

INDEPENDENCE FAULT COLLAPSING

Alok S. Doshi¹ and Vishwani D. Agrawal¹

Abstract -- This paper introduces independence fault collapsing. Faults are grouped into independent fault subsets such that each subset has some faults that cannot be covered by the tests derived for any other subset. Using these fault subsets, optimally compact tests can be found. For an equivalence or dominance collapsed fault set an independence graph is generated using structural and functional independences. Each fault is represented as a node and an undirected edge between two nodes indicates independence of the corresponding faults; two independent faults cannot be detected by the same vector. A "similarity-based" collapsing procedure reduces the graph to a fully-connected graph, whose nodes specify concurrently-testable (possibly testable by a common vector) fault targets for the ATPG. For the four-bit ALU (74181) circuit, our procedure produced 12 independent fault subsets. Each fault set produced one vector thus giving the smallest possible test set.

1. Introduction

The present automatic test pattern generation (ATPG) methodology is based on test generation for single fault targets, followed by fault simulation for fault dropping. Results of Table 1 [15] give the numbers of 100% coverage vectors from a typical ATPG program using various collapsed fault sets and show that, in general, we get more than the necessary number of tests. Most dynamic compaction procedures rely on the single fault ATPG [9, 12]. Although we cite only two papers, much work has been published in this area. For larger circuits, however, in spite of significant compaction, the minimum test set size is either not possible or too costly. We have not seen an ATPG or a vector compaction program that will produce 12 vectors for the four-bit ALU circuit [6, 10]. With this motivation, we intend to develop a new test generation methodology based on independence fault collapsing and concurrent-test generation [8]. The first of these concepts and its application are discussed in this paper.

2. Reexamination of Fault Collapsing

Four possible test conditions can exist between two faults. These are shown in Figure 1, where $T(F_i)$ denotes the set of all test vectors for fault F_i . For ATPG, faults are frequently collapsed via equivalence or dominance [7]. In equivalence collapsing, the faults are partitioned into disjoint equivalent sets and then one fault from each set is targeted by the ATPG. Detection of the targeted faults thus implies detection of all faults. In dominance collapsing, the target set is further reduced. When two faults, F_1 and F_2 , in an equivalence collapsed set

¹ Auburn University, Department of Electrical and Computer Engineering, Auburn, AL 36849, USA. Email: doshias@auburn.edu and vagrawal@eng.auburn.edu.

satisfy the relation $T(F2) \supset T(F1)$, meaning $T(F2)$ *dominates* $T(F1)$, fault F2 is dropped from the target fault set.

Circuit name	Structural fault collapsing				Functional fault collapsing [15]				Minimum vectors needed
	Equivalence		Dominance		Diagnostic dom.		Detection dom.		
	Faults	Vect.	Faults	Vect.	Faults	Vect.	Faults	Vect.	
Full adder	38	6	30	6	12	7	6	6	5
2-bit adder	74	9	58	8	24	6	12	9	5
8-bit adder	290	33	226	28	96	32	48	28	5
4-bit ALU (74181)	293	44	240	44	155	39	92	38	12

Table 1. Numbers of collapsed faults and vectors from an ATPG program.

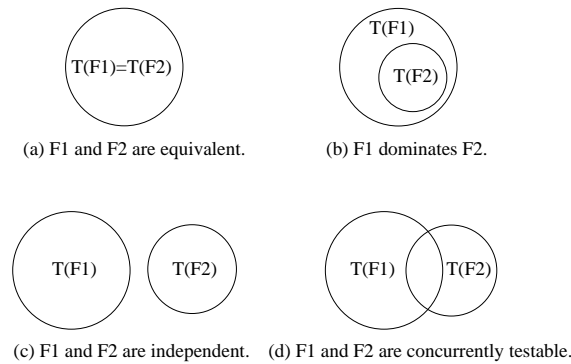


Figure 1. Test relations of faults F1 and F2 with tests T(F1) and T(F2).

