VLSI Design, 2002 Vol. 14 (1), pp. 83–105

Taylor & Francis
Taylor & Francis Group

Spectral Testing of Digital Circuits

BOGDAN J. FALKOWSKI

School of Electrical and Electronic Engineering, Nanyang Technological University, Block S1, Nanyang Avenue, Singapore 639798, Singapore

(Received 20 January 2000; In final form 4 October 2000)

Fault detection techniques using data compression methods have evolved during the last few years. Considerable work using individual Walsh spectral coefficients has been reported. In this paper, the application of spectral methods in testing of digital circuits with the emphasis on their usage for both input and output test compaction of digital circuits is described. Two closely related testing methods are discussed: syndrome testing and spectral testing as well as an overview of syndrome-testing and syndrome-testable design is presented. The necessary background and notation on Walsh spectral coefficients as well as their meaning in classical logic terms is shown.

Keywords: Logic design; Boolean functions; Walsh transform; Walsh spectrum; Syndrome testing; Spectral testing

1. INTRODUCTION

Spectral techniques based on Walsh, Haar, Arithmetic and Reed–Muller transforms in digital logic design have been used for more than 30 years. These techniques have been used for Boolean function classification, disjoint decomposition, parallel and serial linear decomposition, spectral translation synthesis (extraction of linear pre- and post-filters), multiplexer synthesis, threshold logic synthesis, state assignment and testing and evaluation of logic complexity [5,6,14,15,25,26,30,32–38,40,41,44,45,49, 53,56,62–66,81,82,89]. Spectral transforms have many attractive features and have found applications in many fields outside spectral theory of Boolean functions. For example, they have been widely used for data transmission—especially in the theory of error—correcting codes [41]. Another area of usage is signal processing, especially image compression, image processing and pattern analysis [5,6,41,60,81]. Spectral techniques have also been used for digital filtering and play the central role in all Digital Signal Processing applications [5,6,41,81]. Spectral methods for testing of logical network by verification of the coefficients in the spectrum have been developed [3,7,36,38,41,44,62–66,82,89]. The problem of constructing optimal data compression schemes by spectral techniques has also been considered [33,37,38,41,44,55]. The last approach is very useful for compressing test responses of logical networks and memories [38,41].

The renewed interest in applications of spectral methods in design of VLSI digital circuits is caused by their excellent design for testability properties [9,29,49,56,59,87] and the recent development of efficient methods that allow to calculate and operate on spectra of Boolean functions directly from reduced representations of such functions in the form of arrays of disjoint cubes or decision diagrams [12,13,16,20,22–24,27,28,42,50, 68,83]. Following from these recent developments, Hansen and Sekine [30] proposed a technique based on spectral analysis to decompose a circuit into a cascade of two sub circuits: linear block composed of exclusive or gates fed by the primary input of the overall circuit. Such linear decomposition, discussed in Ref. [40], drastically simplifies the synthesis task. The choice of a suitable linear transformation is based on a complexity measure assigned to each Boolean function that is heuristically related to the complexity of the final circuit implementation. To avoid the experimental costs of computing the Walsh transform, Binary Decision Diagrams (BDDs) based techniques have been used for that purpose in Ref. [30]. Due to the fact that many multilevel synthesis systems performs poorly for not good sum of products representations of the target functions, the authors in Ref. [30] proposed to use the complexity measure from Refs. [35,36] for multilevel logic synthesis. In the recent article [47], the idea of linear transformations is applied to decision diagrams. By combining powerful spectral linear techniques with variable reordering techniques the authors in Ref. [47] are able to synthesize large Boolean functions with standard Computer-Aided Design tools that fail otherwise. The experimental results in Ref. [47] show that the combination of standard variable reordering algorithm

ISSN 1065-514X print/ISSN 1563-5171 online © 2002 Taylor & Francis Ltd
DOI: 10.1080/10655140290009828

together with spectral linear transformations reduces the combined size of BDD with an average improvement of 22% in the number of nodes over a huge set of experiments.

The two major goals of testing are fault detection and fault location [8–11,29,38,39,41,43,49,67,86,87,89]. A set of input signals designated to detect or locate some faults of interest is called a *test vector*. A test is a sequence of one or more test vectors (patterns). The complexity of the test depends on the complexity of the test vectors and it increases fast with the size of the circuit. The problems of the complexity of tests of digital circuits were treated in detail in Ref. [39]. The same reference has a long list of bibliography of the classical approach to testing of logical circuit and the author refers the interested readers to this source of information. In short, the classical methods employed for test pattern generation are usually classified into three types: algorithmic, random and exhaustive [3,29,38,73,89]. The output response of the network under test (NUT) can be undertaken in two different ways. The output of an NUT can be compared with a known good circuit or for more complex designs the output of the NUT is compared with expected digital outputs. Algorithmic test generation employs analytic procedures to derive test vector sets of minimum or close to minimum size. The most commonly used circuit models are the gate-level models and the most common models of faults are single line stuck-at 0 or 1. The best known of these algorithms are the D-algorithm, PODEM and FAN [10,11,29,43,67,86,89]. These algorithms are based on the idea of computing an input pattern that enables an error signal generated due to single line stuck at fault to propagate from the faulty line to some observable line via some paths in the circuit. Random test generation produces test sets consisting of pseudo-randomly chosen input patterns [3,44,72,73]. This method can significantly reduce the computation cost of generating tests. However, one needs to know the number of tests necessary to provide an adequate degree of confidence that the circuit is fault-free. Such a number can be very large and it can be determined only after extensive analysis of the circuit behavior. Exhaustive testing consists of applying all possible input vectors to a combinational circuit and comparing the expected responses with the expected ones. It is obvious that in this approach the tests grow exponentially in size with the number of input lines to a circuit and hence truly exhaustive testing is possible only for small circuits with up to 20 input lines. Test generation is usually considerable more difficult for sequential circuits than for combinational circuits due to the presence of feedback loops, as well as memory elements that are not directly observable or controllable. Observability is a measure of how easily the state of a given point within a circuit can be determined from the signals available at the accessible inputs. Similarly, the controllability is a measure of how easily a given point within a circuit can be set to logic 1 or 0 by signals applied to the accessible inputs. A test for a fault in a sequential circuit consists of a sequence of inputs rather than a single input combination. Furthermore, the response of a sequential circuit to a given sequence of input patterns is also a function of its initial state. A number of design-for-testability (DFT) techniques [9,29,43,67,86,87,89] have been developed to help in testing of combinational and sequential logic circuits. They can be broadly classified into two categories: DFT to facilitate testing with external testers, and DFT to allow built-in-self-test (BIST). In this presentation we discuss only testing methods based on spectral coefficients for combinational circuits.

A number of authors have noticed that even though the total number of binary inputs in a system can be large, typical combinational outputs often depend on the restricted subsets of input vectors. Moreover, in many cases individual or local circuit outputs depend upon a disjoint set of input lines. Hence, partitioning of the overall system at the design stage may be important in test vector simplification. Several system outputs can also be compacted into fewer test output lines. Various data compaction methods have been proposed to reduce the response vector to the manageable size [38,39].

This article is a survey of developments in the usage of Walsh spectral coefficients for testing of digital circuits. The methods based on spectral coefficient testing require exhaustive testing, and the responses from the test are accumulated in a compact form. Bennets, for the first time applied spectral methods for testing of digital circuits in Ref. [7]. This presentation confines itself to the usage of only Walsh spectral coefficients in testing. It should, however, be noticed that there have already been attempts to use Reed–Muller transform [22,59,60,68] to testing as well [14]. Other used transforms include Arithmetic transform [21,24–27,68] used for testing in its basic one polarity form in [32], and Haar transform [40,41] used for testing in [62–66]. The testing methods based on the autocorrelation function that are also related to the spectral approach presented in this article have also been developed [1,2,17,18]. The testing of digital circuits can also use the extension of Arithmetic transform known under the name of Linearly Independent Arithmetic Transform [57,58].

Since the testing methods based on Walsh spectrum need the efficient way of the calculation of spectral coefficients, new methods allowing obtaining the spectral coefficients from reduced representations of Boolean functions have been introduced [13,20,23,50,68,83]. The approach based on the disjoint cube representation of Boolean function [20,23] is a general method and applies to the calculation of other transforms such as Reed–Muller [22] and Arithmetic [27]. In short, the basic advantages of methods to calculate spectral coefficients from disjoint cubes in comparison to the Fast Walsh Transform [5,6] are as follows:

- Each logical function is represented by a set of disjoint cubes that completely covers the function—therefore it is not necessary to store in the memory the values for

false minterms of the function what is required in Fast Transform.
- Each spectral coefficient can be calculated separately, order by order or all the coefficients can be calculated in parallel—it is impossible in all other methods of the calculation of Walsh coefficients.
- The algorithm operates not only on completely specified logical functions but also on incompletely specified ones.

Since this survey deals with spectral methods based on Walsh transform used for testing, the necessary background information on Walsh transforms is presented. In addition to classical approach to spectral methods for completely specified Boolean functions discussed in Refs. [5,34,36,38,40,41], this paper extend the previous results for incompletely specified Boolean functions based on Ref. [23]. By investigating links between spectral techniques and classical logic design methods this interesting area of research is presented in a simple manner. The real meaning of spectral coefficients in classical logic terms is shown. Moreover, an algorithm is shown for easy calculation of spectral coefficients for completely and incompletely specified Boolean functions by handwriting manipulations directly from Karnaugh maps. All mathematical relationships between the numbers of true, false, don't care minterms and spectral coefficients are stated. A number of examples as well as avoidance of usage of complicated mathematical formulas, so typical for articles in this subject should introduce these ideas to engineers working in the areas of test generation and logic design automation. It is a very important problem, since unfortunately up to now the unfamiliarity with the mathematical side of spectral approach seems to be too great hurdles to over-come to find a fruitful place for practical applications of these ideas.

Fault detection techniques using data compression methods have evolved during the last years [38,61]. Two of these methods involve *syndrome-testing* and *spectral coefficient testing*, which are closely related to each other. Considerable work using individual spectral coefficients has been reported [33,36,44,49,69–71,82]. The usage of all spectral coefficients requires exhaustive testing when responses of the circuit are accumulated in a compact form. However, such an approach is uneconomical for some circuits and rather a subset of coefficients, ideally one, is sought [49].

The *syndrome* introduced by Savir [69,70] is just one particular coefficient from the Walsh spectrum. For single stuck-at faults and when the circuit is *syndrome-testable*, then this method gives 100% fault coverage. When the circuit is *not syndrome-testable* then either it can be modified by hardware [87,46] or one can use the subset of spectral coefficients or *autocorrelation testing* [1,2,18,19,36,41,54]. The alternative solution to the not syndrome-testable circuits is to constrain a subset of the set of the input patterns and carry out a syndrome test on the remainder [71]. An alternative to the syndrome testing is a count of the number of transitions from 0 to 1 (or vice verse) in the output bit stream [31]. Like the syndrome testing, this transition count test normally considers a fully exhaustive input test set, however, unlike the syndrome testing, a transition count is dependent on the order of application. Hence, the exhaustive input test set can be ordered so that the number of transitions in the output response is minimized [10,31,74,89].

A set of *constrained syndromes* has been found to cover any single output digital circuit against single *stuck-at faults* [71]. However, in the presentation of this problem in the Boolean domain all untestable lines of the circuit have to be *unate* (an unate line is the line that corresponds to an unate Boolean function—see Definitions in the "Links between spectral techniques and classical logic design" section). When the same problem is presented in the spectral domain then the last requirement is not necessary and the sufficient conditions are derived in the spectral domain in a more easy way [36].

Recent researches in VLSI testing have emphasized partitioning of large networks and exhaustive testing of the partitions, rather than the previous concept of global testing of the whole circuit [4,38]. Exhaustive testing has the following advantages: no fault models are assumed, no complex computations of minimal test sets under fault model constraints are necessary, less test-hardware is necessary to generate the test signals and record the network response. However, the exhaustive testing should be performed locally rather than globally in order to reduce the test time involved [37,38].

The methods for compacting the input set of test vectors applied to a system under test, and also the compaction of the output response from multiple outputs have been developed in the spectral domain [38,49,75–77]. The area where the spectral methods show the greatest promise is for *multiple-output networks*. When applied to Programmable Logic Arrays (PLAs), this method gives 100% fault coverage against single *stuck-at faults*, all *single bridging faults*, and all *contact faults*. In case of *multiple faults* it gives a reasonable coverage against them [75–77]. The detailed description of all terms used in this section follows in the sequel.

2. WALSH SPECTRUM

There have been many papers and books published on the use of spectral methods in the design of digital logic circuits which require calculating of forward or inverse transforms [5,6,34–38,40,41,49–56]. All these methods use either minterms, or minimized sum-of-products expressions (SOPE) as the input data.

The Walsh spectrum R of an n-variable combinational logic function F is an alternative representation of this function. When the function F is represented as a truth table of all minterms, then the Walsh spectrum R is formed from the multiplication of the $\langle 0, 1\rangle$ vector representation

$$
\begin{array}{cccc}
 & \mathbf{T} & \mathbf{M} & \mathbf{R} \\
\left[\begin{array}{cccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1
\end{array}\right]
\left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{array}\right]
=
\left[\begin{array}{cc} 7 & \mathrm{R}_0 \\ -1 & \mathrm{R}_1 \\ -1 & \mathrm{R}_2 \\ -1 & \mathrm{R}_{12} \\ -1 & \mathrm{R}_3 \\ 3 & \mathrm{R}_{13} \\ 3 & \mathrm{R}_{23} \\ -1 & \mathrm{R}_{123} \\ -3 & \mathrm{R}_4 \\ -3 & \mathrm{R}_{14} \\ 1 & \mathrm{R}_{24} \\ 1 & \mathrm{R}_{124} \\ -3 & \mathrm{R}_{34} \\ 1 & \mathrm{R}_{134} \\ -3 & \mathrm{R}_{234} \\ 1 & \mathrm{R}_{1234} \end{array}\right]
\end{array}
$$

FIGURE 1 Spectrum R of a four variable completely specified Boolean function.

of the truth table for a completely specified Boolean function, or from the multiplication of the $\langle 0, 0.5, 1\rangle$ vector representation of the truth table for an incompletely specified Boolean function by a $2^n \times 2^n$ Walsh matrix T [5,6,34–36,38,40,41]. In the R coding scheme, the conventional $\langle 0, 1, -\rangle$ values correspond to $\langle 0, 1, 0.5\rangle$ codings, respectively ($-$ stands for a don't care) and the coded vector is denoted by M.

The principal properties of the spectral coefficients from the spectrum R and the different variants of Walsh type transform matrices are as follows [5,20,23,34,36,40,41].

2.1 The transform matrix is complete and orthogonal, and therefore, there is no information lost in the spectrum R, concerning the minterms in a Boolean function F.

2.2 Only the Hadamard—Walsh matrix has the *recursive Kronecker product structure* [5,6,36,40] and for this reason is preferred over other possible variants of Walsh transform known in the literature as Walsh–Kaczmarz, Rademacher–Walsh, and Walsh–Paley transforms.

