relations hold between $S(A_n)$ and $S(B_n)$ (subscripts are the matrix entry indices):

$$\sum_{i=0}^{2^n-1} (c_i^\phi - c_i^{AT}) = \sum_{i=0}^{2^n-1} \sum_{i=0}^{2^n-1} A_{ij} f_j = (2^n - 1) * f_{2^n-1}$$
$$\sum_{i=0}^{2^n-1} (c_i^\phi + c_i^{AT}) = \sum_{i=0}^{2^n-1} \sum_{i=0}^{2^n-1} B_{ij} f_j = (2^n + 1) * f_{2^n-1},$$

which can be solved as a system of linear equations in sums of c_i^ϕs and c_i^{AT}s:

$$\sum_{i=0}^{2^n-1} c_i^\phi = 2^n * f_{2^n-1} \quad \text{and} \quad \sum_{i=0}^{2^n-1} c_i^{AT} = f_{2^n-1}$$

and, finally, the sums of spectral coefficients relate as:

$$\sum_{i=0}^{2^n-1} c_i^\phi = 2^n * \sum_{i=0}^{2^n-1} c_i^{AT}.$$

□

REFERENCES

[1] E.J. Aas, K. Klingsheim, and T. Steen, "Quantifying Design Quality: A Model and Design Experiments," *Proc. EURO-ASIC,* pp. 172-177, 1992.

[2] M.S. Abadir, J. Ferguson, and T. Kirkland, "Logic Verification via Test Generation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 7, no. 1, pp. 138-148, Jan. 1988.

[3] H. Al-Assad and J.P. Hayes, "Design Verification via Simulation and Automatic Test Pattern Generation," *Proc. Int'l Conf. Computer-Aided Design,* pp. 174-180, 1995.

[4] D. Brand, "Verification of Large Synthesized Designs," *Proc. Int'l Conf. Computer-Aided Design,* pp. 534-537, 1993.

[5] R. Bryant, "On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Applications to Integer Multiplication," *IEEE Trans. Computers,* vol. 40, no. 2, pp. 205-213, Feb. 1991.

[6] R.E. Bryant and Y.-A. Chen, "Verification of Arithmetic Functions with Binary Moment Diagrams," *Proc. 32nd Design Automation Conf.,* pp. 535-541, 1995.

[7] J.R. Burch and V. Singhal, "Tight Integration of Combinational Verification Methods," *Proc. Int'l Conf. Computer-Aided Design,* pp. 570-576, 1998.

[8] D. van Campenhout, H. Al-Asaad, J.P. Hayes, T. Mudge, and R.B. Brown, "High-Level Design Verification of Microprocessors via Error Modeling," *ACM Trans. Design Automation of Electronic Systems,* vol. 3, no. 4, pp. 581-599, Oct. 1998.

[9] D. van Campenhout, T. Mudge, and J.P. Hayes, "Collection and Analysis of Microprocessor Design Errors," *IEEE Design and Test of Computers,* vol. 33, no. 4, pp. 51-60, Oct.-Dec. 2000.

[10] K.T. Cheng, S. Dey, M. Rodgers, and K. Roy, "Test Challenges for Deep Sub-Micron Technologies," *Proc. Design Automation Conf.,* pp. 142-149, 2000.

[11] T. Damarla and M. Karpovsky, "Fault Detection in Combinational Networks by Reed-Muller Transform," *IEEE Trans. Computers,* vol. 38, no. 6, pp. 788-797, June 1989.

[12] T. Damarla, "Generalized Transforms for Multiple Valued Circuits and Their Fault Detection," *IEEE Trans. Computers,* vol. 41, no. 9, pp. 1101-1109, Sept. 1992.

[13] B.J. Falkowski, "A Note on the Polynomial Form of Boolean Functions and Related Topics," *IEEE Trans. Computers,* vol. 48, no. 8, pp. 860-864, Aug. 1999.

[14] G. Gupta and N. Jha, "A Universal Test Set for CMOS Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 7, no. 5, pp. 590-597, May 1988.

[15] S.Y. Huang and K.T. Cheng, *Formal Equivalence Checking and Design Debugging.* Kluwer Academic, 1998.

[16] S. Hurst, D.M. Miller, and J.C. Muzio, *Spectral Techniques in Digital Logic.* London: Academic Press, 1985.

[17] M.A. Iyer and M. Abramovici, "FIRE: A Fault-Independent Combinational Redundancy Identification Algorithm," *IEEE Trans. VLSI Systems,* vol. 4, no. 2, pp. 295-301, June 1996.