An equivalence collapsed fault set always, and a dominance collapsed set mostly, generates tests covering all or most faults. However, as Table 1 shows, the number of test vectors is significantly larger than the essential minimum number required. The reason for this is explained by Figures 1 (c) and (d). Two faults, F1 and F2, in the collapsed set can be either *independent* [5, 6], i.e., they have no common test, or *concurrently-testable*, i.e., they have some common tests. In the absence of any knowledge of these behaviors, we target both faults. If they are independent then we get two tests, which are essential. If they are concurrently testable then we may get one vector (if we were lucky) or two vectors, although only one would have been sufficient. Thus, independence and concurrently-testable properties of faults can improve the efficiency of tests.

3. Independence Fault Collapsing

Two faults are called independent if and only if they cannot be detected by the same test vector [5, 6]. Two faults that are not independent must have at least one common test. Such fault pairs can be classified into three categories shown as (a), (b) and (d) in Figure 1. Categories (a) and (b) conform to equivalence and dominance as have been defined in the literature [7]. We define (d) below:

Definition: Two faults that neither have a dominance relationship nor are independent are defined as *concurrently-testable faults*.

A pair of concurrently-testable faults has two types of tests:

1. Each fault has an *exclusive* test that does not detect the other fault [1].
2. A common test detects both faults. We define this as a *concurrent* test.

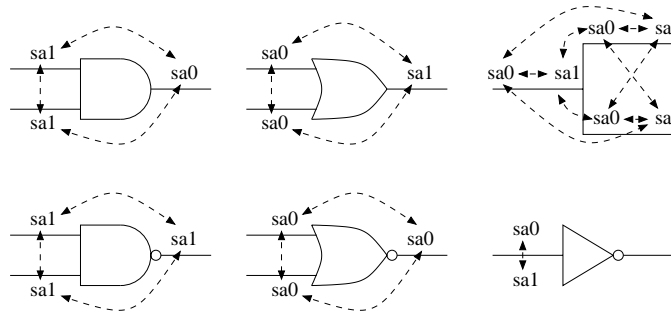


Figure 2. Structural independences of faults of Boolean gates and fanout.

Structural independences of faults of Boolean gates can be easily found and are shown in Figure 2. Here we have shown the faults after equivalence and dominance fault collapsing. This is because the faults with equivalence or dominance cannot be independent. Using the known results, many other independences can be determined:

1. Implication of equivalence: If two faults are equivalent then all faults that are independent of one fault are also independent of the other fault.
2. Implication of dominance: If one fault dominates a second fault then all faults that are independent of the first fault are also independent of the second fault.

In a large circuit, not all independences can be derived by structural analysis. The most general independence relations are functional and we give a procedure to find them. Consider a combinational circuit with output function C_0 and two single stuck-at faults, F_i and F_j . We denote the faulty functions as C_i and C_j , respectively. For F_i and F_j to be independent, the following equation must be satisfied for all inputs:

$$(C_0 \oplus C_i).(C_0 \oplus C_j) = 0 \quad (1)$$

This can be written as,

$$(C_0 \oplus C_i)C_0 \oplus (C_0 \oplus C_i)C_j = 0 \quad (2)$$

Equation 2 shows that if we construct a circuit $(C_0 \oplus C_i)C_0$, then a faulty circuit $(C_0 \oplus C_i)C_j$ will be indistinguishable when F_i and F_j are independent, i.e., they satisfy Equation 1. Figure 3 shows an independence identification procedure using an ATPG that checks for redundant faults. Here, three copies of the circuit under test (CUT) are made. In the third copy a fault F_i is permanently inserted. All three copies have the same primary inputs and their outputs are connected as shown in Figure 3 to construct one or more primary outputs for the composite circuit. An ATPG is used to detect faults in the top CUT. All faults that are found to be redundant are independent of F_i . If a fault F_j is found to be testable, the test is a concurrent test for faults F_i and F_j and can be saved for later use. It is assumed that both faults F_i and F_j are testable in CUT.

By successively inserting each fault in the lower copy of CUT in Figure 3 all pair-wise fault independences can be determined. We might point out that this procedure can be expensive and may be useful for small circuits only, which can be handled by an ATPG. For larger circuits one has to rely on the structural independences. Other procedures for independence identification use Boolean satisfiability or binary decision diagram analyses [16].