2.3 Only the Rademacher–Walsh transform is not *symmetric*; all other variants of Walsh transform are symmetric, so that, disregarding a scaling factor, the same matrix can be used for both the forward and inverse transform operations.

2.4 When the classical matrix multiplication method is used to generate the spectral coefficients for different Walsh transforms, then the only difference is the order in which particular coefficients are created. The values of all these coefficients are the same for every Walsh transform.

2.5 Each spectral coefficient r_I gives a *correlation value* between the function F and a *standard trivial function* u_I corresponding to this coefficient. The standard trivial functions for the spectral coefficients are, respectively, for the coefficient r_0 (dc coefficient)—the universe of the Boolean function denoted by u_o, for the coefficient r_i (first order coefficients)—the variable x_i of the Boolean function denoted by u_i, for the coefficient r_{ij} (second order coefficients)—the exclusive-or function between variables x_i and x_j of the Boolean function denoted by u_{ij}, for the coefficient r_{ijk} (third order coefficients)—the exclusive-or function between variables x_i, x_j, and x_k of the Boolean function denoted by u_{ijk} etc.

2.6 The maximum value of any but r_0 individual spectral coefficient r_I is $\pm 2^{n-1}$. This happens when the completely specified Boolean function is equal to either a standard trivial function u_I (sign $-$) or to its complement (sign $+$). In either case, all but r_0 remaining spectral coefficients have zero values because of the orthogonality of the transform matrix T.

2.7 The maximum value of r_0 spectral coefficient is 2^n. It happens when all the minterms of the Boolean function F are true (i.e. for tautology).

2.8 Each but u_0 standard trivial function u_I corresponding to n variable Boolean function has the same number of true and false minterms equal to 2^{n-1}.

2.9 When r_0 is odd for a given Boolean function then all other r_I coefficients are odd for this function. When

$$
\begin{array}{c}
\mathbf{T} \\
\left[\begin{array}{cccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1
\end{array}\right]
\end{array}
\begin{array}{c}
\mathbf{M} \\
\left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0.5 \\ 0.5 \end{array}\right]
\end{array}
=
\begin{array}{cc}
\mathbf{R} & \\
\left[\begin{array}{c} 6 \\ -1 \\ 0 \\ -1 \\ 0 \\ 3 \\ 2 \\ -1 \\ -2 \\ -3 \\ 0 \\ 1 \\ -4 \\ 1 \\ -2 \\ 1 \end{array}\right] &
\begin{array}{l} \mathbf{R}_0 \\ \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{R}_{12} \\ \mathbf{R}_3 \\ \mathbf{R}_{13} \\ \mathbf{R}_{23} \\ \mathbf{R}_{123} \\ \mathbf{R}_4 \\ \mathbf{R}_{14} \\ \mathbf{R}_{24} \\ \mathbf{R}_{124} \\ \mathbf{R}_{34} \\ \mathbf{R}_{134} \\ \mathbf{R}_{234} \\ \mathbf{R}_{1234} \end{array}
\end{array}
$$

FIGURE 2 Spectrum R of a four variable incompletely specified Boolean function.

r_0 is even for a given Boolean function then all other r_I coefficients are even for this function.

Example 2.1 A method of the calculation of the spectrum R for a four variable Boolean function $F1$ is shown in Fig. 1. The same method is applied for the calculation of the spectrum R of a four variable incompletely specified Boolean function $F2$ that is shown in Fig. 2.

Recursive algorithms, data flow-graph methods and parallel calculations similar to Fast Fourier Transform have been also used to calculate Walsh and other related transforms [5,6,41,81]. A number of efficient methods to calculate various spectra of Boolean functions based on standard and spectral decision diagrams have also been reported [12,13,28,68].

3. LINKS BETWEEN SPECTRAL TECHNIQUES AND CLASSICAL LOGIC DESIGN

The results presented here are based on Refs. [34,36] for completely specified Boolean functions and on Ref. [23] for incompletely specified Boolean functions. Let us show more clearly in classical logic terms what is the real meaning of spectral coefficients. Moreover, let us expand our considerations for incompletely specified Boolean functions as well. The following symbols will be used. Let a_I, be the number of true minterms of Boolean function F, where both the function F and the standard trivial function u_I have the logical values 1; let b_I be the number of false minterms of Boolean function F, where the function F has the logical value 0 and the standard trivial function u_I has the logical value 1; let c_I be the number of true minterms of Boolean function F, where the function F has the logical value 1 and the standard trivial function u_I has the logical value 0; let d_I be the number of false minterms of Boolean function F, where both the function F and the standard trivial function u_I have the logical values 0; let e_I be the number of don't care minterms of Boolean function F, where the standard trivial function u_I has the logical value 1, and f_I be the number of don't care minterms of Boolean function F, where the standard trivial function u_I has the logical value 0. Then, for completely specified Boolean functions having n variables, these formulas hold (when $I \neq 0$):

$$a_I + b_I + e_I + d_I = 2^n \tag{1}$$

and

$$a_I + b_I = c_I + d_I = 2^{n-1} \tag{2}$$

For the r_0 spectral coefficient (when $I = 0$):

$$a_I + b_I = 2^n \tag{3}$$

Accordingly, for incompletely specified Boolean functions, having n variables, the following hold (when $I \neq 0$):

$$a_I + b_I + c_I + d_I + e_I + f_I = 2^n \tag{4}$$

and

$$a_I + b_I + e_I = c_I + d_I + f_I = 2^{n-1} \tag{5}$$

For the r_0 spectral coefficient (when $I = 0$):

$$a_I + b_I + e_I = 2^n \tag{6}$$

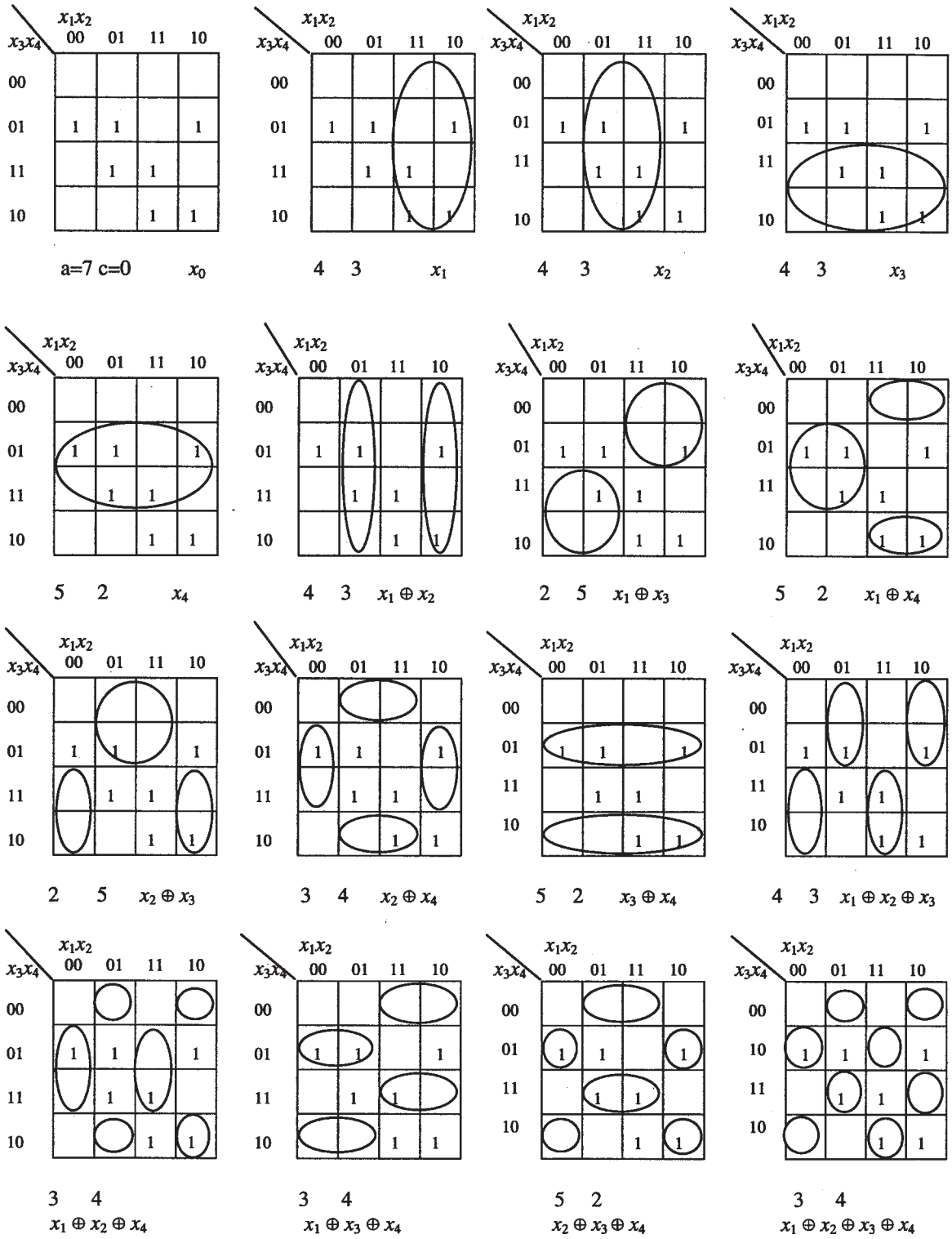

FIGURE 3 Karnaugh maps for Boolean function F1 and corresponding standard trivial functions.

The spectral coefficients for completely specified Boolean functions could be defined in the following way:

$$r_I = a_I, \quad I = 0 \tag{7}$$

and

$$r_I = c_I - a_I = 2^n - b_I - d_I, \quad I \neq 0. \tag{8}$$

Since each standard function has the same number of true and false minterms that is equal to 2^{n-1}, then we can have alternative definitions of spectral coefficients. Please note that only for the spectral coefficient r_0, is the above rule not valid and the appropriate standard function is the universe u_0, i.e. the logical function that is true for all its

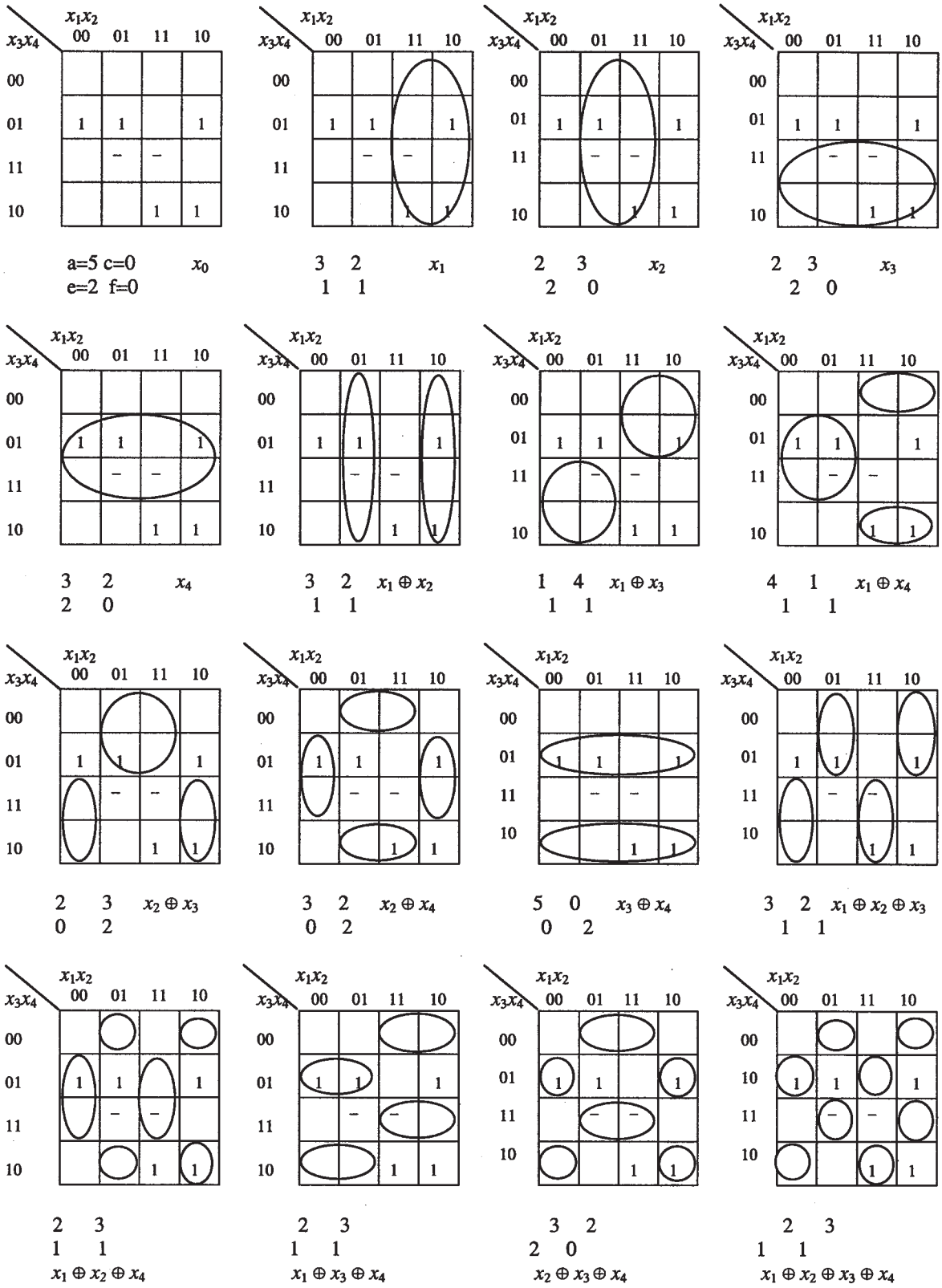

FIGURE 4 Karnaugh maps for Boolean function F2 and corresponding standard trivial functions.

minterms. Thus, we have that:

$$r_I = b_I - d_I, \quad I \neq 0, \tag{9}$$

and

$$r_I = 2^n - b_I, \quad I = 0. \tag{10}$$

The spectral coefficients for incompletely specified Boolean function, having n variables, can be defined in the following way:

$$r_I = a_I + \frac{e_I}{2}, \quad I = 0, \tag{11}$$

and

$$r_I = c_I - a_I + \frac{f_I - e_I}{2}, \quad I \neq 0. \tag{12}$$

As we can see, for the case when $e_I = 0$ and $f_I = 0$, i.e. for the completely specified Boolean function, Eqs. (11) and (12) reduce to Eqs. (7) and (8). And again, by easy mathematical transformations, we can define all but r_0 spectral coefficients in the following ways:

$$r_I = c_I + b_I + \frac{e_I + f_I}{2} - 2^{n-1}, \quad I \neq 0, \tag{13}$$

or

$$r_I = 2^{n-1} - \left[a_I + d_I + \frac{f_I + e_I}{2}\right], \quad I \neq 0, \tag{14}$$

or

$$r_I = b_I - d_I + \frac{e_I - f_I}{2}, \quad I \neq 0. \tag{15}$$

Simultaneously, the r_0 spectral coefficient can be rewritten in the following way:

$$r_I = 2^n - b_I - \frac{e_I}{2}, \quad I = 0 \tag{16}$$

or

$$r_I = \frac{a_I}{2} - \frac{b_I}{2} + 2^{n-1}, \quad I = 0. \tag{17}$$

Example 3.1 Consider the same two Boolean functions, for which spectral coefficients were calculated previously with the standard matrix multiplication method. The appropriate u_I, a_I, c_I, e_I and f_I are shown in Figs. 3 and 4.

