Fault Detection in Multiprocessor Systems and Array Processors

Mark G. Karpovsky, *Fellow, IEEE*, Tatyana D. Roziner, and Claudio Moraga

Abstract — **Off-line testing of large multiprocessor networks or VLSI chips with many outputs requires a large volume of memory for reference data storage. Space compaction combined with time compression of test responses can essentially reduce an overhead required for testing and diagnosis. In this paper, we discuss the problem of optimal design for space compressors (compactors), to minimize the number of observation points for detection of single faulty components in multiprocessor networks. A space compactor is assumed to be followed by a time compressor, to detect a fault not necessarily manifesting itself for a single test pattern.**

We formulate the rules of design for a space compaction matrix for the topology of the circuit-under-test (CUT) modeled by an arbitrary acyclic graph. Tree arrays and Fourier transform networks are considered as examples. The lower and upper bounds on the number of space compactor outputs are obtained, and optimal space compaction matrices are determined for above mentioned CUT topologies. Simple procedures for design of off-line testing devices with built-in self-testing are presented. Estimations on a complexity of proposed designs are given.

Index Terms — **Fourier transform networks, fault detection in VLSI devices, data compression (compaction), integrated circuits testing, off-line testing; VLSI devices testing.**

I. INTRODUCTION

IT is hardly possible to overestimate the importance of the problem of hardware testing. With the development of advanced technologies for VLSI chip manufacturing, the problem of testing becomes more important and also much more complicated. As a result, considerable effort is being devoted to the development of efficient and reliable methods of testing (see e.g., [6], [18], [19]). Built-in off-line self-testing (BIST) becomes in many cases a necessary feature of a VLSI chip. Evidently the area needed for the checking circuitry and for the storage of reference data should be minimized. This requirement gives rise to testing techniques based on reducing the amount of test response data (response compression or response compaction, [3], [8], [9], [10], [12], [13], [14], [15], [21], [22], [24], [25], [26], [28], [29], [30], [31], [32], [34]).

Manuscript received March 25, 1992; revised February 8, 1994.

This work has been partially supported by the National Science Foundation under Grant MIP-9208487 and the NATO under Grant 910411.

M. Karpovsky and T. Roziner are with Boston University College of Engineering, Department of Electrical, Computer, and Systems Enginering, Laboratory for Design and Testing of Computer and Communications Systems, 44 Cummington Street, Boston, MA 02215 USA; e-mail mr@enga.bu.edu. Prof Roziner can be contacted via e-mail at roziner@acs.bu.edu.

C. Moraga is with the University of Dortmund,Department of Computer Science, Dortmund, Germany; e-mail moraga@jupiter.informatik.uni-dortmund.de.

IEEECS Log Number C95006

The general structure for off-line testing by a space-time compaction of test responses essentially follows that of Fig.1.

The circuit-under-test (CUT) receives test patterns from the test generator (TG); the N outputs of the CUT are space-compacted to a smaller number $r < N$ in the space compactor (SC). The outputs of SC are compressed in time by the time compressor (TC) into r signatures and compared in the comparator (Δ) with the reference signatures (pre-stored in the memory reference block (MRB)). The space compactor (SC) is a combinational circuit and the TC is a sequential one.

Time and space compressors described in literature are transition- and edge-counting compressors [8], [21], linear space compressors [24], linear feedback shift register (LFSR) signature analysers/compressors [7], [12], [13], [14], [26], accumulator-based parallel compactors [6], [25], [28], [30]; also, compressors using spectral methods have been developed [10], [15], [31]. Cellular automata (linear finite state machines) have been proposed recently as an alternative for LFSRs for signature analysis [3], [29], [34]. The choice of a certain compaction method depends on the tradeoffs between cost and the percentage of fault coverage (aliasing).

The model of a network that might contain a faulty element (a graph where the nodes represent processing elements (chips, boards, computer nodes, etc.), and the edges are the communication links) is rather general. If the circuit-under-test (CUT) is a standard chip manufactured in large quantities, with a certain percentage of defective units that should be rejected, then the task of the efficient fault detection becomes probably even more important than fault location.

In this paper we will consider the problem of optimal design of space compactors (SCs) for fault detection for off-line testing. (Time compressors following space compactors can be implemented by standard MISRs [22]). The major problem in the design of SCs is the problem of minimization of the number of outputs, r, of an SC for a given network, in such a manner that any single fault that manifests itself in the non-compacted output(s), can be detected also by the circuit with the space compactor. We present a solution of this problem for tree arrays of processing elements and for FFT networks. The proposed optimal SCs are linear and can be implemented by XOR gates only. For FFT networks, the proposed checking circuit requires smaller overhead than e.g., the design of [32] based on the use of multipliers. The gain in complexity of the required overhead (compared to the conventional straightforward method of checking of all outputs of a circuit) can be as high as $N/(\log_2 N)$ for a large size FFT.

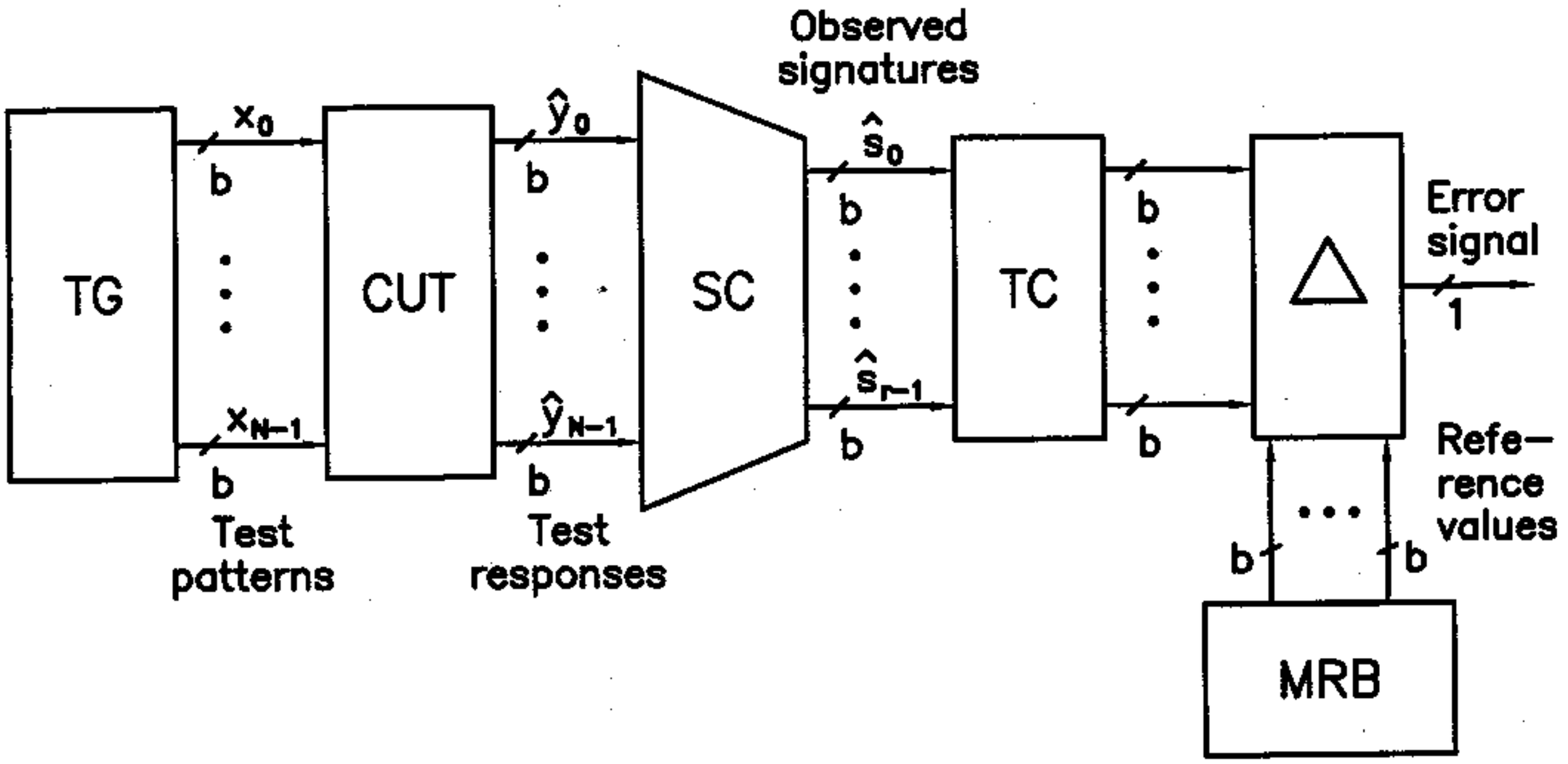

Fig.1. Block diagram of a space-time compactor for off-line testing.