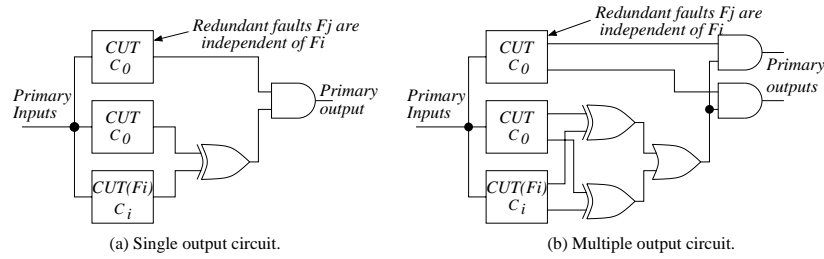


Figure 3. An ATPG-based method to find all faults independent of F_i .

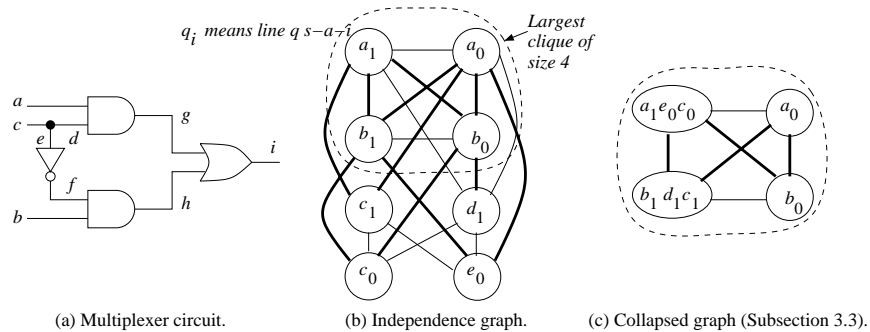


Figure 4. A multiplexer circuit and its independence graphs.

3.1. Independence Graph

An *independence graph* shows the independence relations between the faults of a circuit [16]. Each fault is represented by a node and the independence of two faults is represented by an undirected edge between the corresponding nodes. If all pairwise independences are known, then the absence of an edge between two nodes means that the two faults are testable by a common test; they can be equivalent, dominant or concurrently-testable. If the graph contains a dominance collapsed fault set, then the absence of an edge between two nodes means that the two faults are concurrently-testable.

For the multiplexer circuit of Figure 4(a), we have a set of eight faults, $\{a_0, a_1, b_0, b_1, c_0, c_1, d_1, e_0\}$ obtained after structural dominance collapsing by the Fastest program [11]. In the subscripted fault notation fault a_0 means “line a stuck-at-0”. We construct an independence graph as shown in Figure 4(b) where each fault is represented as a node and an undirected edge between two nodes indicates the independence of the corresponding faults. The edges shown by thin lines are structural and are found from Figure 2. The bold-line edges in the independence graph represent functional independences and are found through the ATPG-based procedure of Figure 3. Since this graph is small, we can easily identify the largest clique of size four. Such identification will be impossible for large circuits due to the high complexity of the largest clique problem [4]. The heuristic algorithm of Subsection 3.3 is found to work well.

In general, for large circuits one must rely only on structural independences and, therefore, the independence graph will be only partially complete. This would affect the minimality of the tests. In the following discussion, however, we will assume that all edges of the independence graph are known. An alternative representation of the independence graph is its connectivity matrix, which we will call *independence matrix*. The independence matrix for the eight-

node graph of Figure 4(b) is shown in Table 2. Here an edge between the i^{th} and j^{th} faults is indicated by 1s at the intersections of the i^{th} row and j^{th} column. The independence matrix has a diagonal symmetry because independence is a bidirectional property.