The spectrum R for the function $F1$ can be calculated as follows:

$$r_0 = a_0 = 7, \quad r_I = c_I - a_I,$$

when $I \neq 0$.

$$r_1 = 3 - 4 = -1, \quad r_2 = 3 - 4 = -1,$$
$$r_3 = 3 - 4 = -1, \quad r_4 = 2 - 5 = -3,$$
$$r_{12} = 3 - 4 = -1, \quad r_{13} = 5 - 2 = 3,$$
$$r_{14} = 2 - 5 = -3, \quad r_{23} = 5 - 2 = 3,$$
$$r_{24} = 4 - 3 = 1, \quad r_{34} = 2 - 5 = -3,$$
$$r_{123} = 3 - 4 = -1, \quad r_{124} = 4 - 3 = 1,$$
$$r_{134} = 4 - 3 = 1, \quad r_{234} = 2 - 5 = -3,$$
$$r_{1234} = 4 - 3 = 1.$$

The spectrum R for the function $F2$ is:

$$r_0 = a_0 + \frac{e_0}{2} = 5 + 1 = 6, \quad r_I = c_I - a_I + \frac{f_I - e_I}{2},$$

when $I \neq 0$

$$r_1 = 2 - 3 = -1, \quad r_2 = 3 - 2 - 1 = 0,$$
$$r_3 = 3 - 2 - 1 = 0 \quad r_4 = 2 - 3 - 1 = -2,$$
$$r_{12} = 2 - 3 = -1, \quad r_{13} = 4 - 1 = 3,$$
$$r_{14} = 1 - 4 = -3, \quad r_{23} = 3 - 2 + 1 = 2,$$
$$r_{24} = 2 - 3 + 1 = 0, \quad r_{34} = -5 + 1 = -4,$$
$$r_{123} = 2 - 3 = -1, \quad r_{124} = 3 - 2 = 1,$$
$$r_{134} = 3 - 2 = 1, \quad r_{234} = 2 - 3 - 1 = -2,$$
$$r_{1234} = 3 - 2 = 1.$$

As we can find out from the above results, they are exactly the same as the results obtained by classical matrix multiplication method (Figs. 1 and 2).

4. SYNDROME, CONSTRAINED SYNDROME AND WEIGHTED SYNDROME SUMS TESTING

Savir [69,70] introduced the syndrome of a Boolean function.

DEFINITION 4.1 The *syndrome*, σ, of a Boolean function F is defined as

$$\sigma(F) = \frac{W(F)}{2^n} \tag{18}$$

where $W(F)$ is the number of true minterms of the Boolean function F, and n is a number of the variables of the Boolean function F.

The $W(F)$ is also called the *weight* of the Boolean function F.

DEFINITION 4.2 The circuit is called to be *syndrome-testable* when its performance can be verified by counting the number of true minterms of the Boolean function F.

DEFINITION 4.3 The faulty syndrome σ^f caused by a fault f is a syndrome that has a different value than a syndrome σ of faultless circuit.

The syndrome corresponds just to one particular coefficient (dc coefficient) from the Walsh spectrum R that can be verified in order to ensure the correct behavior of a digital circuit. The exact relationship between the syndrome $\sigma(F)$ and r_0 is as follows:

$$r_0 = 2^n \sigma(F) \tag{19}$$

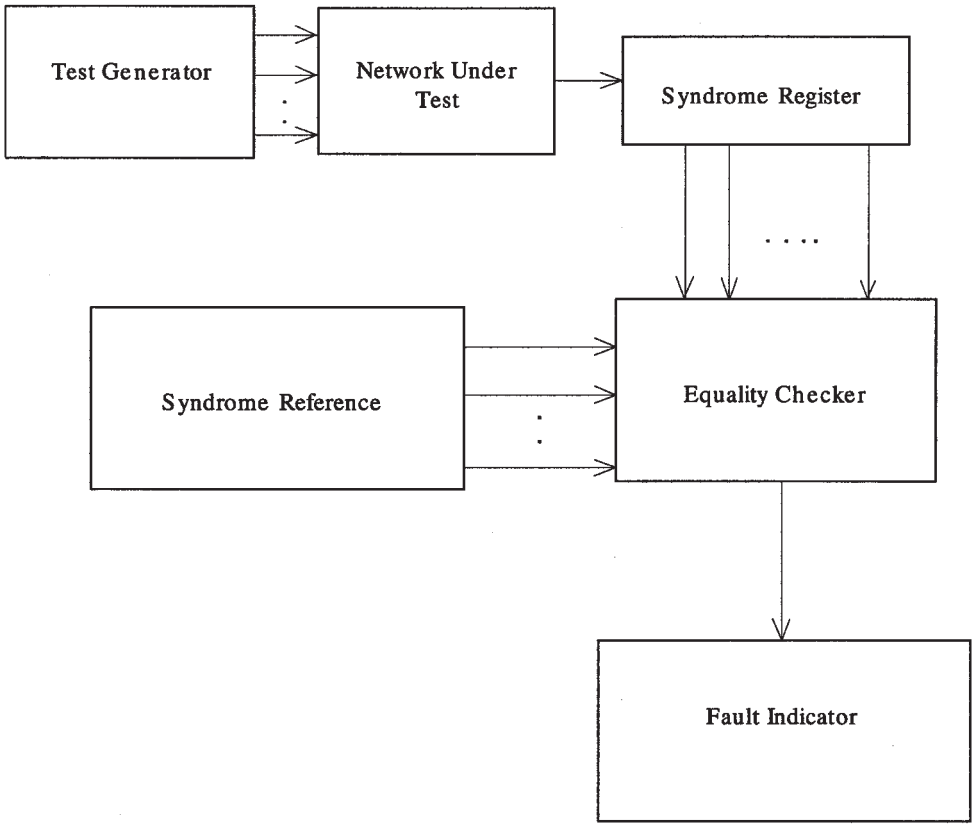

FIGURE 5 The test procedure.

Hence, it is obvious from the above relationship that the meaning of a syndrome-testable fault and r_0-testable fault is the same.

There are a number of topological classes of networks, which are used to characterize the syndrome-testability of circuits. These classes are important not only for syndrome-testability but also for general testability of circuits. The short descriptions of these classes of networks and properties of a Boolean function F are presented below and will be used in the sequel [15,16,36,42,69].

4.1 *Fan-out-free* network is a network in which there is a unique path from any line to the network output.

4.2 *Internally fan-out-free* network is a network where the fan-out restriction applies only to internal lines.

4.3 Boolean function F is said to be *positive* (*negative*) in the variable x_i, if there exist a sum-of-products expression or product-of-sums expression for it in which x_i appears only in uncomplemented (complemented) form.

4.4 Boolean function F is said to be *unate* in x_i, if it is either positive or negative in x_i.

4.5 *Unate* network is a network where every line is unate.

4.6 *Internally unate* network is a network where every internal line is unate.

4.7 *Two-level* network is a network that corresponds to the implementation of a sum-of-products or product-of-sums expression.

4.8 *Internally nonunate* network is a network with one or more nonunate internal lines.

4.9 Network is *redundant* if it contains some fault of the line-stuck line that is undetectable.

4.10 Two paths that emanate from a common line and reconverge at some forward point are said to be *reconverging paths.*

4.11 *Inversion parity* of a reconverging path is equal to the number of inversions modulo 2 in the path, i.e. the number of inversions modulo 2 between the point of divergence and the point of reconvergence.

4.12 *Stuck-at fault* exists when one line x_i is stuck at either logical 1 or 0 that is denoted by $x_i/1$ or $x_i/0$, accordingly.

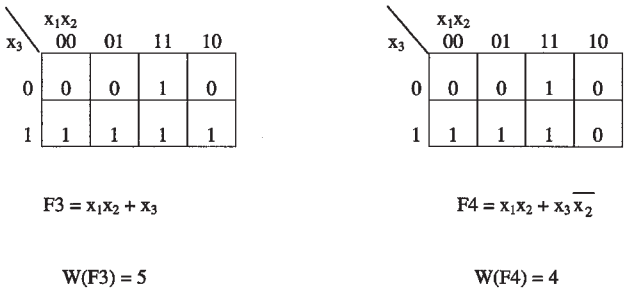

FIGURE 6 Three variable Boolean functions F3 and F4.

4.13 *Multiple-input stuck-at fault* exists when more than one line are stuck at either logical 1 and/or logical 0.

4.14 Multiple-input stuck-at fault is termed *unidirectional* if all stuck lines take the same value; otherwise it is termed *mixed.*

4.15 *Contact fault* exists when there is at least one spurious connection between two contact points in layout.

4.16 Two faults are *equivalent* if they have the same detecting tests.

The classes of faults that are considered for a syndrome-testable realization are the single stuck-at types [69,70]. However, many multiple faults are covered by this design as well. In the beginning two-level networks are considered in the form of sum-of-products. The results can be easily adopted for product-of-sums networks. Consequentially, the results are applicable to a very important class of PLAs [38].

The test procedure for the syndrome-testable circuit is shown in Fig. 5. An input generator provides each possible input combination to the NUT. The syndrome register counts the number of ones in the output data stream. The equality checker compares the syndrome's register content with the expected syndrome. If the syndromes are equal then the network is fault-free, otherwise, a fault is detected and the network is declared to be faulty.

Savir stated the following results concerning syndrome-testability [69,70].

Result 4.1: A two-level irredundant circuit, which realizes an unate function in all its variables, is syndrome-testable.

Result 4.2: There exist two-level irredundant circuits, which are not syndrome-testable.

Example 4.1 Consider the functions $F3$ and $F4$ shown in Fig. 6. Let us present all possible single stuck-at faults for the function $F3$. The weight $W(F_3) = 5$. If input x_1 is stuck-at 0 then the circuit realizes x_3 and the weight $W(x_3) = 4$. The same weight is obtained when input x_2 is stuck-at 0. If input x_1 is stuck-at 1 then the circuit realizes $x_2 + x_3$ and the weight $W(x_2 + x_3) = 6$. The same weight is obtained when input x_2 is stuck-at 1. When input x_3 is stuck-at 0 then the circuit realizes x_1x_2 and the weight $W(x_1x_2) = 2$. Finally, when input x_3 is stuck-at 1 then the circuit realizes the tautology (i.e. the function $F3$ is equal to 1 for all minterms) and its weight is equal to 8. Hence, the function $F3$ is syndrome-testable since only for the fault-free function $F3$ its weight is equal to 5. A similar analysis can be done for the function $F4$ as well. The weight $W(F_4) = 4$. Let us notice that when input x_2 is stuck-at 0 then the circuit realizes x_3 and the weight $W(x_3) = 4$. Thus, the function $F4$ is not syndrome-testable.

From the above discussion it should already be obvious that any two-level combinational network can be made syndrome-testable by increasing the size of the product with extra input addition. For example, the function *F4* can be made syndrome-testable by adding one extra input x_4 to create the new function $F4^* = x_1x_2x_4 + x_3\bar{x}_2$. During normal operation of the circuit the input x_4 is connected to logic 1 and for testing it is used as the valid input. Savir formulated the following result [69,70].

Result 4.3: Every two-level irredundant combinational network can be made syndrome-testable by attaching extra inputs to *AND* gates.

Savir [69,70] describes an algorithm for designing syndrome-testable two-level combinational networks. From Results 4.1 and 4.2, it is obvious that the candidates for *syndrome-untestability* are the nonunate input lines. In the algorithm, the syndrome-testable design is achieved by modifying the original irredundant sum of products. This modification requires an introduction of a nearly minimal number of control inputs.

Let us consider a syndrome-testable design of general combinational networks. Savir [69,70] formulated the following results.

Result 4.4: Every fan-out-free irredundant combinational network, composed of *AND*, *OR*, *NAND*, *NOR*, and *NOT* gates is syndrome-testable.

Let $F = Ag + B\bar{g} + C$ be the sum-of-products expression of the function F with respect to the line, $g = g(x_1, x_2, \ldots, x_n)$, where A, B, and C do not depend on g. Also, let $\sigma(F)$ denote the syndrome of the function F. Savir formulated the following results [69–71].

Result 4.5: The fault g stuck-at 0 (g/0) is syndrome-untestable when $\sigma(A\bar{C}g) = \sigma(B\bar{C}g)$.

Result 4.6: The fault g stuck-at 1 (g/1) is syndrome-untestable when $\sigma(A\overline{Cg}) = \sigma(B\overline{Cg})$.

Result 4.7: If the function F is unate in line g, the faults on g are syndrome-testable.

Markowsky [46] proved that one could always make a syndrome-untestable circuit to be syndrome-testable with the appropriate hardware modification. Hence, the following result can be stated.

Result 4.8: Every combinational circuit can be made syndrome-testable by attaching to it extra input pins and logic.

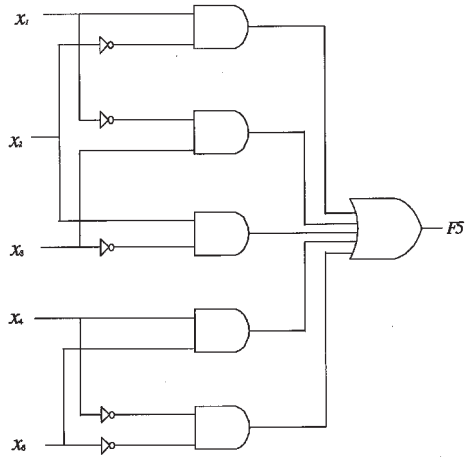

FIGURE 7 A five variable Boolean function F5.

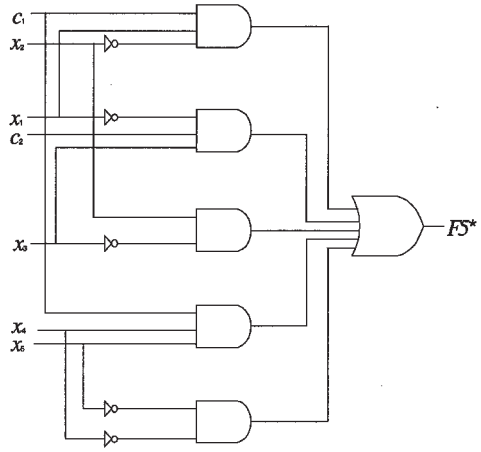

FIGURE 8 The syndrome-testable design for function F5.