II. THE PROBLEM STATEMENT

Consider a directed acyclic graph G which represents a network of processing elements (PEs). The PEs are located at the nodes, and the directed edges of the graph represent b-bit communication links. Let G have N_I input nodes (nodes with outgoing edges only) and N output nodes (nodes with only the edges entering these nodes). In this paper, $N_I = N$ in most cases. We shall assume that the following is valid for the graph:

1) The graph G has depth d (i.e., the maximum number of edges in a path from an input node to an output node is d–1).
2) Any output node is reachable from at least one input node.
3) At most one PE in the system (or any number of incoming lines to one PE) may be faulty.
4) We shall consider *below single additive errors in linear systems* (with acyclic graph topology); *only detectable single faults* at the system nodes will be considered. A detectable fault in a node (i.e., in a PE) is the one that manifests itself in at least one output node and for at least one test pattern from the test pattern set. In other words, any error of detectable type propagates to an output node level. This assumption is reasonable for the case when the network is tested by a large number of randomly chosen test patterns.

We shall assume that any error at a node propagates to all successive nodes. Since the communication links are multibit (b-bit, e.g., $b = 32$) lines, the probability of error masking in all bits is vanishingly small.

Certain assumptions should be done as to the structure of the nodes. As mentioned above, the nodes are assumed to be processing elements (devices, computers, processors, etc.) Our model describes the network (which is actually a CUT) on a system level, and our approach does not depend on the gate implementation of a specific node. This node level model is widely accepted in the literature. We do not consider the case when the nodes of G are just gates.

Let the N outputs of graph G (we shall use this loose notation for the outputs of the network with topology modeled by graph G) be $y = (y_0, y_1, ..., y_{N-1})^{Tr}$, i.e., y is a vector-column of the output values in the fault-free system, and let $\hat{y} = (\hat{y}_0, \hat{y}_1, \ldots \hat{y}_{N-1})^{Tr}$ be the output vector distorted as a result of a fault in one of the PEs (or incoming lines to the PE); $y_j, \hat{y}_j \in \{0, 1, ..., q-1\}$, where $q = 2^b$. The last statement means that $y_j, \hat{y}_j$ are b-bit binary values (fault-free and faulty, respectively) of the j th output PE; $j = 0, 1, ..., N-1$.

Let $\oplus$ denote bitwise (for vectors, also componentwise) modulo 2 addition. For $y \oplus \hat{y}$ vector, consider its support (that is, a vector with i th coordinate equal to 0 if $y_i = \hat{y}_i$ and equal to 1 otherwise) and call it error pattern, e. All possible error patterns for a given graph G (and single fault case) constitute the error set, E.

Example 1. Let $N = 3$, $b = 4$ (three 4-bit outputs); let $y = (0101, 1101, 1111)^{Tr}$ and $\hat{y} = (0101, 1110, 0010)^{Tr}$; then $y \oplus \hat{y} = (0000, 0011, 1101)^{Tr}$ and $e = (0, 1, 1)^{Tr}$, that is, second and third outputs are incorrect.

A linear binary space compactor (SC) can be described by an $(r \times N)$ binary matrix H (we shall call H a "space compac-

tion" matrix). The outputs of the compactor form vector of signatures, $\hat{s}$, where

$$\hat{s} = Hy = (\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{r-1})^{Tr},$$

$$\hat{s}_i \in \{0, 1, \ldots, q-1\}$$

(see Fig.1); all additions and multiplications are modulo 2 operations. For example, if $N = 3$, $b = 2$, $\hat{y} = (01, 11, 01)^{Tr}$, $r = 2$, and

$$H = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

we have

$$\hat{s} = H\hat{y} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

The necessary and sufficient condition for error detection by an *SC* is $Hy \neq H\hat{y}$ for all y and all $\hat{y}$ generated by single faults in the network. Thus, the problem to be solved is the problem of optimal space compaction in a single fault detection task: for a given topology of directed acyclic graph G with N outputs, construct the error-detecting matrix H with N columns and minimal number of rows $r \leq N$, such that for any error pattern $e \in E$, the vector He has at least one nonzero component. Moreover, to eliminate the possibility that two or more "secondary" errors in the outputs (induced by the same single fault at an internal node) will compensate and produce no distortion in s, we require that for any $e \in E$, there exists at least one row h of matrix H such that vector $H \bullet e$ has only one nonzero component ($\bullet$ is componentwise multiplication of binary vectors h and e; multiplications and additions are mod 2 operations, componentwise for vectors and bitwise for a component). The problem of single fault detection in graph G with N outputs is closely related to the problem of construction of a q-ary linear error-detecting code of the length N with $(N-r)$ information digits and r check digits detecting a given set of errors, E (defined by graph G); H is the parity-check binary matrix for the code [16]. If E is a set of all binary vectors with at most l 1s, then our problem is a classical and difficult problem of construction of a best q-ary linear code detecting l errors with a minimal number of check digits [16].

Note that from the viewpoint of minimization of the hardware needed for space compaction, it is advantageous to have an error-detecting matrix H with binary elements since in that case only XOR gates are needed for space compaction (see Section IV).

III. Fault Detection In Tree Structures

The binary tree structure is a model for a number of hierarchical computing systems (e.g., dictionary and searching machines [2], [27]). Many concurrent algorithms can be mapped onto a binary (or a p-ary) tree; the architecture of a general purpose multiprocessor concurrent computer can be modeled by tree structures [17]. Therefore the interest in reliable implementation of tree architectures is justified. A good survey of different approaches to fault-tolerant binary tree structures can be found in [9].

We are interested in finding the structure of a space compactor with a *minimal* number of outputs, r, to detect any single faulty node in the tree of depth d. The space compaction matrix $H_G(d)$ should have r rows that are *linearly independent*, that is, no one of the rows of $H_G(d)$ can be obtained as a sum modulo 2 of other rows (the coefficients in the linear combination can be either 0s or 1s). The maximum number of linearly independent rows (or columns) in $H_G(d)$ is the rank of this matrix over Galois field GF(2^b). Only $r < N$ independent outputs of space compactor will be monitored, instead of N outputs of the tree. We are therefore interested in minimizing r (the number of space compactor outputs and the number of rows in $H_G(d)$), and we would like to have the rank of $H_G(d)$ equal to r.

Consider a directed acyclic graph with the levels numbered 0 to $d-1$ (level $d-1$ is the level of output nodes). A single fault at any node (or at any number of incoming links to the node) results, under the assumptions of Section II, in a certain error pattern, e, with 1s in the positions corresponding to the nodes of the last level with distorted outputs. We shall call two error patterns e_1, e_2 non-intersecting if $e_1 \bullet e_2 = 0$ ($\bullet$ is componentwise multiplication). The following theorem gives the upper bound for the number of rows in $H_G(d)$.

Theorem 1. Let G be a directed acyclic graph with d levels where only single-node faults (or faults at any number of the incoming lines to one node) can occur. If for any two nodes of the same level, their error patterns either coincide or do not intersect, then the minimal number of rows, r, of the space compaction matrix $H_G(d)$ which detects any single detectable fault, is less or equal to d.

Proof. To detect a single fault in any output node, the row of all 1s in $H_G(d)$ is sufficient. Consider all the nodes of the level i, $0 \leq i \leq d-2$. A faulty node of this level implies a certain output error pattern, e^*, with component numbers $j_0, j_1, \ldots, j_k$ equal to 1 (the rest are 0s). To detect the faulty node, the space-compaction matrix must contain at least one row h such that $e^* \bullet h$ ($\bullet$ is componentwise multiplication) has only one nonzero component.

Assume that there are g_i distinct and non-intersecting error patterns for the faults in the nodes of i th level. If h is chosen as a vector with g_i 1s in such a manner that h intersects with each of g_i error patterns in a single component only, then the row h will detect all the single-node faults of the i th level. In other words, it is enough to choose a single 1 in each one of the g_i non-intersecting subsets of 1s in the output error patterns for level i. The rest of the components of row h are to be taken as 0s. The resulting N-tuple will be a row of $H_G(d)$ detecting the faults at the nodes of the i th level. Since the error patterns for different levels of G may coincide, the number of rows, r, in $H_G(d)$ is less or equal to d.

A simple procedure for constructing a space compaction matrix $H_G(d)$ can be suggested as shown below.

- *Step 1*: Construct a set, E, containing all possible distinct

error patterns resulting from the single-node faults for a given acyclic graph G (modeling the array of PEs). An error pattern is an N-bit binary vector (N is the number of output nodes), and the i th bit of error pattern is 1 if and only if the error manifests itself at the i th output of the graph.

- *Step 2*: Partition the set E into the minimum number of subsets, r, where the 1s in any two elements of a subset do not intersect.
- *Step 3*: For each element in a subset, if the number of 1s in this element is j and $j > 1$, then arbitrarily replace $j-1$ 1s to 0s.
- *Step 4*: Construct a set, H, which contains r vectors. Each vector in H is corresponding to a subset of the partition, and it is formed by bitwise XOR (vector XOR) on all elements in the subset. Each element of H will be used as a row vector in space compaction matrix $H_G(d)$.

In a sense, the procedure described above is more general than the result of Theorem 1 since it does not require that the partitioning of E into subsets be done according to the node level. It is evident that the space compaction matrix for a given graph might be not unique.