3.2. Graph Collapsing and Test Set Size

Theorem 1 [5]: *A lower bound on the number of tests required to cover all faults of an irredundant combinational circuit is given by the size of the largest clique of the independence graph, in which all edges are known.* ■

A *Clique* is defined as a fully-connected subgraph, i.e., a subgraph in which every node is connected to every other node. Thus, the largest clique in the independence graph of Figure 4(b), as shown by a dashed line enclosure, has a size 4. The above theorem follows from the fact that a test for a fault in the clique will not detect any other fault in that clique. Finding the largest clique in a graph (or even the *chromatic number*, i.e., the size of the largest clique) is an NP-complete problem [4]. Heuristically, two nodes that are not connected by an independence edge can form a single node whose label combines the fault labels of both nodes. Then, all nodes that had edges connecting to the two nodes will have edges to the combined node. This collapsing procedure ends when the graph becomes fully-connected. However, depending on the order in which the nodes are collapsed the size of the collapsed graph can vary. We have found that for larger circuits it produces non-optimum result. For improved collapsing, we propose a new heuristic method in the next subsection.

Once the independence graph is collapsed into a fully-connected graph (a single clique), the faults in each node label may require one or more tests. However, the concurrent tests generated for one node cannot completely detect all faults in any other node. Thus, the number of nodes in the clique is a lower bound on the number of tests. Suppose i^{th} node contains k_i faults, then any pair of those faults can be detected by a concurrent test. Therefore, we have

$$N_c \leq \text{Number of tests} \leq \sum_{i=1}^N \left\lceil \frac{k_i}{2} \right\rceil \quad (3)$$

where N_c is the number of nodes in the single-clique collapsed graph. How exactly those tests can be generated is discussed elsewhere [8].

3.3. A Graph Collapsing Algorithm

For a fault, we define two metrics, computed from the independence matrix:

Degree of independence (DI): This is the number of edges attached to the fault node and is computed for the i^{th} fault by adding all the elements of either the i^{th} row or the i^{th} column of the independence matrix:

$$DI (i^{\text{th}} \text{ fault}) = \sum_{j=1}^N x_{ij} = \sum_{j=1}^N x_{ji} \quad (4)$$

Where x_{ij} is the element belonging to the i^{th} row and j^{th} column of the $N \times N$ independence matrix. Thus, for the fourth fault, b_0 , of the multiplexer circuit, the matrix of Table 2 gives:

$$DI (b_0) = \sum_{i=1}^8 x_{4i} = 6 \quad (5)$$

Similarity metric (SIM): This is a measure defined for a pair of faults that determines how similar they are in their independence and concurrent-testability with respect to the entire fault set of the circuit:

$$SIM (fault - i, fault - j) = N_{x_{ij}} + (1 - x_{ij}) \sum_{k=1}^N |x_{ik} - x_{jk}| \quad (6)$$

The similarity metric ranges between 0 and N. When fault-i and fault-j are independent, $x_{ij} = 1$, and the metric assumes the largest value N. When the faults are not independent, the metric is simply the Hamming distance between the corresponding row or column vectors of the independence matrix. Although a 0 value may not indicate equivalence of any pair of faults, the similarity metric of two equivalent faults will be exactly 0. The pair-wise similarity metrics for the eight faults of the multiplexer circuit (Figure 4(a)) are shown in Table 2.

	a ₁	a ₀	b ₁	b ₀	c ₁	c ₀	d ₁	e ₀		a ₁	a ₀	b ₁	b ₀	c ₁	c ₀	d ₁	e ₀
a ₁	0	1	1	1	1	0	1	0	a ₁	0	8	8	8	8	1	8	0
a ₀	1	0	1	1	1	0	1	1	a ₀	8	0	8	8	8	2	8	8
b ₁	1	1	0	1	0	1	0	1	b ₁	8	8	0	8	1	8	0	8
b ₀	1	1	1	0	0	1	1	1	b ₀	8	8	8	0	2	8	8	8
c ₁	1	1	0	0	0	1	0	1	c ₁	8	8	1	2	0	8	1	8
c ₀	0	0	1	1	1	0	1	0	c ₀	1	2	8	8	8	0	8	1
d ₁	1	1	0	1	0	1	0	1	d ₁	8	8	0	8	1	8	0	8
e ₀	0	1	1	1	1	0	1	0	e ₀	0	8	8	8	8	1	8	0

Table 2. Independence matrix (left) and similarity matrix (right) for multiplexer of Figure 4(a).