Markowsky's idea is to embed the syndrome-untestable circuit into a larger, syndrome-testable circuit. However, there is no method leading to the solution in his work and only the existence proof is given. On the other hand, Savir [69,70] presented a heuristic method that does quite well in practice and produces a near-minimal hardware addition. However, this method has not been formally proved and requires a lot of computation.

For two-level sum-of-products networks (an analogous version can be formulated for product-of-sums expression) the method for testing inputs is as follows:

1. A control variable is added to one prime implicant of the function F, i.e. an input is added to one *AND* gate in the two-level implementation of the function F. Trying each in turn and choosing the input that changes the greatest number of previously syndrome-untestable inputs to syndrome-testable selects the prime implicant. When this number is the same for several prime implicants then an arbitrary choice is made.
2. During subsequent iterations, an attempt is made to add an existing control variable to another prime implicant in order to reduce further the number of untestable inputs.
3. In a case that step 2 fails to find any improvement, an additional control variable is added and the algorithm proceeds as in the step 1.
4. The algorithm continues iterations as long as all inputs are made syndrome-testable with respect to single stuck-at faults.

The algorithm always stops, however, the number of added control lines not necessarily has to be the minimal one. During each iteration, the method is finding a local minimum, but it does not lead in general to a global optimization.

Example of the execution of the algorithm from [69] follows.

EXAMPLE 4.2 Let us consider the function $F5 = x_1\bar{x}_2 + \bar{x}_1x_3 + x_2\bar{x}_3 + x_4x_5 + \overline{x_4x_5}$ shown in Fig. 7. It can be checked that a two-level realization of this function is syndrome-untestable for all single-input stuck-at faults.

The first pass of the algorithm produces the expression $c_1x_1\bar{x}_2 + \bar{x}_1x_3 + x_2\bar{x}_3 + x_4x_5 + \overline{x_4x_5}$ for which x_3, x_4, x_5 remain untestable. The second pass uses c_1 for the second time and yields $c_1x_1\bar{x}_2 + \bar{x}_1x_3 + x_2\bar{x}_3 + c_1x_4x_5 + \overline{x_4x_5}$, which has a single untestable input x_3. Finally, the last pass yields $F5^* = c_1x_1\bar{x}_2 + c_2\bar{x}_1x_3 + x_2\bar{x}_3 + c_1x_4x_5 + \overline{x_4x_5}$.

The two-level implementation of this expression is shown in Fig. 8 and is syndrome-testable for all single stuck-at faults.

Constrained syndrome testing proposed by Savir [71] is an alternative to hardware modification. The main advantage of this method is the fact that the network does not need to be changed in order to make it testable. Hence, no additional extra pins as the control inputs are required. The idea of this approach is to fix some subset of the inputs at constant values and exhaustively test the remaining inputs recording their syndrome.

Suppose that a circuit has n inputs and m of them are fixed during the test. Then it is obvious that the constrained test consists of running only 2^{n-m} input combinations what makes the execution of the test faster than the full one, but there is a certain portion of the network which cannot be tested by this constrained test.

EXAMPLE 4.3 Consider the function $F6$ from Fig. 9. This network is syndrome untestable for $g/0$ and $g/1$. The cause of the problem is the fact that there exists the possibility of a cancellation effect in the syndrome due to unequal inversion parity from point g to the output. Hence, a constrained syndrome test can be executed on g when these two paths are broken. When $x_1 = 0$ then g is syndrome-testable in $F6$ $(0,x_2,x_3,x_4)$. In fact, the function $F6$ is unate in g, and according to Result 4.7, it ensures that a line g is syndrome-testable in the constrained function. The method from [72] involves the choice of the variable

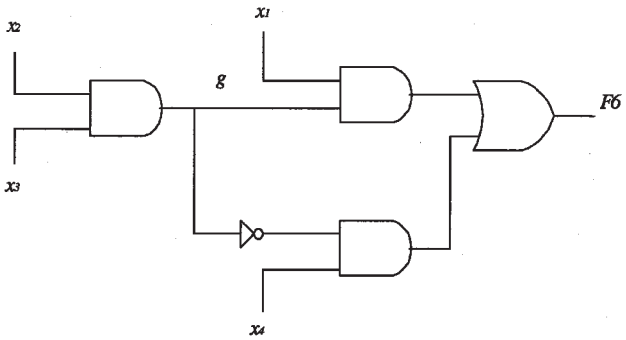

FIGURE 9 Function F6 and its disjoint decomposition

constraints to ensure that all syndrome-untestable lines become unate for one of the constrained tests. Although the condition of unateness is sufficient, it is not necessary. Hence, a weaker condition developed in spectral domain ensures that a line is a constrained syndrome-testable [36,56].

According to Ref. [4] it is always possible to use one weighted syndrome sum as a reference for multiple-output networks instead of separate syndromes for all outputs. This method of combining output syndromes allows reducing significantly the number of references needed for syndrome testing.

Let $\vec{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ be a vector representing the set of output syndromes of all outputs. Let also the arithmetic function

$$g = \sum_{i=1}^{m} w_i Z_i \tag{20}$$

be a linear combination of the variables $Z_1, \ldots, Z_m$ with some integer coefficients $w_1, w_2, \ldots, w_m$, where $i = 1, 2, 3, \ldots, m$.

DEFINITION 4.4 A *weighted syndrome sum* (WSS), for an m-output network with coefficients w_i, $i = 1, 2, 3, \ldots, m$ is defined as

$$\text{WSS} = g(\vec{\sigma}) = \sum_{i=1}^{m} w_i \sigma_i. \tag{21}$$

Hence, instead of the complete collection of syndromes for each output of the NUT, one or more WSS are to be used. However, even with the assumption that each output function is designed to be syndrome-testable, one should notice that when WSSs are used as the references then the undetectable faults could still remain. The next Definition deals with such a possibility.

DEFINITION 4.5 Let the faulty syndromes caused by a fault f be denoted by σ_i^f, $i = 1, 2, 3, \ldots, m$ and $\Delta\vec{\sigma} = \vec{\sigma} - \sigma^f$. Then the fault f is said to be *weighted sum syndrome untestable* (WSSU) iff

$$g(\Delta\vec{\sigma}) = \sum_{i=1}^{m} w_i \Delta\sigma_i = 0. \tag{22}$$

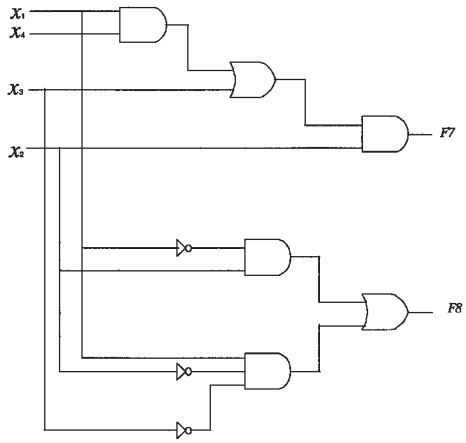

FIGURE 10 Four variable Boolean functions F7 and F8.

It has been proven in Ref. [4] that coefficients w_i can always be selected in such a way that every single stuck-at fault is WSS-testable. The following example explains the concept of WSSU.

Example 4.4 Consider the circuit of Fig. 10. Assume that the weighted sum with coefficients 2 and 3 has been chosen: $g = 2Z_1 + 3Z_2$.

Assume that the input line x_3 is stuck-at 0. Then the fault-free and faulty syndromes of the functions $F7$ and $F8$ are

$$\sigma_1 = \frac{5}{16}, \quad \sigma_1^f = \frac{1}{8}, \quad \sigma_2 = \frac{3}{8}, \quad \sigma_2^f = \frac{1}{2}.$$

Thus

$$\Delta\sigma_1 = \frac{5}{16} - \frac{1}{8} = \frac{3}{16}, \quad \Delta\sigma_2 = \frac{3}{8} - \frac{1}{2} = \frac{-1}{8},$$

and therefore, $g(\Delta\bar{\sigma}) = 2\Delta\sigma_1 + 3\Delta\sigma_3 = 0$.

According to Definition 4.5 the fault $x_3/0$ is WSSU.

It should, however, be noticed that as long as all w_i coefficients are nonzero a fault which affects only one output function can never yield the WSSU condition from Definition 4.5. It is due to the fact that each such fault would change the corresponding syndrome and therefore change the WSS as well.

5. SPECTRAL COEFFICIENTS TESTING

Results presented in this section are mainly from Ref. [48] although some are also reported in Refs. [36,38,41,44,49,85,78]. The first two presented results are the same as the ones in the previous section. Such an approach allows presenting all the developments in a coherent way and it is also much easier to understand the additional testing possibilities that are given by spectral coefficients in comparison to syndrome testing.

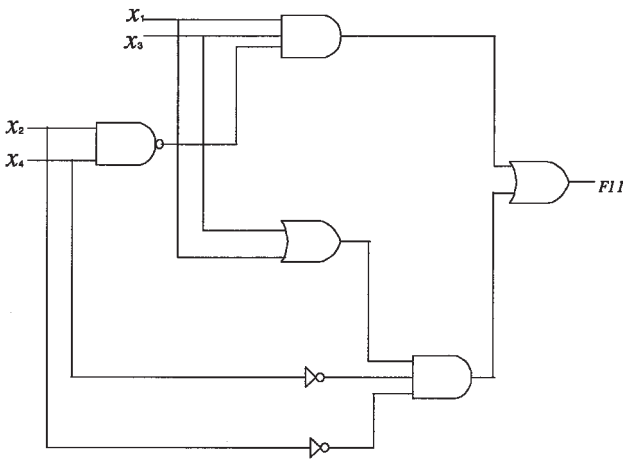

FIGURE 11 A four variable Boolean function F11.

Let F^f denote the n-variable Boolean function F in which one variable x_i is stuck-at 0 or stuck-at 1. The spectrum of the Boolean function F is denoted by R and that of F^f is denoted by R^f. Accordingly then, a symbol r_α will be used for a spectral coefficient of a fault-free circuit and r^f_α for the same coefficient from a circuit with a fault f. The following definitions of attributes and signatures of Boolean functions are taken from Ref. [85].

DEFINITION 5.1 An *attribute* $A(F)$ of a Boolean function F is some property, which partitions the set of all Boolean functions into at least two nonempty subsets.

In this presentation, all the attributes of Boolean functions are simply particular spectral coefficients.

Example 5.1 Consider two Boolean functions: $F9 = x_3x_1 + x_3x_2 + x_2x_1$ and $F10 = x_3x_1 + x_3x_2x_1$. Then, for $F9$ the values of $r_0 = 4$, and $r_1 = 2$, while for $F10$ the values of $r_0 = 3$, and $r_1 = -1$. Hence, both coefficients r_0 and r_1 are examples of attributes for these functions.

Assume now, that by N a fault-free circuit that realizes the Boolean function F is denoted. The N^f denotes the network in which the fault f is present. $A(N)$ and $A(N^f)$ denote the values of attributes for the fault-free and the faulty circuit, respectively.

DEFINITION 5.2 The attribute is said to *cover the fault f* iff $A(N) \neq A(N^f)$.

Example 5.2 If input x_1 is stuck-at 0 for the function $F9$ from Example 5.1, then r_0 will have the value 6 and the attribute r_0 covers this fault. Please notice that in this example the meaning of attribute-testability is synonymous to syndrome-testability as well as to r_0 spectral coefficient testability. However, in the general case, the attribute-testability has much wider meaning than only syndrome-testability that was described in the previous section.

DEFINITION 5.3 Let N be a combinational circuit and ϕ a set of faults in N. Then an ordered set of attributes S is a *signature* for $\langle N, \phi \rangle$ iff every fault in ϕ is covered by at least one attribute in S.

The set of gates G from which Boolean functions are implemented is constrained to $G = \{\text{NAND}, \text{NOR}, \text{AND}, \text{OR}, \text{NOT}\}$. Hence the XOR gates are excluded from the considerations.

DEFINITION 5.4 A fault f that is testable by a spectral coefficient r_α is termed r_α-testable, if $r^f_\alpha \neq r_\alpha$. This is a generalization of syndrome-testability.

At the beginning, the network topologies are also limited to an arbitrary fan-out free network denoted by T and composed of gates from G. Let ϕ_M denote the complete set of multiple-stuck-at faults in T. Later, in the sequel, more complex topologies will be considered.

Result 5.1: Set $\{r_0\}$ is a signature for $\langle T, \phi_M \rangle$.

Result 5.1 is equivalent to Result 4.4 stated by Savir.

Consider now the usage of r_0 to the wider class of circuits. Let N be an arbitrary irredundant network composed of gates from G that has equal inversions parity along two reconvergent paths. It is obvious that this type of circuits can realize only unate functions. Let ϕ_s denote the complete set of single stuck-at faults in N. Then, the following result is valid:

Result 5.2: Set $\{r_0\}$is a signature for $\langle N, \phi_s \rangle$.

Example 5.3 Consider the Boolean function $F11$ shown in Fig. 11. This network is of N type since it has equal number of inversions parity along any two-reconvergent paths. Then the signature to protect this circuit against any single stuck-at fault is r_0. The correct value of r_0 for this circuit is 5.

One can notice that the class N of circuits is much wider than fan-out-free circuits. However, it is still limited to only unate functions. Now, the class of the circuits will be widen in such a way, that any of the input lines may fan out without the restriction on even inversion parity. Internal lines are still subject to this restriction. Circuits of this type are much more general, allowing all functions to be realized. This type of networks will be denoted by O. Please notice, that this class of circuits includes all two-level sum-of-products realizations and PLAs so large classes of practical circuits are in this class. As previously, an arbitrary circuit from the class O is built of gates from G. ϕ_S denotes the set of all single stuck-at faults in 0. The following results can be stated for such kind of circuits.

Result 5.3: Set $\{r_0\}$ is a signature for $\langle O, \phi_s \rangle$ for all circuits O for which all first-order coefficients are nonzero.

Result 5.4: If r_0 is odd, then set $\{r_0\}$ is a signature for $\langle O, \phi_s \rangle$.

Result 5.5: If $1/2(r_0 - r_i)$ is odd and $r_{i\alpha} \neq 0$, for some i, where $i \in \{1, 2, \ldots, n\}$, then set $\{r_0, r_i\}$ is a signature for $\langle O, \phi_s \rangle$.

Result 5.6: If 1/2 r_0 is odd and $r_i = 0$ for some i, where $i \in \{1, 2, \ldots, n\}$, then set $\{r_0, r_i, r_{i\alpha}\}$ is a signature for $\langle O, \phi_s \rangle$ for any α such that $r_{i\alpha} \neq 0$, where $i \notin \alpha$, $\alpha \in \{1, 2, 12, 3, 13, 123, \ldots, 12, \ldots, n\}$.