Example 2. Consider a four-point FFT graph of depth 3 with 12 nodes (see Fig. 2). There are seven distinct error patterns for single-node faults. For any input node fault, the error pattern is $(1111) = 1^4$; two patterns $(1100) = 1^2 0^2$ and $(0011) = 0^2 1^2$ for the middle level do not intersect as well as the four patterns with a single 1 for the output nodes. The space compaction matrix for this graph can be selected as

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

A balanced p-ary tree of the depth d is a directed graph with a single input node (the root) and p outgoing edges for each node except of the output nodes; the number of output nodes is p^{d-1}. The following theorem states that under the following rather weak condition (Equation (1)), we have $r \geq d$ for a p-ary tree, that is, one cannot go below d in the number of rows of space-compaction matrix for a p-ary tree topology.

Theorem 2. Let graph G be a p-ary tree of depth d having $N = p^{d-1}$ output nodes, and let the outputs of the system represented by graph G have values b from the Galois field $GF(q)$ where $q = 2^b$ (b-bit output values). Then any space compaction matrix $H_G(d)$ with r rows and N columns for the tree with d levels G has a rank at least d (i.e., $r \geq d$) if

$$d \leq \frac{1}{\log_p\left(1+(q-1)^{-1}\right)} \quad , \quad q = 2^b. \qquad (1)$$

The proof of Theorem 2 is given in Appendix A. We note that (1) is satisfied for most practical cases (e.g., $b = 32$, $q = 2^{32}$).

Corollary. It follows from Theorems 1 and 2 that for a p-ary tree such that (1) is satisfied we have $r = d = 1 + \log_p N$; N is the number of tree outputs, r is the number of (independent) rows in space-compaction matrix, and d is the tree depth.

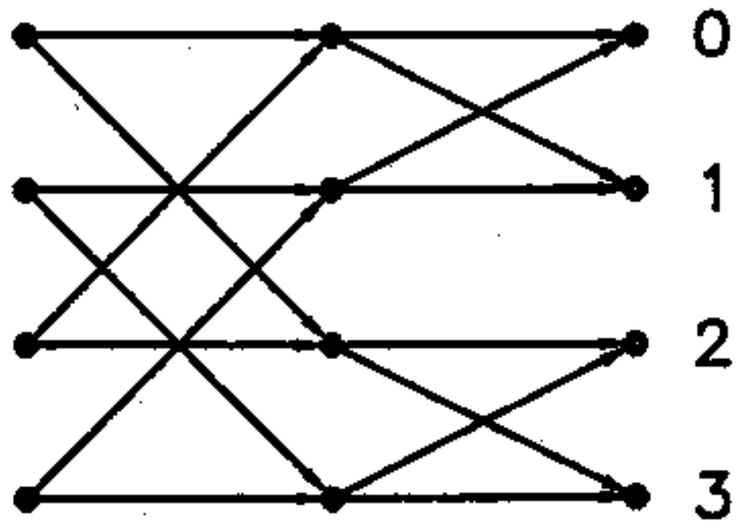

Fig. 2. Example of a graph with error patterns that either coincide or do not intersect for any two nodes of the same level.

Example 3. For a binary tree of depth $d = 4$ (with $2^{d-1} = 8$ output nodes), the space compaction matrix $H(4)$ has four rows and eight columns:

$$H(4) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

If the columns of $H(4)$ are numbered 0 to 7 starting from the left, it can be seen that there are four linearly independent columns in $H(4)$, say numbers 0, 1, 2, and 4. Using (1) it can be obtained that for $q > 6$ (i.e., more than 2-bit output values), it is impossible to construct $H(4)$ with less than four rows.

Example 4. This example illustrates that in the case when (1) is not satisfied, it is possible to construct the space-compaction matrix with the number of rows (and rank) less than the depth of a tree. Let $p = 2$, $d = 3$, $q = 2$ (a binary tree of depth 3, see Fig. 3, with the system outputs that can have values 0 or 1 only). The condition (1) is not met:

$$d = 3 > 1/(\log_2 2) = 1.$$

There are seven distinct error patterns: (10^3), (010^2), $(0^2 10)$,

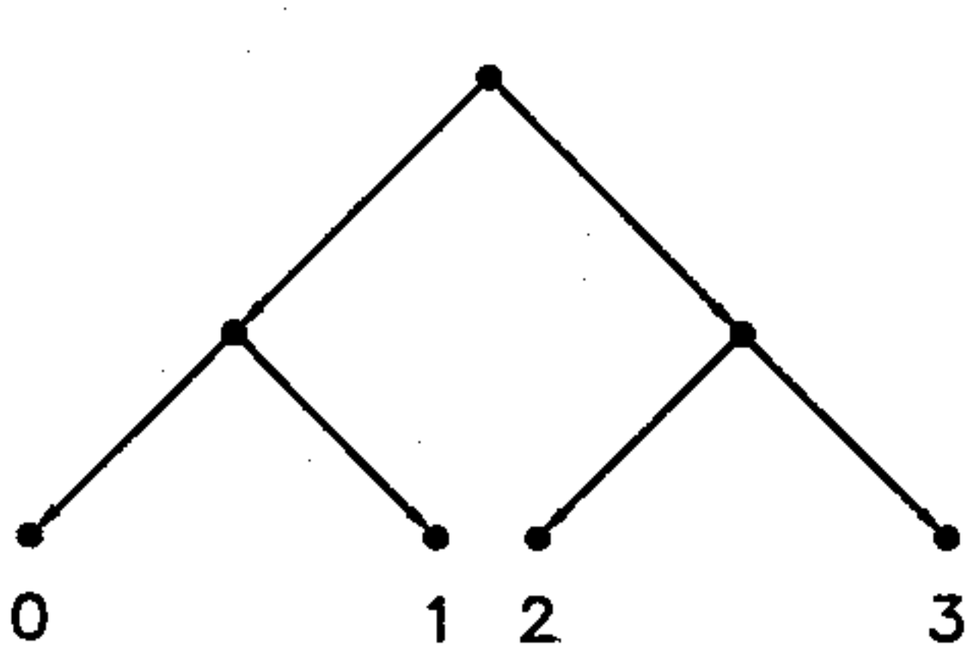

Fig.3. A binary tree of depth 3 (p = 2, d = 3) for Example 4.

(0^31), (1^20^2), (0^21^2), and (1^4), and the space compaction matrix $H(3)$ can be constructed with two rows only:

$$H(3)=\begin{bmatrix}1 & 1 & 0 & 1\\ 1 & 0 & 1 & 1\end{bmatrix}$$

The general rule of constructing $H(d+1)$ for a p-ary tree, if $H(d)$ is known and (1) is satisfied, is as follows: $H(d)$ (having d rows) is repeated p times, and one more row (the last one having a single 1) is added:

$$H(d+1)=\begin{bmatrix}H(d) & | & H(d) & | & \cdots & | & H(d)\\ --- & | & --- & | & --- & | & ---\\ 1\cdots0 & | & 0\cdots0 & | & \cdots & | & 0\cdots0\end{bmatrix}$$

As a special case of p-ary tree, the star topology of graph G needs H-matrix of two rows only as specified above. For the single fault case in a star graph, instead of checking all $N = p$ output values, it is sufficient to check only two functions: the sum of all the outputs and any single output.

IV. Fault Detection In Acyclic Graphs Containing Trees As Subgraphs

The results presented above for tree stuctures can be extended to other topologies of practical interest containing tree structures as subgraphs, since any single node fault manifests itself propagating through the graph in a treelike manner. The faulty node is the root of a tree subgraph of G, and the leaves of the tree are the distorted output nodes. We consider below several special important cases.

A. Fast Fourier Transform.

Consider the graphs corresponding to the flow diagrams of the fast Fourier transform (FFT) [5]. The processing elements (PEs) located at the nodes of an FFT graph are simple (multiply-add cells) but their quantity may be very large. For an N-point FFT, there are $N\log_2 N$ PEs if we do not consider the input nodes as processing elements (they are just fan-out points). However, for the purpose of generality it will be convenient to consider the input nodes also as PEs and consequently as possible sources of faults; in that case, there are $N(\log_2 N + 1)$ PEs in the whole graph. Note that an FFT flow diagram can be viewed as a structure composed of intersecting binary trees; therefore the results formulated for tree graphs in the previous section, are valid for FFT graphs.

For an N-point FFT, the number of graph levels is $d = 1 + \log_2 N$, and it will be shown below that the space compaction matrix $H(d)$ has d rows.

Evidently the space compaction based on $H(d)$ can be very efficient in the case of large size FFTs: instead of monitoring N outputs, one can check only $1 + \log_2 N$ outputs of the SC, and any single detectable fault will be detected.