The similarity metric and similarity index (defined below) are used to determine how likely a fault is for detection by a vector that also detects another fault or a group of faults. Notice that the faults that we are considering are neither equivalent nor have dominance relations because of the prior fault collapsing. These measures differ from the “level of similarity” defined in the literature [13], which determines how close a fault is to being equivalent or dominant with respect to another fault. Independence collapsing algorithm is:

- Compute the degree of independence for all faults and the similarity metric for each pair of faults.
- Starting with an empty graph, place faults in an order of decreasing degree of independence. Create the first node consisting of the fault with the highest degree of independence.
- Until all faults placed, place a fault F with the same or the next highest degree of independence:

- Compute a *similarity index* for F for each existing node i as:

$$Max_{k=1}^K SIM (F, k^{th} \text{ fault of node } i)$$

where K is the number of faults in node i .

- If the similarity index for all nodes is N (maximum value), i.e., all nodes contain at least one fault that is independent of F , then create a new node for F . Otherwise, place F in the node for which it has the smallest similarity index. ■

This algorithm groups faults into nodes such that the similarity metrics among faults within each group are minimized. This increases the possibility of finding a single concurrent test for the group. For the multiplexer circuit, we order the faults in the decreasing order of their degree of independence (shown in parenthesis) as: $a_0(6)$, $b_0(6)$, $a_1(5)$, $b_1(5)$, $d_1(5)$, $e_0(5)$, $c_1(4)$, and $c_0(4)$. The collapsed graph is shown in Figure 4(c). We start with a_0 and create the first

node for it. Since $SIM(b_0, a_0) = 8$ (indicating independence), we create a new node for b_0 . Similarly, two new nodes are created for a_1 and b_1 because they are independent of each other as well as independent of both a_0 and b_0 . Next, d_1 is placed in the node with b_1 because it has the lowest similarity index for that node. Proceeding in similar ways, all other faults are placed as shown in Figure 4(c). The edges in the collapsed graph indicate that a minimal set of tests for faults in any node cannot completely cover the faults in any other node.

The independence graph and its collapsed clique of size 4 ($N_c = 4$) are shown in Figure 4(c). Formula 3 gives the lower and upper bounds on the number of test vectors as 4 and 6, respectively. In Section 4, we will see that concurrent test generation provides four vectors for this circuit. This circuit has a reconvergent fanout, which can cause additional (functional) fault dominances not found by structural fault collapsing. If we use functional dominance collapsing [2, 3, 14] we will find that faults b_1 and c_1 can be eliminated because they functionally dominate d_1 . Similarly, a_1 and c_0 can be eliminated because they dominate e_0 . That will leave just one fault in each node of the collapsed graph of Figure 4(c).

4. Concurrent Test Generation

Using a concurrent test generation algorithm [8], we derived tests for faults in each node of Figure 4(c). Four tests were obtained, $\{a, c, b\} = 011, 100, 110$ and 001 , for fault sets, $\{b_1, d_1, c_1\}$, $\{a_1, e_0, c_0\}$, a_0 and b_0 , respectively.

The circuit diagram of the four-bit ALU (74181) used in this work is given in [8]. The exclusive-OR gates were implemented by four-NAND gate subcircuits. We obtained a collapsed set of 92 faults through functional dominance collapsing [15]. From this set, eight redundant faults were removed leaving 84 faults to be tested. An independence graph was obtained using the structural and functional independence identification methods of Section 3. The algorithm of Subsection 3.3 collapsed all faults into 12 nodes shown in Table 3. The maximum similarity indexes of faults within groups range between 21 and 49. Further shuffling of faults may reduce this but we did not attempt that.