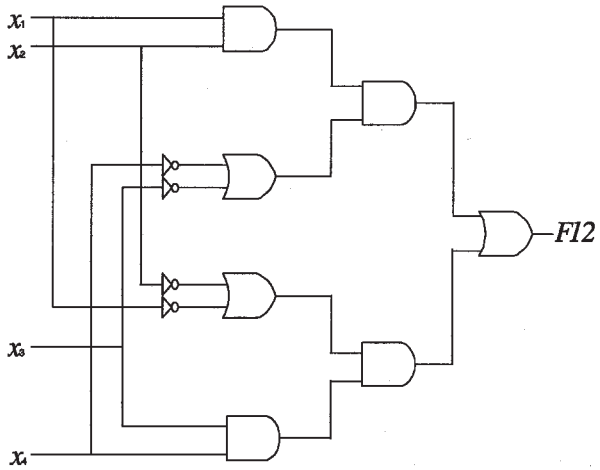

FIGURE 12 A four variable Boolean function F12.

Example 5.4 Consider the Boolean function $F12 = x_1x_2 \oplus x_3x_4$ that is shown in Fig. 12. The spectrum R of the function $F12$ is as follows:

$$r_0 = 6, \quad r_1 = -2, \quad r_2 = -2, \quad r_3 = -2, \quad r_4 = -2,$$
$$r_{12} = 2, \quad r_{13} = -2, \quad r_{14} = -2, \quad r_{23} = 2,$$
$$r_{24} = -2, \quad r_{34} = 2, \quad r_{123} = 2, \quad r_{124} = 2,$$
$$r_{134} = 2, \quad r_{234} = 2, \quad r_{1234} = -2.$$

Since all r_1, r_2, r_3, r_4 are nonzero then the conditions of Result 5.3 are fulfilled and $\{r_0\}$ is a signature for this circuit that covers all single stuck-at faults.

Example 5.5 Consider the Boolean function $F13 = x_1x_3 + \overline{x_1x_3} + x_2$ that is realized by the circuit in Fig. 13. The spectrum R of the function $F13$ is as follows:

$$r_0 = 6, \quad r_1 = 0, \quad r_2 = -2, \quad r_3 = 0, \quad r_{12} = 0,$$
$$r_{13} = 2, \quad r_{23} = 0, \quad r_{123} = 2.$$

For the function $F13$ the conditions of Result 5.3 are not satisfied since $r_1 = r_3 = 0$. Hence, set $\{r_0\}$ is not enough on its own. What is more, $1/2(r_0 - r_2) = 4$ what is an even number and the conditions of Result 5.5 are not fulfilled. However, as $1/2(r_0) = 3$ then the conditions of Result 5.6 are fulfilled and one can use set $\{r_0, r_1, r_{13}\}$ as a signature to protect this circuit against all single stuck-at faults.

All developments that have been presented up to now give a number of conditions, which, if satisfied, lead to particular signatures. One should, however, notice that the complexity of the signatures increases as the results are sought. However, when none of the stated Results is satisfied then more general conditions are needed to derive a signature for a circuit. Three different possible general signatures are described below.

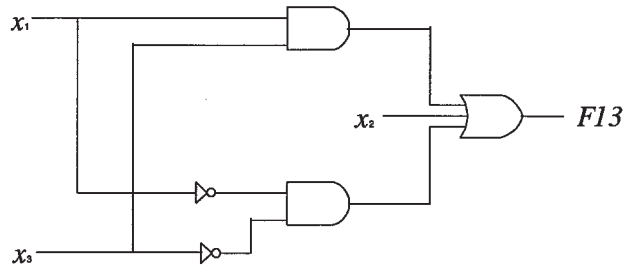

FIGURE 13 A three variable Boolean function F13.

DEFINITION 5.5 A *basis signature* consists of r_0 and n other spectral coefficients whose subscripts, when written as n-bit vectors, form a basis over the space of n-bit vectors.

In order to present a subscript of a coefficient as a binary vector, the variables involved in the coefficient are indicated by 1's in the binary vector. Hence, for the spectrum R of a three variable Boolean function the spectral coefficients $r_0, r_1, r_2, r_{12}, r_3, \ldots, r_{123}$ are denoted by $r_{000}, r_{001}, r_{010}, r_{011}, r_{110}, \ldots, r_{111}$, respectively.

A basis signature is developed in an iterative way. At every step, the spectral coefficients that are eligible to enter the basis are examined. First, the selection is made by highest magnitude and, within the highest magnitude, by the order of the coefficient (i.e. by the number of ones in its binary subscript). In a case that two spectral coefficients have the same magnitude and order, then the one whose binary subscript has the smallest decimal equivalent is chosen. Next example will show the application of the procedure.

Example 5.6 Consider the Boolean function $F14 = \Sigma(1, 3, 5, 7, 8, 10, 12, 15)$ that has the following spectrum R which will be given for spectral coefficients described in both notations.

$$r_0 = r_{0000} = 8, \quad r_1 = r_{0001} = -2, \quad r_2 = r_{0010} = 0,$$
$$r_{12} = r_{0011} = 2, \quad r_3 = r_{0100} = 0, \quad r_{13} = r_{0101} = 2,$$
$$r_{23} = r_{0110} = 0, \quad r_{123} = r_{0111} = -2,$$
$$r_4 = r_{1000} = 0, \quad r_{14} = r_{1001} = -6,$$
$$r_{24} = r_{1010} = 0, \quad r_{124} = r_{1011} = -2,$$
$$r_{34} = r_{1100} = 0, \quad r_{134} = r_{1101} = -2,$$
$$r_{234} = r_{1110} = 0, \quad r_{1234} = r_{1111} = 2.$$

For the *basis signature*, in addition to r_{0000} one chooses:

1. r_{1001}—highest magnitude
2. r_{0001}—lowest order of magnitude 2
3. r_{0011} and r_{0101}—lowest remaining order of coefficients

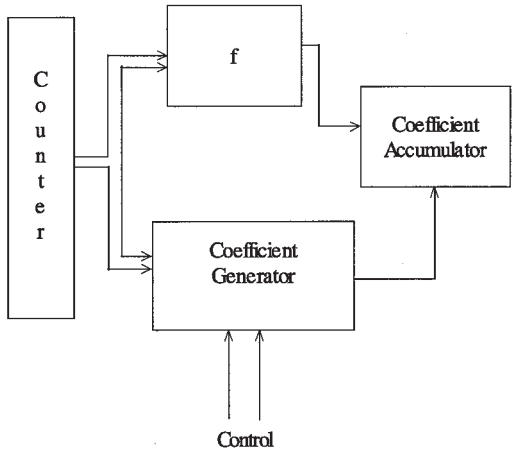

FIGURE 14 Basic circuit checking the signature.

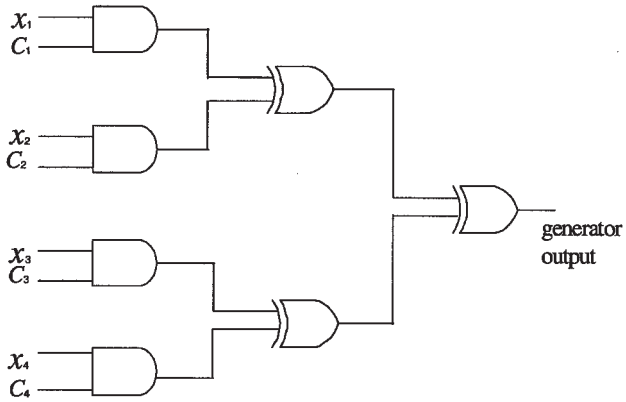

FIGURE 15 A four variable coefficient generator.

with magnitude 2 that have been chosen according to the order of their decimal equivalent numbers.

Then by Definition 5.5 the basis signature for Example 5.6 is $\{r_0, r_{14}, r_1, r_{12}, r_{13}\}$

Result 5.7: A *basis signature* is a signature for $\langle 0, \phi_s \rangle$.

DEFINITION 5.6 A *minimal covering signature* consists of r_0 and a minimal set of nonzero spectral coefficients, selected in such a way that each x_i for which $r_i = 0$ is involved in at least one coefficient in the signature.

Hence, the selection of a minimal covering signature is an example of a set-covering problem for which there exist many algorithms. The next result follows directly from Definition 5.6.

Result 5.8: A *minimal covering signature* is a signature for $\langle 0, \phi_s \rangle$.

The next result is a direct consequence of Definition 5.6 and Result 5.8.

Result 5.9: When the highest-order coefficient is not equal to 0 then a minimal covering signature consists of just r_0 and the highest-order coefficient.

Let us introduce as an alternative to the minimal covering signature, a minimal input signature, which is constructed according to the following result. It deals with the condition when one variable reflects stuck-at faults in the other variable.

Result 5.10: A *minimal input signature* $\{r_0, r_{i_1}, r_{i_2}, \ldots, r_{i_r}\}$, where $i_k \in \{1, 2, \ldots, n\}$ is a signature for $\langle 0, \phi_s \rangle$ if, for every j such that $r_j = 0$, there exists an $i \in \{i_1, i_2, \ldots, i_r\}$ and $r_{ij} \neq 0$.

EXAMPLE 5.7 Consider the Boolean function $F5 = x_1\bar{x}_2 + \bar{x}_1x_3 + x_2\bar{x}_3 + x_4x_5 + \overline{x_4x_5}$ which is realized by the circuit shown in the Fig. 7. This is the same function for which previously the syndrome-testable design was shown in Fig. 8. In the spectrum R of this function all but $r_0 = 28$, $r_{12} = -4$, $r_{13} = -4$, $r_{23} = -4$, $r_{45} = 4$, $r_{1245} = 4$, $r_{1345} = 4$, $r_{2345} = 4$ spectral coefficients are equal to 0. For this function the following alternative spectral signatures can be used.

1. basis—$\{r_0, r_{12}, r_{13}, r_{45}, r_1, r_4\}$
2. minimal covering—$\{r_0, r_{1345}, r_{12}\}$
3. minimal input—$\{r_0, r_1, r_2, r_4, r_5\}$

In order to verify the signature, a simple circuit needs to be built. The basic scheme is shown in the Fig. 14. Each coefficient in the signature is tested separately. The counter counts through all possible 2^n combinations of n variables. The generator of the coefficients can have different realizations—an example of a four variable coefficient is shown in Fig. 15. The x_i inputs come from the counter, while the values c_i come directly from the binary subscripts labeling each coefficient, for example, r_{0011} requires $c_4 = c_3 = 0$ and $c_2 = c_1 = 1$. Such a design of the coefficient generator can be generalized easily to any number of variables. The generator itself can be easily checked for errors.

When the coefficient generator is built in a way that has just been described above, then the coefficient accumulator is only build out of an up/down counter which is incrementing or decrementing for true minterms of the Boolean function F accordingly to the output from the coefficient generator of 0 or 1, respectively. It is then the direct implementation of the graphical method of the calculation of spectrum R described in the "Links between spectral techniques and classical logic design" section. When one wants to test more than one coefficient in parallel then the coefficient generator and the coefficient accumulator for each coefficient have to be repeated.

Let us consider now multiple-input stuck-at faults. Single input stuck-at faults are a special case. In a network with p input lines, there are $3^P - 1$ possible multiple faults. This large number makes an exhaustive analysis of faults impractical. The research on spectral signatures for multiple-faults has been presented in details in Ref. [45]. The main ideas from this paper are presented below.

Consider a k-multiple-input fault in a network realizing the Boolean function F that has the spectrum R. Suppose $\{x_{\lambda_1}, \ldots, x_{\lambda_k}\}$ are the stuck-at inputs with x_{λ_i} stuck at u_i, $1 \leq i \leq k$, $u_i \in \{0, 1\}$. Let us denote this fault by

"$x_\lambda(s/u)$", where $u = \sum_{i=1}^{k} u_i 2^{i-1}$. Please notice that this class of multiple-input stuck-at faults is sufficient, since all others are equivalent among themselves to input variables permutation. Let also α be a possible empty string of variable labels from $\{1, \ldots n\}$ such that α and $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ are disjoint. When $\alpha = \varnothing$ then $r_\alpha = r_0$. Let λ' be a nonempty subset of λ.

Result 5.11: $x_\lambda(s/u)$ is

(a) $r_{\alpha\lambda'}$—testable iff $r_{\alpha\lambda'} \neq 0$.
(b) r_α—testable iff $r_\alpha \neq [r_\alpha, r_{\alpha\lambda_1}, r_{\alpha\lambda_2}, \ldots, r_{\alpha\lambda_1\lambda_2\ldots\lambda_k}]T_u^k$ where T_u^k, is the $(u+1)$th column of the transform matrix T^k.

Example 5.8 Consider the Boolean function $F15 = x_1x_2x_3 + x_4(x_2 + x_3)$ that has the following spectrum R:

$$r_0 = 8, \quad r_1 = -2, \quad r_2 = 0, \quad r_3 = 0, \quad r_4 = -6,$$
$$r_{12} = 2, \quad r_{13} = 2, \quad r_{14} = 0, \quad r_{23} = 0, \quad r_{24} = -2,$$
$$r_{34} = -2, \quad r_{123} = -2, \quad r_{124} = 0, \quad r_{134} = 0,$$
$$r_{234} = 2, \quad r_{1234} = 0.$$

All possible combinations of multiple-input stuck-at faults for inputs x_1 and x_2 will be considered. When x_1 and x_2 are both stuck-at 0 then it is denoted by $(x_2, x_1)/0$, when $x_1/1$ and $x_2/0$ then it is denoted by $(x_2, x_1)/1$, when $x_1/0$ and $x_2/1$ then it is denoted by $(x_2, x_1)/2$, and finally, when both x_1 and x_2, are stuck-at 1 then it is denoted by $(x_2, x_1)/3$. Hence, every time the decimal number following the bracket with the multiple-input stuck-at faults corresponds to the binary number describing the types of stuck-at faults (either 0 or 1) for each stuck-at input line. Of course, it is true for any number of input stuck-at lines. Applying Result 5.11, one can find the described faults to be syndrome-testable (i.e. r_0-testable) using Tokmen's decomposition theorem [84] iff $r_0 \neq [r_0 r_1 r_2 r_{12}]T_2^2$.

Then $(x_2, x_1)/0$ is not syndrome-testable since $8 = [8\ -2\ 0\ 2][1\ 1\ 1\ 1]^t$. On the other hand, $(x_2, x_1)/1$ and $(x_2, x_1)/2$ are both syndrome-testable since $8 \neq [8\ -2\ 0\ 2] \times [1\ -1\ 1\ -1]^t$ and $8 \neq [8\ -2\ 0\ 2][1\ 1\ -1\ -1]^t$ accordingly. And finally $(x_2, x_1)/3$ is not syndrome-testable since $8 \neq [8\ -2\ 0\ 2][1\ -1\ -1\ 1]^t$.

As one can notice it is difficult to use Result 5.11 in general and that is why the more practical results can be derived from the following results that, unfortunately, deal with a restricted cases.

Result 5.12:

(a) If $r_\alpha \neq 0$, r_α covers all multiple faults involving any x_i if $i \in \alpha$.
(b) If the highest-order coefficient $r_{1\ldots n} \neq 0$, it is a signature for all $3^n - 1$ multiple-input faults.
(c) A sufficient condition for r_α to cover a k multiple-input fault is that the fault-free value of r_α is not divisible by 2^k.
(d) If r_0 is odd, any coefficient covers all multiple-input faults.