The space compaction matrix $H(d)$ has slightly different forms for DIF FFT and DIT FFT (decimation-in-frequency and decimation-in-time fast Fourier transform, [5]) but the recursive rules to obtain $H(d + 1)$ are similar. Let $H_N(d)$ denote space compaction matrix for the N-point FFT; $d = 1 + \log_2 N$. It can be shown (as follows from the Procedure described in previous section) that the space compaction matrices for N-point DIF FFT and for N-point DIT FFT ($1 + \log_2 N$ rows, N columns) can be written as shown below:

Space compaction matrix $H_{N,DIF}(d)$:

Row 0	All 1s in the row:	(1^N)
Row 1	$N/2$ pairs (10):	$(10)^{N/2}$
Row 2	$N/4$ 4-tuples (1000):	$(1000)^{N/4}$
.....	.....	
Row (d–2)	Two $(N/2)$-tuples, each with a single 1 on the left:	$(10^{(N/2)-1})^2$
Row (d–1)	A single (leftmost) 1 in the row:	(10^{N-1})

(3)

Space compaction matrix $H_{N,DIT}(d)$:

Row 0	All 1s in the row	(1^N)
Row 1	$N/2$ and $N/2$ 0s:	$(1^{N/2}0^{N/2})$
Row 2	$N/4$ 1s on the left, the rest are 0s:	$(1^{N/4}0^{3N/4})$
.....	.....	.....
Row (d–2)	Two 1s on the left, the rest are 0s:	$(1^20^{(N-2)})$
Row (d–1)	A single (leftmost) 1 in the row:	(10^{N-1})

(4)

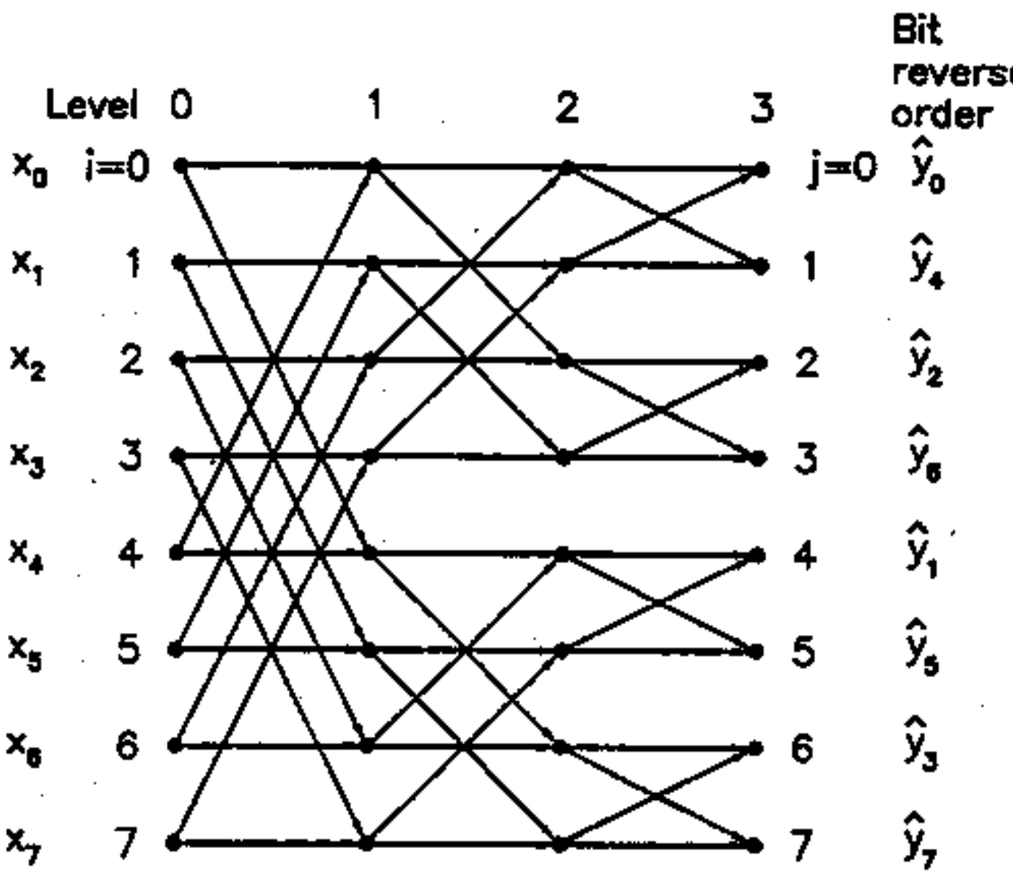

Fig. 4. The flowgraph of the eight-point DIF FFT.

Example 5. The graph for the eight-point DIF FFT is shown in Fig. 4. The error patterns of the eight-point DIF FFT implied by single-node faults are:

- (10^7), (010^6), . . . , (0^71) (level 3);
- (1^20^6), $(0^21^20^4)$, $(0^41^20^2)$, (0^61^2) (level 2);
- (1^40^4) and (0^41^4) (level 1);
- (1^8) (level 0).

For the eight-point DIT FFT, the error patterns are the same as in DIF FFT for levels 3 and 0. For level 2, there are patterns

$(10^3 10^3)$, $(010^3 10^2)$, $(0^2 10^3 10)$, and $(0^3 10^3 1)$. For level 1, the patterns are $(10)^4 = 10101010$ and $(01)^4 = 01010101$. Thus

$$H_{8,DIF}(4) = \begin{bmatrix} 1111 & 1111 \\ 1010 & 1010 \\ 1000 & 1000 \\ 1000 & 0000 \end{bmatrix} = \begin{bmatrix} 1^8 \\ (10)^4 \\ (10^3)^2 \\ 10^7 \end{bmatrix} \quad (5)$$

and

$$H_{8,DIT}(4) = \begin{bmatrix} 1111 & 1111 \\ 1111 & 0000 \\ 1100 & 0000 \\ 1000 & 0000 \end{bmatrix} = \begin{bmatrix} 1^8 \\ 1^4 0^4 \\ 1^2 0^6 \\ 10^7 \end{bmatrix}. \quad (6)$$

B. Walsh-Hadamard Transform.

The flowgraph of the Walsh-Hadamard Transform (WHT) [5] differs from the flow-graphs of FFT (Fig.4 and Fig. 5) only in the values of twiddle factors (±1s for WHT). Therefore the space compaction matrices for WHT flowgraphs are the same as for the FFT matrices (one of the two types can be chosen depending on the factorization of the fast WHT matrix).

V. Hardware Complexity Of Space Compressors For FFT-Networks

Consider the conventional method of testing of an FFT network for an N-point transform when each of the N b-bit outputs is compressed to an b-bit signature by a b-bit MISR and bitwise compared to b-bit pre-stored reference values. This method requires $2Nb$ flip-flops (N b-bit MISRs and Nb bits of storage for reference values, Nb gates for MISRs, Nb one-bit comparators (i.e., Nb two-input XORs), and $Nb-1$ two-input OR gates (See Fig. 6). Thus the complexity of a conventional testing circuit is:

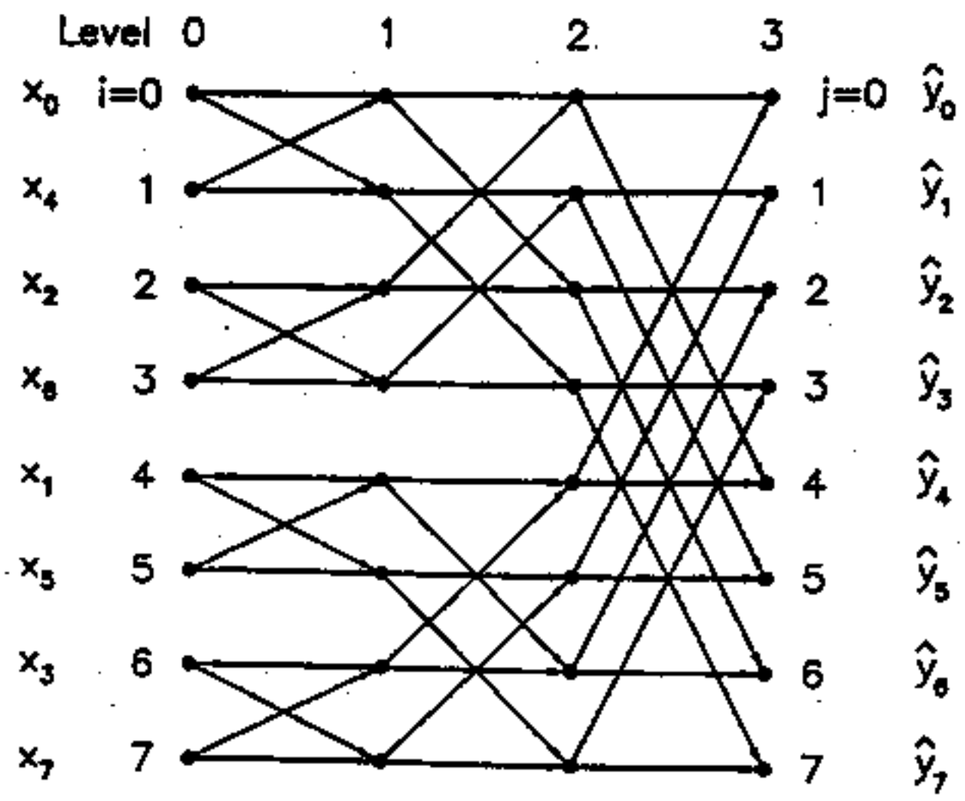

Fig.5. The flowgraph of the eight-point DIT FFT.

$$L_1 = 2Nb\, L_{ff} + (3Nb-1)\, L_g \quad (7)$$

where L_{ff} is a flip-flop complexity (in terms of two-input gates), and L_g is a complexity of a two-input gate.