[18] U. Kalay, D. Hall, and M. Perkowski, "A Minimal Universal Test Set for Self-Test of EXOR-Sum-of-Products Circuits," *IEEE Trans. Computers,* vol. 49, no. 3, pp. 267-276, Mar. 2000.

[19] H. Kim and J.P. Hayes, "Realization-Independent ATPG for Designs with Unimplemented Blocks," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 20, no. 2, pp. 290-306, Feb. 2001.

[20] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 11, no. 1, pp. 4-15, Jan. 1992.

[21] N. Linial, Y. Mansour, and N. Nisan, "Constant Depth Circuits, Fourier Transform and Learnability," *J. ACM,* vol. 40, no. 3, pp. 607-620, July 1993.

[22] R. Mukherjee, J. Jain, K. Takayama, M. Fujita, J.A. Abraham, and D.S. Fussell, "An Efficient Filter-Based Approach for Combinational Verification," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 18, no. 11, pp. 1542-1557, Nov. 1999.

[23] K. Radecka and Z. Zilic, "Using Arithmetic Transform for Verification of Datapath Circuits via Error Modeling," *Proc VLSI Test Symp.,* pp. 271-277, May 2000.

[24] K. Radecka and Z. Zilic, "Identifying Redundant Wires for Synthesis and Verification," *Proc. IEEE Asian Design Automation Conf. (ASP-DAC),* pp. 517-523, Jan. 2002.

[25] K. Radecka and Z. Zilic, "Identifying Redundant Gate Replacements in Verification by Error Modeling," *Proc. IEEE Int'l Test Conf. (ITC),* pp. 803-812, Oct. 2001.

[26] R.M. Roth and G.M. Benedek, "Interpolation and Approximation of Sparse Multivariate Polynomials over GF(2)," *SIAM J. Computing,* vol. 20, no. 2, pp. 291-314, Apr. 1991.

[27] H. Savoj and R. Brayton, "The Use of Observability and External Don't Cares for the Simplification of Multi-Level Logic Networks," *Proc. Int'l Conf. Computer-Aided Design,* pp. 297-301, 1990.

[28] C. Scholl, B. Becker, and T.M. Weis, "Word-Level Decision Diagrams, WLCDs and Division," *Proc. IEEE Int'l Conf. Computer-Aided Design,* pp. 672-677, Nov. 1998.

[29] S.W. Tung and J.Y. Jou, "Verification Pattern Generation for Core-Based Design Using Port Order Fault Model," *Proc. Asian Test Symp.,* pp. 402-407, 1998.
[30] P. Vado, Y. Savaria, Y. Zoccarato, and C. Robach, "A Methodology for Validating Digital Circuits with Mutation Testing," *Proc. Int'l Symp. Circuits and Systems,* pp. 343-346, May 2000.
[31] A. Veneris and M. Abadir, "Design Error Diagnosis and Correction via Test Vector Simulation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 18, no. 12, pp. 1803-1816, 1999.
[32] Z. Zilic and Z.G. Vranesic, "A Deterministic Multivariate Interpolation Algorithm for Small Finite Fields," *IEEE Trans. Computers,* vol. 51, no. 9, pp. 1100-1105, Sept. 2002.

Katarzyna Radecka received the PhD degree from McGill University, Montreal, Canada, in 2003. She received the MEng degree and BEng degree (Honors) from the same university in 1997 and 1995. From 1996 to 1998, she worked for Lucent Technologies in Allentown, Pennsylvania. Currently, she is an assistant professor at Concordia University in Montreal, Canada. Her interests include verification and testing of digital hardware, as well as reversible and quantum computing. She coauthored the book *Verification by Error Modeling* (Kluwer, 2003). She is a member of the IEEE.

Zeljko Zilic received the PhD degree from the University of Toronto in 1997. From 1997 to 1998, he worked at Lucent Technologies. Currently, he is an assistant professor at McGill University, Montreal, Canada, and the director of McGill's Microelectronics and Computer Systems Laboratory. He holds a chercheur strategique research chair from the Province of Quebec. He received the Myril B. Reed Best Paper Award from the IEEE International Midwest Symposium on Circuits and Systems in 2001 for work on substrate coupling and its suppression. He holds four patents in the area of clock and power management. He coauthored the book *Verification by Error Modeling* (Kluwer, 2003). He is a member of the IEEE.