One should notice, that Result 5.12 (c) gives a sufficient, but not necessary condition. It has been shown in Ref. [45] that a fan-out free network, without XOR gates, having two or more inputs, has to realize a function with the same parity. Since r_0 is odd for such a function, all spectral coefficients are odd due to Property 2.9 of the spectrum R. Since all stuck-at faults, single or multiple, in a fan-out free circuit, are functionally equivalent to single or multiple stuck-at input faults [36], it follows from Result 5.12 (c) that they all are testable by r_α From Result 5.4 it is also known that all input faults are testable by r_0 if r_0 is odd. Hence, the above results generalize Result 5.4 for any r_α and for cases when r_0 is even.

As mentioned before, the analysis of r_α-testability for input faults not fulfilling the conditions of Result 5.12 can be difficult. However, the simplification is possible for some cases, giving the possibility of obtaining more efficient results.

Result 5.13:

(a) $x_i/0$ and $x_i/1$ are either r_α-testable or r_α-untestable.
(b) The 2^n-multiple faults involving all of $x_1, \ldots, x_n$ are either all r_α-testable or all r_α-untestable.

Example 5.9 Consider the function $F6$ shown in Fig. 9, for which each of $\{r_1, r_4, r_{12}, r_{13}, r_{24}, r_{34}, r_{123}, r_{234}\}$ tests all multiple faults involving two or more inputs, while r_0 tests those faults involving all four inputs.

Consider now a single-internal stuck-at fault. The network output is considered to be an internal line as well. For internal lines, the function F is rewritten to make the Boolean dependence on an internal line g explicit. Thus $F(x) = h(x, g(x))$, where $g(x)$ is the function realized by the line g, and h is a function of $n+1$ inputs, as shown in the Fig. 16.

Example 5.10 Consider again the function $F6$ shown in Fig. 9. The line labeled $g(x) = x_2x_3$ and $h(x, g) = x_1g + x_4(\bar{x}_2 + \bar{x}_3)$. As is shown in Fig. 16, $g(x)$ and $h(x, g)$ may have common inputs and their realizations may share gates.

In the case that the inputs to g and the rest of the circuit are disjoint then the following simplified results can be used. The $g(x)$ and $h(x, g)$ are defined by the network in question and not by $F(x)$ alone. It is possible that either $g(x)$ and $h(x, g)$ may be independent of one or more x_i, $1 \leq i \leq n$. In order to prove that the line g is testable, by either the syndrome r_0 or by some other spectrum signature, some coefficients for the spectrum of function h are required and the proofs depend heavily on the properties of the decomposition of the spectra [84]. Let R^h be the spectrum of $h(x, g)$, and x_{n+1} denote the line g. Then the following result can be stated.

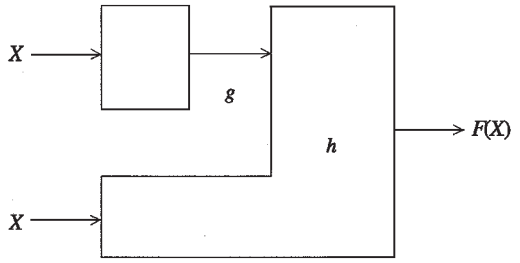

FIGURE 16 Dependence of the function F on an internal line g.

Result 5.14: The fault g stuck-at 0 is testable by a coefficient r_α iff $2r_\alpha \neq (r^h_\alpha + r^h_{\alpha n+1})$. The fault g stuck-at 1 is testable by a coefficient r_α iff $2r_\alpha \neq (r^h_\alpha - r^h_{\alpha n+1})$.

The above result depends on both R, the spectrum of the function F realized by the network, and R^h, the spectrum of the function realized by the residual network found by treating g as a primary input. The multiplication by 2 is due to the fact that $h(x, g)$ is a function of $n + 1$ variables, whereas $h(x, 0)$ and $h(x, 1)$ are functions of only n variables. As for input faults, it is convenient to express the results for syndrome-testability r_0.

Result 5.15: The fault g stuck-at 0 is syndrome-testable iff $2\ r_0 \neq (r^h_0 + r^h_{n+1})$. The fault g stuck-at 1 is syndrome-testable iff $2r_0 \neq (r^h_0 - r^h_{n+1})$.

For the circuit in Example 5.10 shown in Fig. 9, $r_0 = 8$, while $r^h_0 = 17$ in the function $h(x, g)$, and $r^h_{n+1} = -5$ corresponding to the input g. Thus from Result 5.15 g stuck-at 0 and stuck-at 1 are syndrome-testable.

It is obvious that the evaluation of the network g and h for a large number of lines in a network would be time consuming. However, such an evaluation is needed only for comparatively small number of cases when the internal line is not unate. In determining the testability of large networks, unate lines, as described in the previous section, play an important role. Consequently, it is useful to determine the unate lines of a large network before testing. A simple method exists that does this in a linear time [36,41,48]. Here, the algorithm presented there for one output circuit is described.

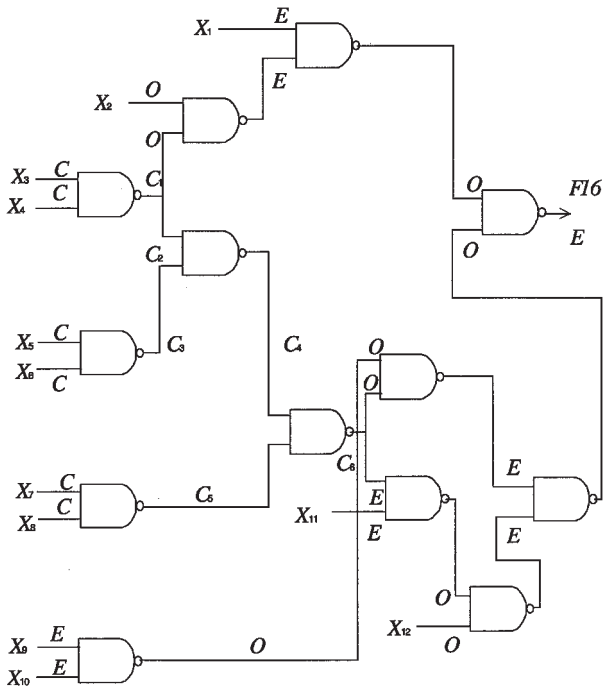

FIGURE 17 Labeling unate and not unate lines in Boolean function F16.

TABLE I Commutative and associative operators

	E	O	C
E	E	C	C
O	C	O	C
C	C	C	C

The following procedure identifies the unate lines in an irredundant single-output combinational circuit. Nonunate lines are labeled C. Unate lines are labeled E or O according to the fact whether there is an even or odd number of inverters on every path from that line to the circuit output.

Procedure:

1. Label the circuit output E.
2. For each gate whose output line is labeled but whose input lines are not, do one of the following:
   (a) AND, OR gates: label all input lines identical to the output line;
   (b) NOT, NAND, NOR gates: label the input lines E if the gate's output line is labeled O, O if the gate's output line is labeled E, and C if the gate's output line is labeled C;
   (c) EXOR, EXNOR gates: label each input C.
3. For each fan-out point whose branches are labeled but whose stem is not, construct the stem label from the branch labels using the following commutative and associative operator shown in Table I.
4. Repeat steps 2 and 3 until all network lines are labeled.

Example 5.11 Consider the Boolean function $F16$ shown in Fig. 17 labeled according to the above algorithm. This circuit realizes a function with an odd number of ones, so all input stuck-at faults, both single and multiple, are r_0-testable. There are only six internal nonunate lines which are candidates for syndrome testing—in this figure they are labeled $c1 \ldots c6$. Of these, $c1/1$, $c3/1$ and $c5/1$ are r_0-testable since they are functionally equivalent to input stuck-at faults. So are $c1/0$, $c3/0$ and $c5/0$. Faults on $c4$ and $c6$ are syndrome-testable since they result in a faulty function independent of some x_i. $c2/0$ is syndrome-testable since it is functionally equivalent to $c3/0$. $c2/1$ is the only fault that has to be explicitly considered using Result 5.15. By checking the conditions of this result, the last mentioned fault has been found to be syndrome-testable. Hence the whole circuit from Fig. 17 is syndrome-testable for all single stuck-at faults.

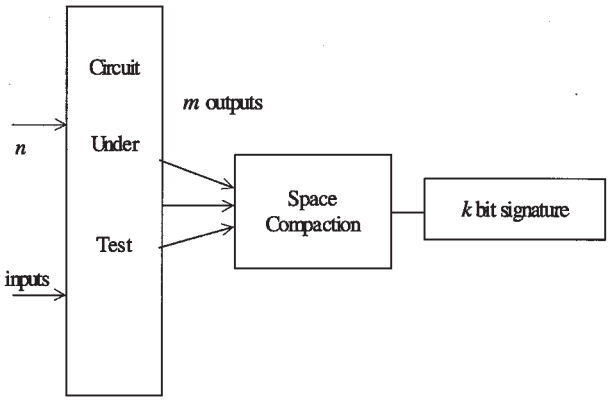

FIGURE 18 The parallel space compaction.

6. SPECTRAL COEFFICIENTS SIGNATURES AND TESTING OF PROGRAMMABLE LOGIC ARRAYS

Miller and Muzio [48,54] developed the notion of spectral coefficients signatures. The idea is very similar to WSS described in the "Syndrome, constrained syndrome and weighted syndrome sums testing" section. The only difference is the fact that WSS deals only with syndrome, i.e. r_0-testability while Muzio and Miller used the whole spectrum R. It should be noticed that for both these approaches, a single signature could be computed in parallel for all outputs of a multiple-output circuit, as opposed to having separate signatures to be compared for each of them. In the literature, such an approach is called the *space compaction* [37,38,56] against *time compaction*, which is the actual compaction to produce a signature of length k bits from a stream of up to 2^n bits. The research presented here can be found with all the proofs and derivations in Ref. [75–77]. In this section it is shown also that the space compacted counting signature is excellent for certain types of circuits, specifically PLAs. For PLAs full testability of all single faults is guaranteed. The *space compaction* implies using a compaction method on the output vectors both in time and in space. In a circuit with n inputs and m outputs, there are m output vectors each of length 2^n to be tested. The testing proposed here (by application of either a syndrome counter or a linear feedback shift register—LFSR) reduces them to m vectors of length r each, where $r \ll 2^n$, after 2^n time cycles. For an LFSR, r is usually 16 bits, while for a syndrome counter $r = n + 1$ bits [89]. The space compaction proposed here reduces the m vectors of length r to one of length k, where k is typically of the order $m + n$. Figure 18 gives a schematic view of the processes which are applied in parallel to conserve both space and time for the tester. The space compacted signature presented here tries to achieve full testability with the simplest possible coding. The goal is to use as a tester a weighted sum of function weights. As the circuits with many outputs are considered, it is necessary to identify the vectors and spectra for the individual outputs. Square brackets $[F]_i$ and $[R]_i$ indicate respectively the truth vector and the spectrum for the i-th output in the circuit.

DEFINITION 6.1 The *weighted spectral sum* (WSPS) for a network of m outputs is as follows:

$$K = \sum_{i=1}^{m} w_i T^n [F]_i = \sum_{i=1}^{m} w_i [R]_i \tag{23}$$

with individual entries in K denoted by k_α. The weights w_i are positive nonzero integers (the same as for WSS discussed previously). Note also, that the first coefficient of the weighted spectrum, k_0, is the weighted sum of the first coefficients of the individual spectra for each function $[F]_i$, since the first row of $[T^n]$ are all ones. Thus k_0 is the weighted sum of the weights of all functions in the circuit. The exact relationship between k_0 and WSS from Definition 4.4 is as follows:

$$\mathrm{WSS} = \frac{1}{2^n} k_0 \tag{24}$$

The WSPS is the unnormalized weighted sum of spectra and thus its first coefficient, k_0, is the unnormalized weighted sum of the function weights. Often however, the two names "WSS" and "syndrome sum" are also informally used to denote k_0 [44,55]. In practice, it is the latter value that is computed by the tester. The testability properties of the WSS and k_0 are isomorphic, and the conditions hold for both. Formally due to the way of the calculation of K, one always has to consider all functions to depend on n inputs, and all $[F]_1 [F]_2, \ldots, [F]_m$ to be vectors of 2^n entries. It means that the Boolean functions that have smaller number of variables than n, where n denotes the number of the input variables for this Boolean function that has the biggest number of them, have to have "dummy" input variables on which their truth values do not depend. Hence, in such a setting, $2^n \mathrm{WSS} = k_0$, so that WSS is a real number while k_0 remains a larger integer.

The testability for the space compaction of the WSPS is different from the single output case discussed in the previous section since *masking of faults* is more complex to detect.

DEFINITION 6.2 *Masking of faults* or *aliasing* is said to occur when a fault free and a faulty circuit have the same signature.

When a fault occurs on a line in the circuit, it is denoted by $[F]_i^f$, i.e. the function produced by the faulty circuit at output i. The syndromes or some spectral coefficients at the outputs, which depend on the faulty line, change if the fault is testable at those outputs. When a set of coefficients from the WSPS is used as a signature for a circuit, there still may be faults that are untestable due to two reasons: (1) they do not change the signature at any of the outputs, (2) there is a cancellation effect created by the weighted summation process inherent in the WSPS definition.

In the former case, the problem is the same as that for separately testing all the single outputs. If the signature is masked at all outputs for a particular fault, then the design must be changed to achieve testability, or signature must be increased. A signature from a space-compacted WSPS is very useful in this case. It is shown later in this section that one simple solution to this problem is to introduce an extra output in the network that acts as a test point for untestable lines. Since the testing is done in parallel for all outputs, the extra output does not increase the testing time and only increases the storage of the space-compacted signature by a small amount.

For the second case, the aliasing is the problem new in itself due to the space compaction method. The weighted summation may cause the cancellation of two faulty signatures, which does not alias on their own. The same problem and example (Example 4.4) has already been presented for WSS. There is always, however, the possibility of choosing such different weights as to avoid such a cancellation. The designs for testability changes are simple and can be checked in a deterministic way [75,76]. In practical implementation, it is achieved by changing the layout with no extra hardware being required [38].

At the beginning, the case for a single stuck-at fault on an input line for a circuit with n inputs and m outputs will be considered.

Result 6.1: Let $k_{\alpha i}$ be a coefficient from a WSPS with some set of weights, where $k_{\alpha i} = \sum_{i=1}^{m} w_l [r_{\alpha i}]_l$. Then the input stuck-at faults $x_i/0$ and $x_i/1$ are k_α-testable and $k_{\alpha i}$-testable iff $k_{\alpha i} \neq 0$.

Similarly as for the results from the previous Section in the case of syndrome-testability (i.e. r_0-testability), it is always useful to state explicitly the condition for k_0-testability (WSS-testability). The next result deals with WSS-testability.