For the design with space-compaction matrix H_G for N-point FFT graph, $N-1$ b-bit mod 2 adders can provide the needed $\log_2 N + 1$ outputs. For example, it can be seen (Fig.7) that for the eight-point DIT FFT, seven b-bit adders will perform the needed multiplication of the graph outputs by the matrix (5) which results in a signature $(\hat{s}_0, \hat{s}_1, \hat{s}_2, \hat{s}_3)$:

$$s_0 = \sum_{i=0}^{7} \hat{y}_i; \quad s_1 = \sum_{i=0}^{3} \hat{y}_i; \quad s_2 = \hat{y}_0 + \hat{y}_1; \quad s_3 = \hat{y}_0.$$

For this small size example (eight-point transform), one has to monitor four outputs $\hat{s}_0, \hat{s}_1, \hat{s}_2, \hat{s}_3$ instead of eight $(\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_7)$; in general case, the number of the signature components is $\log_2 N + 1$ which gives an essential gain compared to N. The proposed space-compaction design requires $b(\log_2 N + 1)$ storage cells, $b(\log_2 N + 1)$ flip-flops of b-bit MISRs, $b(\log_2 N + 1)$ gates of MISRs, $b(\log_2 N + 1)$ for b-bit XORs of comparators, $b(\log_2 N + 1) - 1$ two-input OR gates, and $N-1$ b-bit adders mod2 (see Fig. 7, $n = 8$). Each adder mod 2 is actually a set of b two-input XORs (bitwise modulo 2 addition), and the complexity of a space-compaction circuit for N-point FFT equals

$$L_2 = 2b(\log_2 N + 1)\, L_{ff} + [(3b(\log_2 N + 1) + (N-1)b - 1]\, L_g \quad (8)$$

Note that the number of mod 2 adders is minimal if a parallel space compressor is considered: no less than $N-1$ additions mod 2 are to be performed for any network with N outputs. For example, if $b = 32$, $N = 2^{10} = 1024$, and $L_{ff} = L_g$ (one gate complexity for a flip-flop), then $L_1 = 1.64 \times 10^5\, L_g$, $L_2 = 3.45 \times 10^4 L_g$.

In (8), the highest order term is the last one including the factor of N. This term arises due to the high number of adders mod 2 used in the parallel design of Fig. 7. For a large size FFT chip, the spectral components are generally obtained in a bit-parallel, component-serial manner (one component per clock; see e.g., [20]). Therefore it makes sense to use serial accumulation of the output values instead of parallel bitwise addition mod 2 and save significantly in the complexity of the checking circuit. If bitwise adders/accumulators mod 2 (T flip-flops) are used, their number can be $\log N_2 + 1$ only (see Fig. 8).

The controlling counter in Fig.8 will provide addition enables for $\log_2 N + 1$ adders for the accumulation of signature components. One of the adders/accumulators (computing s_3) will add all the components of the spectrum; the next one will add a value once in two counter clocks, the next one will add once in four counter clocks, etc. The complexity of such a design will be as follows:

$$L_3 = [3b(\log_2 N + 1) + \log_2 N)]\, L_{ff} + [3b(\log_2 N + 1) + 0.5(\log_2 N)^2 + 0.5\log N - 1]\, L_g \quad (9)$$

The complexity of (9) (see Fig. 8 for N = 8) will be of the order of $b\log_2 N$ for large N which provides an essential gain of the order $N/\log_2 N$ compared to the straightforward ap-

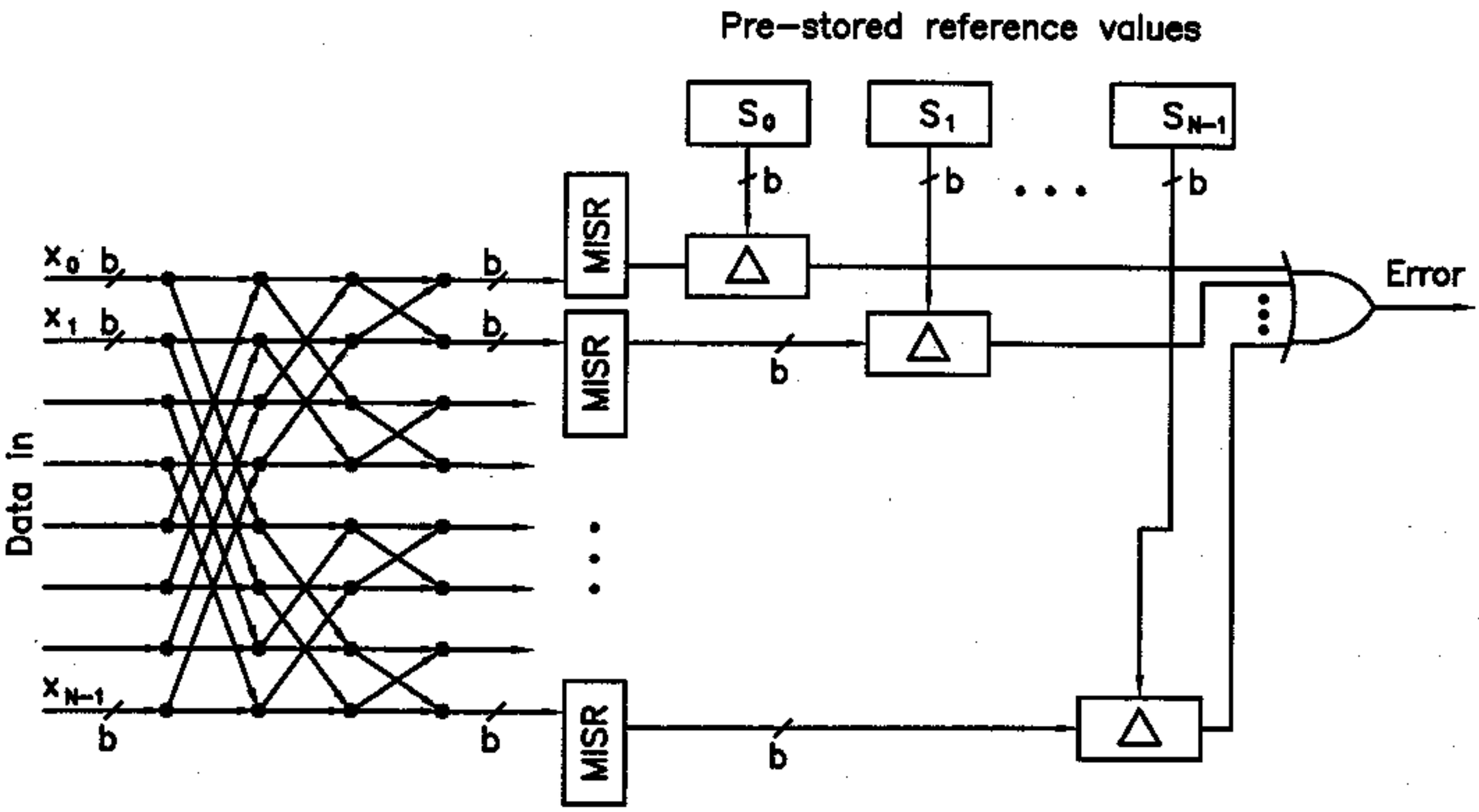

Fig. 6. A straightforward comparison of b-bit outputs of the eight-point DIT FFT to pre-stored correct signature values $s_0, s_1, \ldots, s_{N-1}$ ($N = 8$).

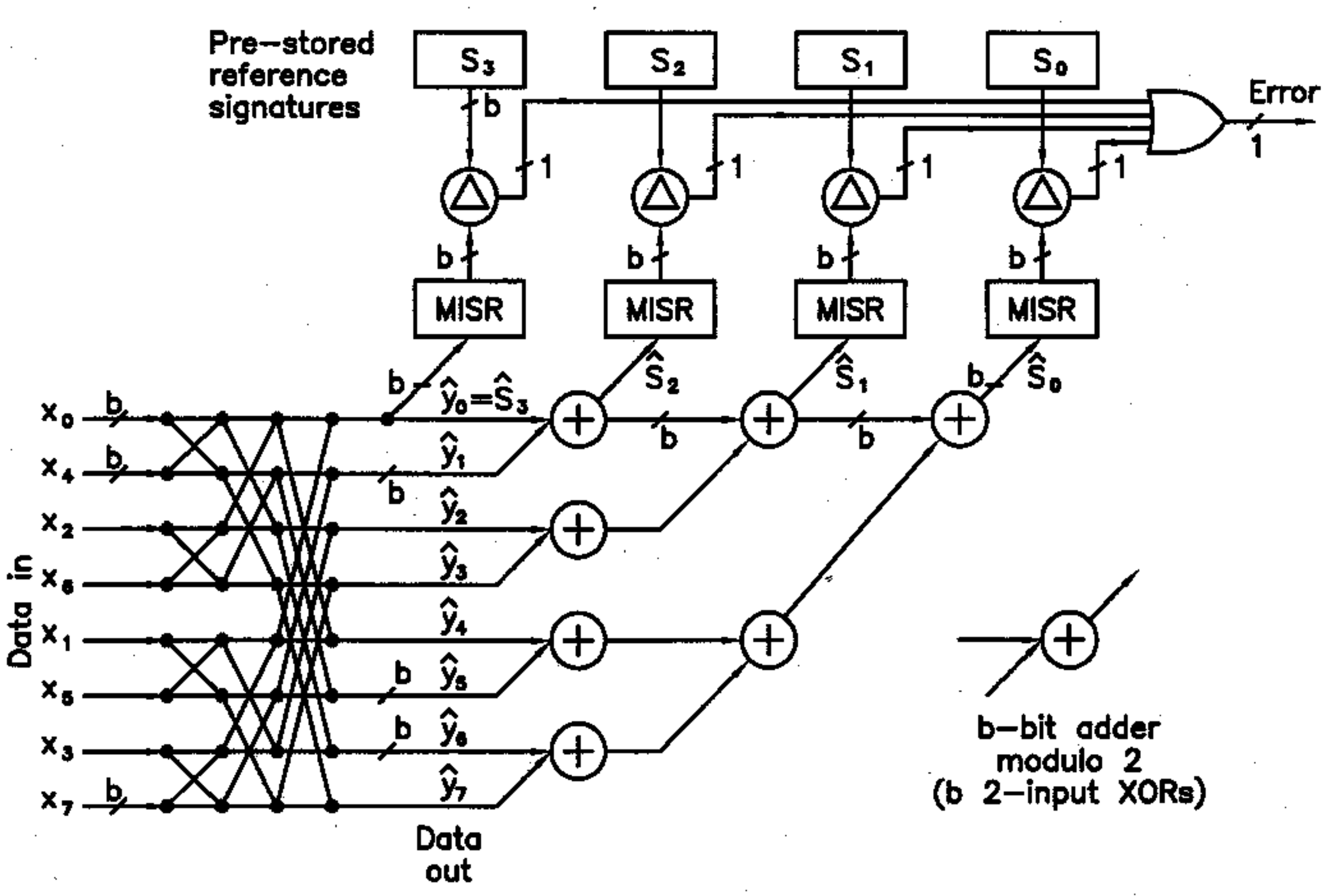

Fig. 7. The graph of the eight-point DIT FFT ($N = 8$) and $N - 1 = 7$ adders that perform parallel multiplication of the outputs of DIT FFT by space-compaction matrix $H_{8,\text{DIT FFT}}(4)$.

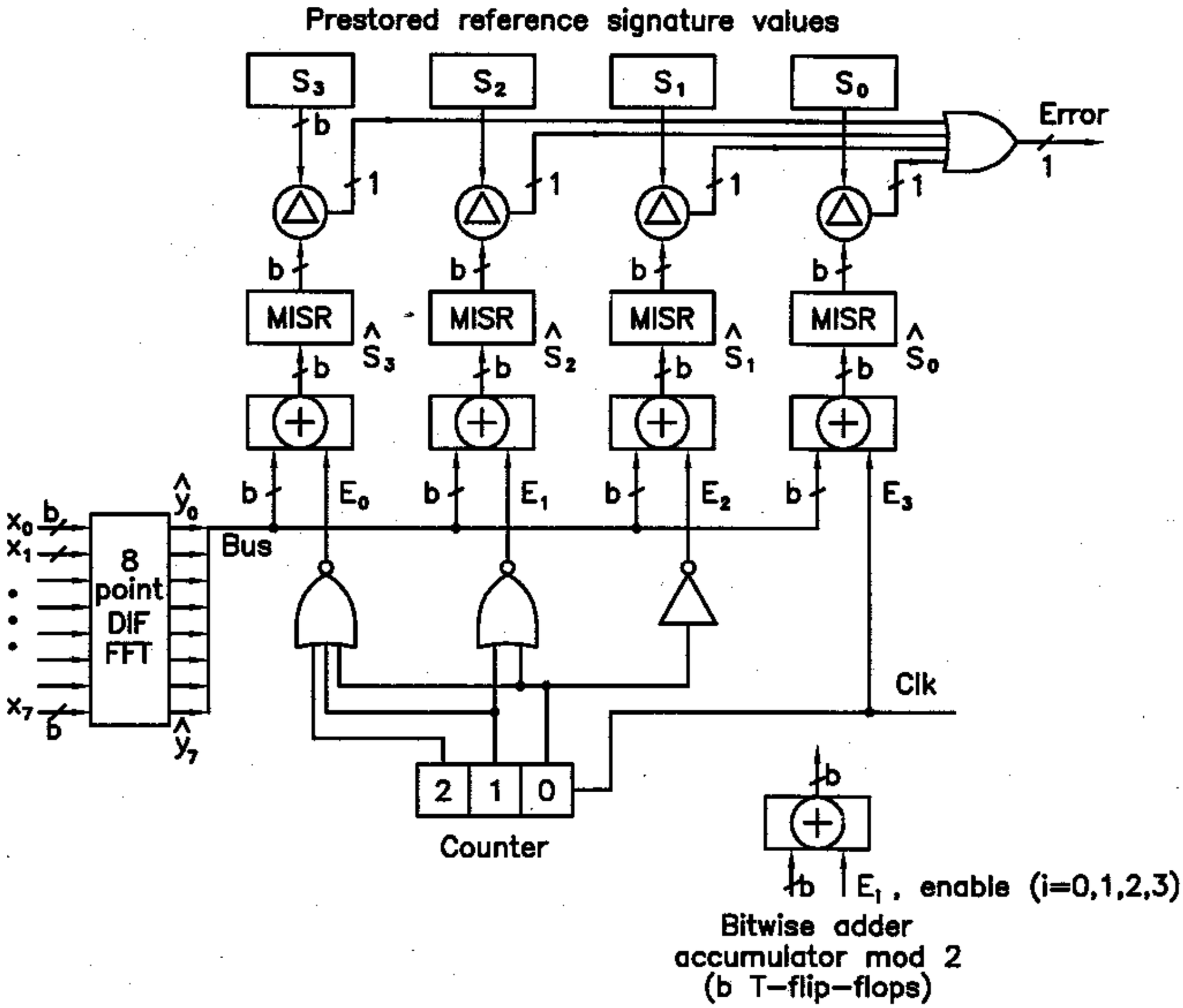

Fig. 8. The space-compaction circuit for the eight-point DIT FFT ($N = 8$) with $\log_2 N + 1 = 4$ adders/accumulators used for word-serial signature generation.

proach (7). For $b = 32$, $N = 2^{10}$, and $L_{ff} = L_{\hat{g}}$, $L_3 = 2.2 \times 10^3 L_g$ (whereas $L_1 = 1.64\ 10^5 L_g$; the gain $L_1/L_3 = 74.5$).

Fig. 9 illustrates the comparison of (7), (8), and (9) for $b = 16$.

VII. Conclusions

Optimal design for space compactors minimizing the number of the observation points for single fault detection in a linear circuit-undertest has been presented. The networks modeling the structure of chip-under-test are tree arrays and FFT networks; also, more general examples (directed acyclic graphs) have been considered. The estimation of hardware complexity of the proposed design of space compactors for FFT networks shows that for large size transforms, the gain in hardware complexity can be of the order $N/\log_2 N$ where N is the size of the transform.

VIII. Acknowledgments

The authors would like to thank Dr. L.B. Levitin (Boston University) for valuable discussions of the results presented in this paper. The useful comments of the reviewers are appreciated, especially the comments of the reviewer who suggested a constructive algorithm (procedure) for a space compaction matrix generation.

Appendix A. Proof of Theorem 2.

We shall prove Theorem 2 for binary trees; the proof can be easily extended to the case of p-ary trees. The theorem can be proved by induction. The theorem statement is correct for $d = 2$ (and $q > 2$, for a binary tree); in this case the matrix $H_G(2)$ has two rows, one consisting of 1s and the other having single 1 (the rest of the elements are 0s). Assume that the theorem statement is true for a tree $G(d)$ of depth d, i.e., that $\text{rank}(H_G(d)) \geq d$ when (1) is satisfied. We shall suppose that