Result 6.2: Input stuck-at faults $x_i/0$ and $x_i/1$ are k_0-testable (WSS-testable) iff $k_i \neq 0$.

Example 6.1 Consider the circuit of Fig. 10 showing two Boolean functions $F7$ and $F8$ having four inputs. This circuit is the same as used previously in Example 4.4 and in Ref. [4]. There, exhaustive simulation had to be performed in order to show that this circuit is WSS untestable for the weights chosen in Example 4.4. Below, a set of weights is shown that make this circuit testable. The functions are $F7 = (x_1x_4 + x_3)x_2$ and $F8 = \bar{x}_1x_2 + x_1\overline{x_2x_3}$. The weights chosen in Example 4.4 are $w_1 = 2$ and $w_2 = 3$, giving $K = T^4\ (2[\mathrm{F}]_1 + 3[F]_2)$. Then, the WSPS is as follows:

$$k_0 = 28, \quad k_1 = 4, \quad k_2 = -16, \quad k_{12} = -16, \quad k_3 = 0,$$
$$k_{13} = -8, \quad k_{23} = 12, \quad k_{123} = -4, \quad k_4 = -2,$$
$$k_{14} = 2, \quad k_{24} = 2, \quad k_{124} = -2, \quad k_{34} = -2,$$
$$k_{134} = -2, \quad k_{234} = 2, \quad k_{1234} = -2.$$

Consider the fault $x_3/0$. Since $k_3 = 0$, then $x_3/0$ is k_0 untestable (WSS untestable). However, $k_{13} \neq 0$. So $x_3/0$ is k_1-testable. Hence, the circuit which is not WSS-testable can be WSPS-testable as it is for the input $x_3/0$. It is also possible to choose the values of coefficients w_i to produce the change of WSPS in such a way that it fulfills the conditions of Result 6.2 according to which the functions $F7$ and $F8$ are WSS-testable. Let us now consider the weights for the WSPS equal $w_1 = 1$ and $w_2 = 2$. Then, the corresponding WSPS is as follows:

$$k_0 = 17, \quad k_1 = 3, \quad k_2 = -9, \quad k_{12} = -11, \quad k_3 = 1,$$
$$k_{13} = -5, \quad k_{23} = 1, \quad k_{123} = -3, \quad k_4 = -1,$$
$$k_{14} = 1, \quad k_{24} = 1, \quad k_{124} = -1, \quad k_{34} = -1,$$
$$k_{134} = 1, \quad k_{234} = 1, \quad k_{1234} = -1.$$

Let us note that $k_3 = 1$ and $x_3/0$ is k_0-testable (WSS-testable). What is more, since all $k_i \neq 0$ then all input stuck-at faults are k_0-testable. As all coefficients are nonzero then all input stuck-at faults are k_α-testable for any α. Hence, only one computation is necessary to find the fault-free vector to which the transform is applied. All input faults are checked for testability directly from the WSPS coefficients.

Let us consider now an internal line g. The appropriate functional model is given by $[F(x)]_j = h_j(x, g(x))$ for each j, $1 \leq j \leq m$. Let us notice that the model for each separate function $[F(x)]_j$ is exactly the same as the model considered previously for a single function $F(x)$ that was shown in the Fig. 16. Each h_j is considered as a function of $n + 1$ inputs and it measures the Boolean dependence of an output $[F]_j$ on the line g. Let $[R^h]_j$ be the spectrum of h_j, $[V]_j$ the vector representation of h_j, and let $K^h = \sum_{j=1}^{m} w_j [R^h]_j$. As before there is no need to calculate the $[R^h]_j$ separately. One applies the transform once to obtain $K^h = \sum_{j=1}^{m} w_j T^{n+1} [V]_j$. In order to check the WSPS testability of an internal line one has to construct the Boolean functions h_j from the circuit description. In some cases only a subset of the m outputs may depend on the line g and only that subset of corresponding h_j functions need to be explicitly evaluated. The remaining h_j functions are equal to the corresponding $[F]_j$, although they must be formally considered as the functions of $n + 1$ variables.

Result 6.3: The fault $g/0$ is k_α-testable iff $k_\alpha \neq 1/2(k_\alpha^h + k_{\alpha n+1}^h)$. The fault $g/1$ is k_α-testable iff $k_\alpha \neq 1/2(k_\alpha^h - k_{\alpha n+1}^h)$.

It is also useful to state the condition for k_0-testability (i.e. *WSS*-testability).

Result 6.4: The fault $g/0$ is k_0-testable iff $k_0 \neq 1/2(k_0^h + k_{n+1}^h)$. The fault $g/1$ is k_0-testable iff $k_0 \neq 1/2(k_0^h - k_{n+1}^h)$.

When the weights chosen for the WSPS are positive, then all internally unate lines are automatically k_0-testable. This is particularly important for two-level

circuits and for special circuits such as PLAs discussed below.

The main application of the theory developed by Miller, Serra and Muzio is for testing of PLAs. The summary of the results from Refs. [75–77] is presented. Several approaches for testing PLA's are discussed in the literature [38]. However, most of the methods are still based on the classical testing using test sets [29]. In the classical approach, mainly decreasing the time needed for the development of the tests and changing the PLAs functions in order to incorporate parity check lines can achieve the improvements. Probabilistic testing [78] has also been proposed as the solution to this problem, however, the fault coverage for this method is not a full one. Since the PLAs have the regular structure, then it is difficult to test them efficiently by using automated test pattern generators. Then the common solution in many methods dealing with testing of the PLAs is to increase the testability of the PLAs by introducing extra lines, in order to achieve better controllability of the two arrays.

Yamada [88] discussed the syndrome testing of PLAs. His approach is the closest to the method discussed here. In Yamada's method, the outputs are tested separately and a complex algorithm is required to determine the extra hardware that is necessary to test input faults. By introducing space compaction testing, and by using k_0 from the WSPS, both these problems can be overcome.

The structure of a PLA consists of the AND section of the array that forms a number of p_i product terms which are combined in the OR section of the array. Each output of the array is logically a two-level network and hence internally unate.

Smith [79,80] has identified four classes of faults unique to PLAs:

1. *Growth fault*: a missing contact in the AND section causes a product term to loose a literal and, thus, double in size.
2. *Shrinkage fault*: an extra contact in the AND section causes a literal to be added to a product term causing it to include half or none of its original minterms.
3. *Disappearance fault*: a missing contact in the OR section causes a product term to be dropped from the corresponding output.
4. *Appearance fault*: an extra contact in the OR section causes a product term to be added to the corresponding output.

DEFINITION 6.3 A contact fault is called a *functional fault* if it alters the behavior of at least one array output.

It is important to introduce this qualification since there exist certain contact faults that have no influence on the functional behavior. For example, an appearance fault may simply introduce a redundant product term to an output. Such an addition does not alter the functional behavior of the output.

A functional growth or an appearance fault results in $r_0^f > r_0$. On the other hand, a functional shrinkage or a disappearance fault results in $r_0^f < r_0$ and all single functional contact faults are r_0-testable. What is more, all multiple faults composed of growth and appearance and all multiple faults composed of shrinkage and disappearance faults are r_0-testable. Other multiple faults must be explicitly examined to ensure $r_0^f \neq r_0$.

Then all the types of single faults in PLAs can be considered in three groups.

1. *Stuck-at faults*: one line or line segment stuck at 0 or 1.
2. *Bridging faults*: two adjacent lines are shorted together assuming the value of the logical AND and OR of their own individual signals.
3. *Cross-point faults*: the loss of contact of a connection or its spurious presence.

In was shown in Refs. [75,76] that the coefficient k_0 is a complete test for all internal single faults (i.e. stuck-at, cross-point, and bridging). Result 6.2 covers the testability condition for primary input stuck-at faults. In the case of input bridging faults, any special cases can be detected by the following results. However, only the bridging faults of AND type are considered here.

Result 6.5: An AND-bridging fault between x_i and x_j is both r_α- and $r_{\alpha ij}$-testable, where $i,j \notin \alpha$ iff $r_{i\alpha} + r_{j\alpha} + 2r_{ij\alpha} \neq 0$.

Result 6.6: An AND-bridging fault between x_i and x_j is $r_{i\alpha}$-testable, where $i,j \notin \alpha$ iff $r_{i\alpha} - r_{j\alpha} \neq 0$.

Example 6.2 Consider the function that has the following R spectrum:

$$r_0 = 7, \quad r_1 = 1, \quad r_2 = 3, \quad r_3 = -1, \quad r_4 = 3,$$
$$r_{12} = 1, \quad r_{13} = -3, \quad r_{14} = 1, \quad r_{23} = -1,$$
$$r_{24} = -1, \quad r_{34} = -1, \quad r_{123} = 1, \quad r_{124} = 1,$$
$$r_{134} = 1, \quad r_{234} = -1, \quad r_{1234} = 5.$$

A network realizing this function will be r_0-testable for AND-bridging faults between variable pairs except for (x_2, x_3) and (x_3, x_4). For (x_2, x_3), $r_{24} + r_{34} + 2r_{234} \neq 0$, so the bridge is r_4-testable. For (x_3, x_4), $r_{23} + r_{24} + 2r_{234} \neq 0$, so the bridge is r_2-testable. Both these bridges are also r_{13}-testable since $r_{13} \neq r_{12}$ and $r_{13} \neq r_{14}$.

The above results mean that a single coefficient, k_0 which corresponds to a WSS, is sufficient as a complete signature for all single faults in a PLA. Only one signature is required for all outputs, and it is computed in parallel at the same time. However, the weights must be properly chosen to ensure input stuck-at fault coverage. A PLA can be made weighted syndrome-testable for all single stuck-at faults and all single contact faults. By introducing space compaction testing, using k_0, which is a weighted sum of the syndromes of the individual functions, it is shown that k_0 is a complete test for all internal single faults. This

means that k_0 is sufficient as a complete signature for all single faults in a PLA [75,76]. Hence, only one signature is required for the entire test computed in parallel. There are two advantages of such an approach: there exists a deterministic algorithm to check the testability at the design stage, without fault simulation, and when the changes of testability are required then it can be easily achieved. Any verification of testability is performed directly at the design stage, without fault simulation.

Moreover, the conditions for testability of input faults lead to a direct and simple hardware modification in the few cases where there is a problem. This would apply only in the cases when a fault on a primary input proves to be untestable after the direct verification of the corresponding k_i coefficient. There are two cases. The first one is the selection of scalar multipliers in the WSPS such that the resulting k_i are nonzero. The second possibility arises when, for all choices of scalar weights, the WSPS contains first order coefficients that are equal to 0.

As it was described in Property 2.9 for completely specified Boolean functions all the coefficients are either all odd or all even. This property is of course valid for the WSPS as well. Hence, if at least one function in the circuit contains an odd number of 1's in the output truth vector (and thus its weight is odd), one can assign a scalar multiplier of 1 to that output in the WSPS computation, while all other outputs receive scalar multipliers which are even numbers. By such an approach one can assure that the weighted sum of all the weights is odd, that is k_0 is odd. In the consequence, all coefficients in K become odd and are thus nonzero. That in turn, by Result 6.2 ensures that all stuck-at faults on input lines x_i are k_0-testable.

A problem arises when a stuck-at fault on x_i is not testable at any output, that is, it aliases for every signature computed separately on each output. When such a problem exists for PLAs then its solution is still very simple. Let us assume that after the checking, the k_i coefficient is equal to 0, and the weighted sum of $[r_i]_j$, $1 \leq j \leq m$, where each of them is equal to 0. Then, the required hardware change is only the addition of one product line and one output line. The extra function that is realized by such modified PLA must have one product term covering only a single minterm, $c_1 c_2 \ldots c_n$ where each $c_i \in \{x_i, \bar{x}_i\}$. Any minterm suffices, but in general, any function of odd weight could be used.

The extra output $[F]_{m+1}$ is assigned a scalar multiplier of $w_{m+1} = 1$, while $[F]_1, \ldots, [F]_m$ are assigned values of w_1 to w_m that are even. By doing that, the value of k_i becomes either 1 or -1 (depending on the chosen minterm) and obviously all other coefficients k_α become odd and nonzero. This is due to the introduction of the additional function with odd weight. In some cases it can be more advisable to achieve the same effect by using an extra output line connected to already existing product lines. Since, in practice, in the PLAs there are often unused lines in the unmasked layout, then this solution of adding an extra output may not add any overhead. A similar solution can be used to solve the analogous input bridging fault testability problem [75,76]. This hardware addition for PLAs appears to be the simplest proposed so far in the literature [37,38]. The only addition is for the tester, which requires one more bit, since the extra function has a weight of 1, and must be incorporated in the signature. The weighting suggested in Refs. [75,76] is to select powers of 2 for the weights, so that the overall summation can be achieved by the bit position to which each output is connected in the single WSS accumulator. The multiple faults problems have been considered in Ref. [76] as well. It has been shown that the probability of multiple faults not being detected by the WSS signature is small, and this probability is decreasing as the multiplicity of faults or the number of paths through the PLA increases.

7. CONCLUSION

This paper presents developments in the theory of Walsh transforms and its application to fault detection and analysis. The results presented are important both for the derivation of the practical signatures for testing and for the theoretical analysis of testing methods. Space compaction testing may not be the best solution for all circuits, however, it can be very useful for particular models. The main attraction of data compression testing is the small hardware requirement. As the implementation of a Boolean test generation called Difference Propagation has relied heavily on the concept of OBDDs [11], the recent advantages in efficient calculation of various spectral transforms through reduced representations in the form of arrays of cubes and decision diagrams should also increase the interest in application of spectral methods in testing.

References

[1] Aborhey, S. (1989) "Autocorrelation testing of combinational circuits", *IEE Proc. E* **136**(1), 57–61.

[2] Aborhey, S. (1989) "Autocorrelation testing of combinational circuits—erratum", *IEE Proc. E* **136**(2), 100.

[3] Bardell, P.H., McAnney, W.H. and Savir, J. (1987) Built-in Test for VLSI: Pseudo Random Techniques (Wiley, New York).

[4] Barzilai, Z., Savir, J., Markowsky, G. and Smith, M.G. (1981) "The weighted syndrome sum approach to VLSI testing", *IEEE Trans. Comput.* **C-30**(12), 996–1000.

[5] Beauchamps, K.G. (1984) Applications of Walsh and Related Functions (Academic Press, New York).

[6] Beauchamps, K.G. (1987) Transforms for Engineers—A Guide to Signal Processing. (Clarendon Press, Oxford).

[7] Bennetts, R.G. and Hurst, S.L. (1978) "Rademacher–Walsh spectral transforms: a new tool for problems in digital-network fault diagnosis?", *IEE Proc. Comput. Digital Techniques* **1**, 38–48.

[8] Bennetts, R.G. (1982) Introduction to Digital Board Testing (Crane Russak, London).

[9] Bennetts, R.G. (1984) Design of Testable Logic Circuits (Addison-Wesley, London).