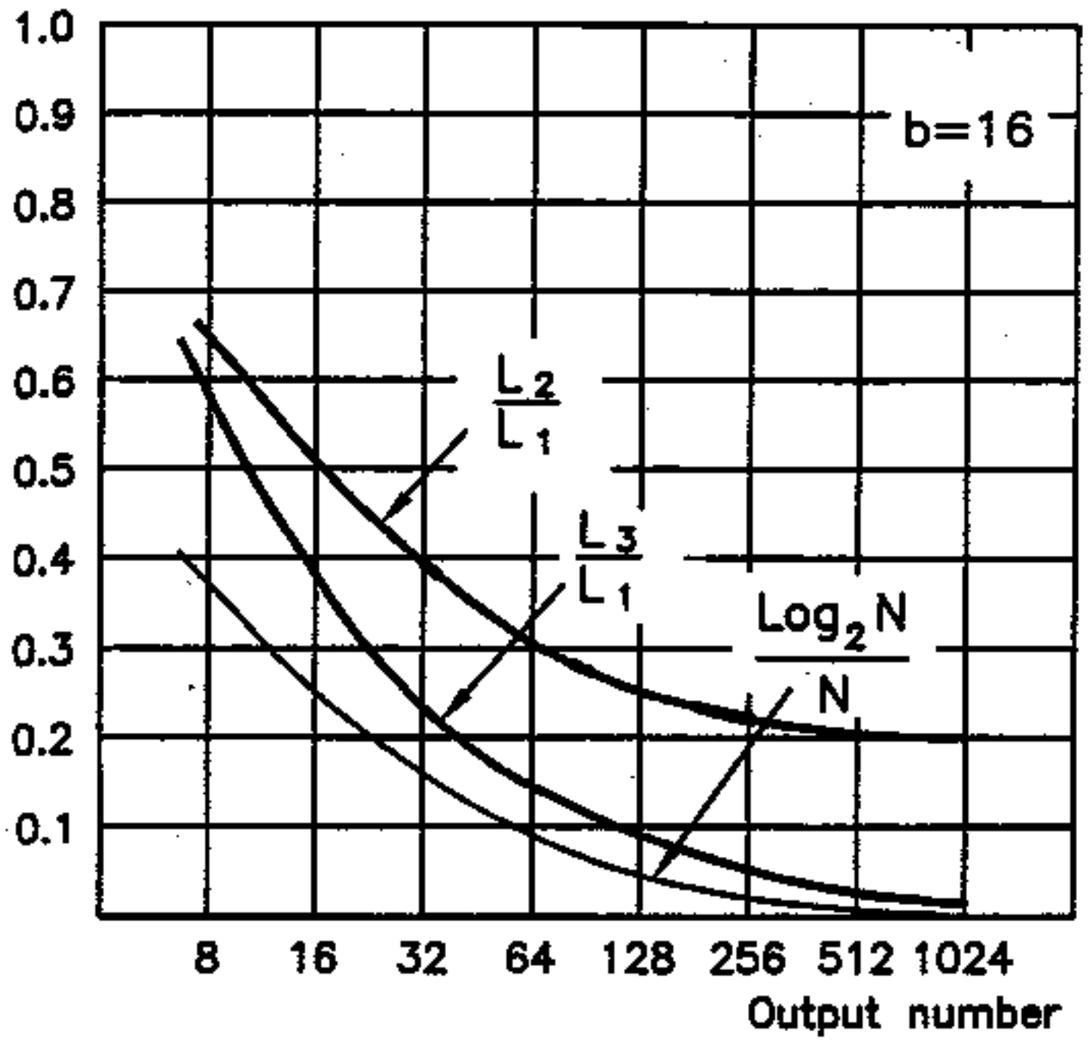

Fig. 9. The comparison of complexity of hardware needed for off-line testing for the proposed approach (L_2 for the design of Fig. 7 and L_3 for the design of Fig. 8) compared to the complexity L_1 of the straightforward testing hardware with compression in time only. L_1, L_2, and L_3 are the equivalent numbers of two-input gates for 16-bit data ($b = 16$).

rank($H_G(d + 1) = d$ while (1) is satisfied; it will be shown that this results in a contradiction.

We repeat below for reference purpose the statement of Theorem 1 (formula (Ap1), same as formula (1) of Section III:

$$d \le \frac{1}{\log_p\left(1+(q-1)^{-1}\right)} \quad , \quad q = 2^b \qquad \text{(Ap1)}$$

A binary tree $G(d + 1)$ of depth d + 1 can be constructed from two identical binary trees $G_1(d)$ and $G_2(d)$ and one additional node (the root) as shown in Fig. A-1.

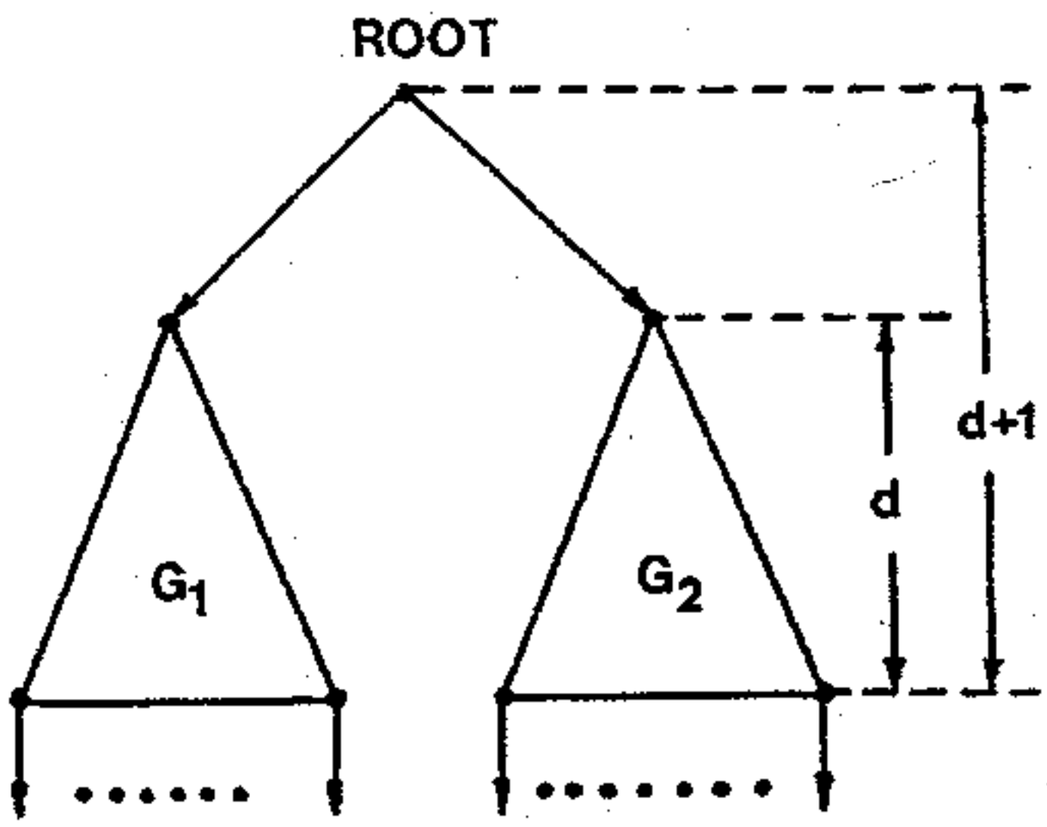

Fig. A-1. A binary tree of depth $d + 1$ constructed from two binary trees of depth d.

Let a space compaction matrix for the binary tree of depth d + 1 be H_G(d + 1) and let its rank be d. If we subdivide the matrix $H_G(d + 1)$ into two equal parts:

$$H_G(d+1) = \left[H_1(d) \vdots H_2(d)\right]$$

then $H_1(d)$ must be the space compaction matrix for $G_1(d)$, and $H_2(d)$ must be the space compaction matrix for $G_2(d)$. By assumption, rank($H_1(d)$) > d (as well as the rank of $H_2(d)$). Therefore each one of the matrices $H_1(d)$, H_2d) must have at least d linearly independent columns.

Consider now the case of a single fault in the root PE of $G(d + 1)$. All the outputs of G_1 and G_2 will be distorted; if e is any error vector (2^d-tuple), then the r-tuple $H_G(d + 1)$ e has at least one nonzero component since $H_G(d + 1)$ must detect any single-node fault in $G(d + 1)$. Consider the linear combination of all columns of $H_G(d + 1)$ taking the linearly independent columns of $H_1(d)$ and $H_2(d)$ with any nonzero coefficients

$$a_1^0,\ a_1^1,\ \ldots,\ a_1^{d-1},\ a_2^0,\ a_2^2,\ \ldots,\ a_2^{d-1}$$

(from the field $GF(q)$), and the rest of the columns of $H_G(d + 1)$ can be taken with some fixed coefficients, e.g., all 1s. This linear combination of columns of H_G(d + 1) must produce a nonzero vector.

Let h_1^i and h_2^i denote linearly independent columns in $H_1(d)$ and $H_2(d)$, respectively. Then

$$\bigoplus_{i=0}^{d-1} h_1^i a_1^i \oplus A_1 \neq \bigoplus_{i=0}^{d-1} h_2^i a_2^i \oplus A_2 \qquad \text{(Ap2)}$$

where A_1, A_2 are the vector sums of all the columns of matrices $H_1(d)$, $H_2(d)$ except of the d linearly independent ones. (All additions and multiplications are modulo 2 operations).

Since (Ap2) should hold for any choice of nonzero

$$a_1^0,\ a_1^1,\ \ldots,\ a_1^{d-1} \text{ and } a_2^0,\ a_2^1,\ \ldots,\ a_2^{d-1},$$

the inequality in (Ap2) means that there are two sets (with $(q-1)^d$ distinct d-component vectors in each set) in the d-dimensional space of a_j^i-coefficients, and no vector of one set can be found in the other set. This contradicts the assumption since from (Ap1) (same as (1) of Section III) for $p = 2$ we have $2(q-1)^d \ge q^d$.

For a p-ary tree, the proof is similar, only the logarithm in (1) is to be taken as a base-p one.

APPENDIX B. NOTATIONS (IN ALPHABETICAL ORDER)

b	Number of bits in the inputs and outputs of CUT, SC, TC.
d	Depth of acyclic graph modeling the topology of CUT.
e	Error pattern.
E	Set of all error patterns resulting from single faults in the CUT.
G	Acyclic graph modeling the topology of CUT.
g_i	The number of distinct no$N$-intersecting error patterns for the i th level of graph G.
H	The set of vectors (rows of compaction matrix).
$H_G(d)$	Compaction matrix for graph G of depth d.
L	Complexity of CUT.
ΔL	Complexity of testing hardware (except of test pattern generator).
L_1, L_2, L_3	Complexities of different off-line testing designs.
L_{ff}	Complexity of a flip-flop.

L_g	One gate complexity.
N	Number of the CUT outputs.
N_1	Number of CUT inputs.
q	Number of elements in the Galois field $GF(q)$; $q = 2^b$.
r	Number of space compactor outputs.
s, s_i	Signatures (fault-free).
$\hat{s}, \hat{s}_i$	Observed signatures.
T	The matrix of CUT.
Tr	Transposed vector (superscript).
$x_0, \ldots, x_{N-1}$	The inputs of CUT.
$y_0, \ldots, y_{N-1}$	Fault-free CUT outputs.
$\hat{y}_0, \ldots, \hat{y}_{N-1}$	Observed CUT outputs (test responses).
$\triangle$	Comparator block.

REFERENCES

[1] T. Beth. "*Verfahren der Schnellen Fourier-Transformazionen*," *Leitfaden der Angewandten Mathematik und Mechanik* LAMM, Bd. 61. Stuttgart: B.Teubner, 1984.

[2] J. Bentley and H.T. Kung. "A tree machine for searching problems." *Proc. Int'l. Conf. Parallel Processing*, pp.257-266, 1979.

[3] M. Damiani, P. Olivio, and B. Ricco. "Analysis and design of linear finite state machines for signature analysis testing," *IEEE Trans. Computers*, vol. 40, pp. 1034-1045, Sept. 1991.

[4] L. Dornhoff, *Group Representation Theory*, New York: Marcel Dekker, 1971.

[5] D.F. Elliott and K.R. Rao, *Fast Transforms: Algorithms, Analyses and Applications*, New York: Academic, 1982.

[6] H. Fujiwara, *Logic Testing and Design for Testability*, Cambridge, Mass.: M.I.T. Press, 1985.

[7] R.A. Frohwerk. "Signature Analysis: a new digital field services method." *Hewlett-Packard Journal*, pp.2-8, May 1977.

[8] J.P. Hayes. "Transition count testing of combinational logic circuits," *IEEE Trans. Computers*, vol. 23, pp. 613-620, June 1976.

[9] M.C. Howells and V.K. Agrawal. "A reconfiguration scheme for yield enhancement of large area binary tree architectures." *IEEE Trans. Computers*, vol. 37, pp.463-468, April 1988.

[10] T.C. Hsiao and S.C. Seth. "An analysis of the use of Rademacher-Walsh spectrum in compact testing." *IEEE Trans. Computers*, vol. 33, pp.934-937, Oct. 1984.

[11] M.G. Karpovsky. *Finite Orthogonal Series in the Design of Digital Devices*, New York: Wiley, 1976.

[12] M.G. Karpovsky and P. Nagvajara. "Board level diagnostic," *Proc. Int'l. Test Conf.*, 1988, pp.47-53.

[13] M.G. Karpovsky and P. Nagvajara. "Design of self-diagnostic boards by signature analysis," *IEEE Trans. Industrial Electronics*, vol. 36, pp. 241-245, February 1989.

[14] M.G. Karpovsky, L.B. Levitin, and F.S. Vainstein. "Diagnosis by signature analysis of test responses," *IEEE Trans. Computers*, vol. C-43, pp.141-153, February 1994.

[15] J.C. Muzio, D.M.Miller. "Spectral techniques for fault detection," *Proc. 12th Int'l. Symp. Fault Tolerant Computing*, pp. 297-302, 1982.

[16] F.J. MacWilliams and N.J.A. Sloane. *Theory of Error-Correcting Codes*, North - Holland Publishing, 1978.

[17] C.A. Mead and L.A. Conway. "Hierarchically organized machines." *Introduction to VLSI Systems*, Ch. 8, Reading, Mass: Addison - Wesley, 1980.

[18] E.J. McCluskey. "Built-in self-test techniques" and "Built-in self-test structures," *IEEE Design and Test of Computers*, pp. 21-36, April 1985.

[19] E.J. McCluskey. Digital test principles, Stanford Logical Systems Institute, 1991.

[20] J. O'Brien, J. Mather, and B. Holland. "A 200 MIPS single-chip 1K FFT processor," *IEEE Int'l. Solid-State Circuits Conf.*, 1989, THPM 12.5.

[21] S. Pilarsky and K. Wiebe. "Counter-based compaction: an analysis for BIST," *Technical Report CSS/LCCR TR 91-02*, Simon Fraser University, Feb. 1991.

[22] D.K. Pradhan, S.K. Gupta, and M.G. Karpovsky. "Aliasing probability in multiple input signature analyzer," *IEEE Trans. Computers*, vol. 37, pp. 1151-1156, Sept. 1988.

[23] T.D. Roziner, M.G. Karpovsky, and L.A. Trachtenberg. "Fast Fourier transforms over finite groups by multiprocessor systems." *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp.226-240, Feb. 1990.

[24] S.M. Reddy, K.K. Saluja, and M.G. Karpovsky. "A data compression technique for built-in self test," *Proc. of Int'l. Symp. on Fault - Tolerant Computing*, pp. 294-299, 1985.

[25] J. Rajski, J. Tyszer, and B. Salimi. "On the diagnostic resolution of signature analysis," *Proc. ICCAD'90*, Santa Clara, Calif, pp. 364-367, 1990.

[26] J. Smith. "Measure of effectiveness of fault signature analysis," *IEEE Trans. Computers*, vol. 29, pp.510-514, June 1980.

[27] A.K. Somani, V.K. Agrawal. "An efficient unsorted VLSI dictionary machine." *IEEE Trans. Computers*, vol. 34, pp.841-852, Sept. 1985.

[28] J. Savir, "Syndrome-testable design of combinational circuits," *IEEE Trans. Computers*, vol. 29, pp. 442-451, June 1980.

[29] M. Serra, T. Slater, J.C. Muzio, and D.M. Miller. "The analysis of one-dimensional linear cellular automata and their aliasing properties," *IEEE Trans. CAD*, vol. 9, pp. 767-778, July 1990.

[30] N.R. Saxena, J.P. Robinson. "Accumulator compression testing." *IEEE Trans. Computers*, vol. 35, pp.35-44, April 1984.

[31] A.K. Susskind. "Testing by verifying Walsh coefficients," *Proc. of 11th Int'l. Symp. on Fault-Tolerant Computing*, pp. 206 - 208, 1981.

[32] D.L. Tao and C.R.P. Hartmann. "A novel concurrent error detection scheme for FFT networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, pp. 198-221, Feb. 1993.

[33] E.A. Trachtenberg and M.G. Karpovsky. "Filtering in a communication channel by Fourier transforms over finite groups," *Spectral Techniques and Fault Detection*, M. Karpovsky, ed., New York: Academic, 1985.

[34] J. Van Sas, F. Catthoor, S. Vernalde, M. Wouters, and H. De Man. "IC realization of a cellular automata based self-test strategy for programmable data paths." *Proc. 2nd European Test Conf.*, Munich, Ger., pp.35-44, 1991.

Mark G. Karpovsky received the MS degree in 1963 and the PhD degree in 1967 in computer engineering.

He has published more than 100 papers and several books in the areas of signal processing, logic design, testing, fault-tolerant computing, and error-correcting codes. He is the author of the book *Finite Orthogonal Series in the Design of Digital Devices*, Academic, New York, 1976 and the editor of the book *Spectral Techniques and Fault Detection*, Academic, New York, 1985.

Since 1983, he has been teaching at Boston University, Boston, Mass. and doing consulting work for IBM, Digital Corporation, Honeywell, and AT&T. He is currently a professor of computer engineering and director of the research laboratory on design and testing of computer and communication systems at Boston University.

Dr. Karpovsky has been a senior member of the IEEE since 1984 and a Fellow of the IEEE since 1993.

Tatyana D. Roziner received the MS degree in mathematics from Moscow University, Moscow, USSR in 1959. She received the PhD degree in applied mathematics and physics from the Acoustics Institute of the USSR Academy of Sciences in 1975.

From 1960 to to 1976, she worked as a research scientist at the Acoustics Institute in Moscow. She joined the National Institute for Industrial Automation in 1976, and in 1977-1978 she worked a the Central Institute for Integrated Automation. Between 1979 and 1985, she worked as senior research scientist for Israel Aircraft Industries. She joined the Department of Electrical, Computer, and Systems Engineering of Boston University in 1986.

Her research interests include data processing in multiprocessor environments, parallel architectures, and fast algorithms for parallel processing, digital design and VLSI testing.

Claudio Moraga received the BSc degree in electronics from Catholic University of Valparaíso, Chile in 1961, the MSc degree in electrical engineering from MIT in 1962, and DSc in electrical engineering from the Technical University Federico Santa María in Valparaíso in 1972.

From 1974 to 1976 he was an Alexander von Humboldt Research Fellow at the University of Dortmund, Germany, where he is presently an associate professor with the Department of Computer Science and Engineering.

His research interests include multiple-valued switching theory, CAD for microsystems, parallel architectures and computational intelligence.