[10] Bhattacharya, D. and Hayes, J.P. (1990) Hierarchical Modeling for VLSI Circuit Testing (Kluwer Academic, Boston).

[11] Butler, K.M. and Mercer, M.R. (1992) Assessing Fault Model and Test Quality (Kluwer Academic, Boston).

[12] Chang, C.H. and Falkowski, B.J. (1997) "Efficient symbolic computation of generalized spectra", *Electronics Lett.*, **33**(22), 1837–1838.

[13] Clarke, E.M., McMillian, K.L., Zhao, X., Fujita, M. and Yang, J. "Spectral transforms for large Boolean functions with applications to technology mapping", *Proc. 30th ACM/IEEE Design Automation Conference*, June 1993 pp. 54–60.
[14] Damarla, T. and Karpovsky, M.G. (1989) "Reed–Muller spectral techniques for fault detection", *IEEE Trans. Comput.* **C-38**, 788–797.
[15] DeMicheli, G. (1994) Synthesis and Optimization of Digital Circuits (McGraw-Hill, New York).
[16] Dietmayer, D.L. (1978) Logic Design of Digital Systems (Allyn and Bacon, Boston).
[17] Eris, E. and Miller, D.M. (1983) "Syndrome-testable internally unate combinational networks", *Electronics Lett.* **19**(16), 637–638.
[18] Eris, E. and Muzio, J.C. (1984) "Syndrome- and autocorrelation-testable internally unate combinational networks", *Electronics Lett.* **20**(6), 264–266.
[19] Eris, E. and Muzio, J.C. (1986) "Spectral testing of circuit realizations based on linearizations", *IEE Proc. E* **133**(2), 73–78.
[20] Falkowski, B.J. and Perkowski, M.A. "Algorithms for the calculation of Hadamard–Walsh spectrum for completely and incompletely specified Boolean functions", *Proc. of 9th IEEE Int. Phoenix Conf. on Computers and Communications*, March 1990, pp. 868–869.
[21] Falkowski, B.J. and Perkowski, M.A. "Family of all essential radix-2 addition/subtraction multi-polarity transforms: algorithms and interpretations in Boolean domain", *Proc. of 23rd IEEE Int. Symp. on Circuits and Systems*, May 1990, pp. 2913–2916.
[22] Falkowski, B.J. and Perkowski, M.A. "On the calculation of generalized Reed–Muller canonical expressions from disjoint cube representation of Boolean functions", *Proc. of 33rd IEEE Midwest Symp. on Circuits and Systems*, August 1990, pp. 1131–1134.
[23] Falkowski, B.J., Schaefer, I. and Perkowski, M.A. (1992) "Effective computer methods for the calculation of Rademacher–Walsh spectrum for completely and incompletely specified boolean functions", *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.* **CAD-11**(10), 1207–1226.
[24] Falkowski, B.J. and Chang, C.H. (1998) "Mutual conversions between generalized arithmetic expansions and free binary decision diagrams", *IEE Proc. Circuits Devices Syst.* **145**(4), 219–228.
[25] Falkowski, B.J., Shmerko, V.P. and Yanushkevich, S.N. "Arithmetical logic—its status and achievements", *Proc. 4th Int. Conference on Applications of Computer Systems*, Szczecin, Poland, November 1997, pp. 208–223.
[26] Falkowski, B.J. (1999) "A note on the polynomial form of Boolean functions and related topics", *IEEE Trans. Comput.* **C-48**(8), 860–864.
[27] Falkowski, B.J. and Chang, C.H. (1999) "An efficient algorithm for the calculation of generalized arithmetic and adding transforms from disjoint cubes of Boolean functions", *VLSI Des., Int. J. Custom-Chip Des., Simulation Testing* **9**(2), 135–146.
[28] Falkowski, B.J. and Stankovic, R.S. (2000) "Spectral interpretation and applications of decision diagrams", *VLSI Des., Int. J. Custom-Chip Des., Simulation Testing* **11**(2), 85–105.
[29] Fujiwara, L. (1985) Logic Testing and Design for Testability (The MIT Press, Cambridge, MA).
[30] Hansen, J. and Sekine, M. "Synthesis by spectral translation using Boolean decision diagrams", *Proc. of 33rd ACM/IEEE Intl. Conf. on Design Automation*, November 1996, pp. 248–253.
[31] Hayes, J.P. (1976) "Transition count testing of combinational logic circuits", *IEEE Trans. Comput.* **C-25**(6), 613–620.
[32] Heidtmann, K.D. (1991) "Arithmetic spectrum applied to fault detection for combinational networks", *IEEE Trans. Comput.* **C-40**, 320–324.
[33] Hsiao, T. and Seth, S.C. (1984) "An analysis of the use of Rademacher–Walsh spectrum in compact testing", *IEEE Trans. Comput.* **C-33**(10), 934–938.
[34] Hurst, S.L. (1978) The Logical Processing of Digital Signals (Crane-Russak, London).
[35] Hurst, S.L., Miller, D.M. and Muzio, J.C. (1982) "Spectral method of boolean function complexity", *Electronics Lett.* **18**(13), 572–574.
[36] Hurst, S.L., Miller, D.M. and Muzio, J.C. (1985) Spectral Techniques in Digital Logic (Academic Press, London/New York).
[37] Hurst, S.L. (1989) "Use of linearization and spectral techniques in input and output compaction testing of digital networks", *IEE Proc. E* **136**(1), 48–56.
[38] Hurst, S.L. (1992) Custom VLSI Microelectronics (Prentice Hall International, Hertfordshire).
[39] Karkouri, Y. and Aboulhamid, E.M. (1990) "Complexite du Test des Circuits Logiques", *Techniques Sci. Inf.* **9**(4), 273–287.
[40] Karpovsky, M.G. (1976) Finite Orthogonal Series in Design of Digital Devices (Wiley, New York).
[41] Karpovsky, M.G., ed., (1985) Spectral Techniques and Fault Detection (Academic Press, Orlando, FL).
[42] Kinosita, K., Asada, K. and Karazu, O. (1985) Logic Design for VLSI (Iwanami Shoten Publishers, Tokyo), in Japanese.
[43] Landrault, C. (1995) Test, Testabilite et Test Integre des Circuits Integres Logiques (Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier, Universite de Montpellier).
[44] Lombardì, F. and Samì, M. (1987) Testing and Diagnosis of VLSI and ULSI (Kluwer Academic Publishers, Boston).
[45] Lui, P.K. and Muzio, J.C. (1986) "Spectral signature testing of multiple stuck-at faults in irredundant combinational networks", *IEEE Trans. Comput.* **C-35**(12), 1088–1092.
[46] Markowsky, G. (1981) "Syndrome-testability can be achieved by circuit modification", *IEEE Trans. Comput.* **C-30**(8), 604–606.
[47] Meinel, C., Somenzi, F. and Theobald, T. (2000) "Linear sifting of decision diagrams and its applications in synthesis", *IEEE Trans. Comput.-Aided Des.* **CAD-19**(5), 521–533.
[48] Miller, D.M. and Muzio, J.C. (1984) "Spectral fault signatures for single stuck-at faults in combinational networks", *IEEE Trans. Comput.* **C-33**(8), 760–765.
[49] Miller, D.M., ed., (1987) Developments in Integrated Circuit Testing (Academic Press, London).
[50] Miller, D.M. (1998) "An improved method for computing a generalized spectral coefficient", *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.* **CAD-17**(3), 233–238.
[51] Muzio, J.C. (1978) "Evaluation of the spectra of sum and product functions", *IEE Proc. Comput. Digital Techniques* **1**, 45–48.
[52] Muzio, J.C. (1980) "Composite spectra and the analysis of switching circuits", *IEEE Trans. Comput.* **C-29**, 750–753.
[53] Muzio, J.C., Miller, D.M. and Hurst, S.L. (1982) "Number of spectral coefficients necessary to identify a class of Boolean functions", *Electronics Lett.* **18**, 577–578.
[54] Muzio, J.C. and Miller, D.M. (1983) "Spectral fault signatures for internally unate combinational networks", *IEEE Trans. Comput.* **C-32**, 1058–1062.
[55] Muzio, J.C. (1985) "Data compression techniques for digital testing", *Congressus Numerantium* **46**, 211–236.
[56] Muzio, J.C. "Spectral methods for logic design and testability", *Proc. of 22nd IEEE Int. Symp. on Circuits and Systems*, 1989, pp. 752–755.
[57] Rahardja, S. and Falkowski, B.J. (1999) "Application of linearly independent arithmetic transform in testing of digital circuits", *Electronics Lett.* **35**(5), 363–364.
[58] Rahardja, S. and Falkowski, B.J. (1999) "Fast linearly independent arithmetic expansion", *IEEE Trans. Comput.* **C-48**(9), 991–999.
[59] Reddy, S.M. (1972) "Easy testable realizations for logic functions", *IEEE Trans. Comput.* **C-21**, 1183–1188.
[60] Reddy, B.R.K. and Pai, A.L. (1988) "Reed–Muller transform image coding", *Comput. Vision, Graphics, Image Process.* **42**, 48–61.
[61] Robinson, J.P. and Saxena, N.R. (1987) "A unified view of test compression methods", *IEEE Trans. Comput.* **C-36**(1), 94–99.
[62] Ruiz, G., Michell, J.A. and Buron, A. "NMOS and CMOS physical faults diagnosis by spectral techniques: simulation and associated tools", *IEEE Workshop on Languages for Automation*, The Technical University of Vienna, Aug. 1987, pp. 191–194.
[63] Ruiz, G., Michell, J.A. and Buron, A. "Fault detection and diagnosis for MOS circuits from Haar and Walsh spectrum analysis: on the fault coverage of Haar reduced analysis", *Proc. 3rd Int. Workshop on Spectral Techniques*, University of Dortmund, Oct. 1988, pp. 97–106.
[64] Ruiz, G., Michell, J.A. and Buron, A. "Diagnosis of stuck-open faults in CMOS by spectral techniques with pseudo random excitation", *Proc. 7th IASTED Int. Symp. on Applied Informatics*, Switzerland, Feb. 1989, pp. 89–92.
[65] Ruiz, G., Michell, J.A. and Buron, A. "Automatic test vector set generation in the spectral testing of sequential circuits", *Proc. 40th ISMM Int. Symp. on Mini- and Microcomputers and Their Applications*, Switzerland, June 1990, pp. 55–58.
[66] Ruiz, G., Michell, J.A. and Buron, A. (1992) "Switch-level fault detection and diagnosis environment for MOS digital circuits using spectral techniques", *IEE Proc. E* **139**(4), 293–307.

[67] Russell, G. and Sayers, I.L. (1989) Advanced Simulation and Test Methodologies for VLSI Design (Van Nostrand Reinhold, London).
[68] Sasao, T., Fujita, M., eds, (1996) Representations of Discrete Functions (Kluwer Academic Publishers, Boston).
[69] Savir, J. (1980) "Syndrome-testable design of combinational circuits", *IEEE Trans. Comput.* **C-29**(6), 442–451.
[70] Savir, J. (1980) "Correction to syndrome-testable design of combinational circuits", *IEEE Trans. Comput.* **C-29**(11), 1012–1013.
[71] Savir, J. (1981) "Syndrome-testing of syndrome-untestable combinational circuits", *IEEE Trans. Comput.* **C-30**(8), 606–608.
[72] Savir, J., Ditlow, G.S. and Bardell, P.H. (1984) "Random pattern testability", *IEEE Trans. Comput.* **C-33**(1), 79–90.
[73] Savir, J. and Bardell, P.H. (1984) "On random pattern test length", *IEEE Trans. Comput.* **C-33**(6), 467–474.
[74] Savir, J. and McAnney, W.H. "On the masking probability with one's count and transition count", *Proc. on IEEE Int. Conference Comput.-Aided Des.*, 1985, pp 111–113.
[75] Serra, M. and Muzio, J.C. (1987) "Testing programmable logic arrays by sum of syndromes", *IEEE Trans. Comput.* **C-36**(9), 1097–1101.
[76] Serra, M. and Muzio, J.C. (1987) "Spectral testing of multiple-output circuits", In: Miller, D.M., eds, Developments in Integrated Circuit Testing (Academic Press, London), pp 115–146.
[77] Serra, M. and Muzio, J.C. (1988) "Space compaction for multiple-output circuits", *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.* **CAD-7**, 1105–1113.
[78] Smith, J.E. (1979) "Detection of faults in programmable logic arrays", *IEEE Trans. Comput.* **C-28**(11), 845–853.
[79] Smith, J.E. (1980) "Measures of effectiveness of fault signature analysis", *IEEE Trans. Comput.* **C-29**(6), 510–514.
[80] Smith, J.E. (1986) "Author's reply on comments on detection of faults in programmable logic arrays", *IEEE Trans. Comput.* **C-35**(10), 931.
[81] Stankovic, R.S., Stojic, M.R. and Stankovic, M.S. (1996) Recent Developments in Abstract Harmonic Analysis with Applications in Signal Processing (Nauka, Belgrade).
[82] Susskind, A.K. (1983) "Testing by verifying Walsh coefficients", *IEEE Trans. Comput.* **C-32**, 198–201.
[83] Thornton, M.A. and Nair, V.S.S. (1995) "Efficient calculation of spectral coefficients and their applications", *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.* **CAD-14**(11), 1328–1341.
[84] Tokmen, V.H. "Disjoint decomposability of multivalued functions by spectral means", *Proc. of 10th Int. Symp. on Multiple-Valued Logic*, 1980, pp. 88–90.
[85] Tzidon, A., Berger, I. and Yoeli, Y.M. (1978) "A practical approach to fault detection in combinational circuits", *IEEE Trans. Comput.* **C-27**, 968–971.
[86] Wang, F.C. (1991) Digital Circuit Testing, A Guide to DFT and Other Techniques (Academic Press, San Diego).
[87] Williams, T.W. and Parker, K.P. (1983) "Design for testability—a survey", *IEEE Trans. Comput.* **C-31**(1), 2–15.
[88] Yamada, T. "Syndrome-testable design of programmable logic arrays", *Proc. Conference IEEE Int.*, 1983, pp 453–458.
[89] Yarmolik, V.N. (1990) Fault Diagnosis of Digital Circuits (Wiley, Chichester).

Author's Biography

Bogdan J. Falkowski received the MSEE degree from Technical University of Warsaw, Poland and the Ph.D. degree in Electrical and Computer Engineering from Portland State University, Oregon, USA. His industrial experience includes research and development positions at several companies. He then joined the Electrical and Computer Engineering Department at Portland State University. Since 1992 he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University in Singapore where he is currently an Associate Professor. His research interests include VLSI systems and design, switching circuits, testing, and design of algorithms. He specializes in the design of digital circuits with the use of spectral methods and has published 3 book chapters and over 150 refereed journal and conference articles in this area. He is a senior member of the IEEE, member of international advisory committee for International Conference on Applications of Computer Systems and technical chair for IEEE International Conference on Information, Communication and Signal Processing held in December 1999 in Singapore.