Node no.	Max. sim. index in node	Faults in node				Cumulative fault coverage	Test vectors (input bit order as in circuit diagram of [8])
		Total	Targeted	Detected from			
				This node	Other nodes		
1	21	5	5	5	6	11	01001111010001
2	29	3	3	3	2	16	01001111110101
3	41	8	7	7	3	26	01011101000001
4	37	3	3	3	3	32	101x0101010000
5	43	5	3	3	4	39	10100101011000
6	49	6	6	6	2	47	11111000001001
7	49	7	4	4	3	54	11100000100000
8	38	14	11	11	1	66	11100110101011
9	47	8	6	5	1	72	10010100110101
10	41	8	4	3	2	77	1x101011101100
11	43	8	3	3	1	81	01010000101100
12	36	9	2	2	1	84	1x0111110001100

Table 3. Concurrent test generation for 4-bit ALU (74181) circuit. The input bit order for vectors in the last column is [8]: {s3, s2, s1, s0, b3, a3, b2, a2, b1, a1, b0, a0, m, c_n}.

For concurrent test generation, nodes were targeted in the order they are listed in Table 3. For example, the first node has 5 faults and all were targeted simultaneously [8]. The test vector, shown in the last column, detected all 5 targeted faults. Fault simulation of that vector showed that it detected 6 faults from other nodes, giving a cumulative coverage of 11. Although there were very few don't care bits in these vectors they were enumeratively filled during fault simulation and the combination of values that covered most extra faults was retained. The don't care bits shown in Table 3 are those that did not affect the fault coverage. For subsequent nodes, already detected faults were not targeted. Thus, node 3, which has 8 faults, only provided 7 target faults. By the time we reached the twelfth node, 7 of its faults had been detected and one fault from a previous node was left over. All three were detected by the twelfth vector. The set of 12 vectors in Table 3 is the smallest possible for this circuit [6, 10].

6. Conclusion

This paper is an attempt to present new ideas that have not been discussed before. Our search of the literature has found no references to the independence fault collapsing and concurrent test, introduced in this paper. The algorithms presented here give minimal test sets. However, the problem of completely determining all edges of the independence graph is complex and there is need to extend the procedure for incompletely-specified independence graph [8].

References

- [1] Agrawal V. D., Baik D. H., Kim Y.C. and Saluja K. K., 2003. Exclusive Test and Its Applications in Fault Diagnosis. *In: Proc. 16th Int. Conf. VLSI Design, Jan. 2003.* 143-148.
- [2] Agrawal V. D., Prasad A. V. S. S. and Atre M. V., 2003. It Is Sufficient to Test 25% of Faults. *In: Proc. 7th IEEE VLSI Design and Test Workshops (VDAT), Aug. 2003.* 368-374.
- [3] Agrawal V. D., Prasad A. V. S. S. and Atre M. V., 2003. Fault Collapsing via Functional Dominance. *In: Proc. Int. Test Conf., Sep. 2003.* 274-280.
- [4] Aho A. V., Hopcroft J. E. and Ullman J. D., 1987. *Data Structures and Algorithms.* Reading, Massachusetts: Addison-Wesley.
- [5] Akers S. B., Joseph C. and Krishnamurthy B., 1987. On the Role of Independent Fault Sets in the Generation of Minimal Test Sets. *In: Proc. Int. Test Conf., 1987.* 1100-1107.
- [6] Akers S. B. and Krishnamurthy B., 1989. Test Counting: A Tool for VLSI Testing. *IEEE Design & Test of Computers*, 6 (5), 58-77.
- [7] Bushnell M. L. and Agrawal V. D., 2005. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Boston: Springer.
- [8] Doshi A. S., 2005. *Independence Fault Collapsing and Concurrent Test Generation.* Thesis (MS). Auburn University, Department of ECE, Auburn, Alabama, USA. *In preparation.*
- [9] Hamzaoglu I. and Patel J. H., 2000. Test Set Compaction Algorithms for Combinational Circuits. *IEEE Trans. on CAD*, 19 (8), 957-963.
- [10] Hayes J. P., 2005. *74181 4-bit ALU and Function Generator and complete gate-level tests.* Available from: www.eecs.umich.edu/~jhayes/iscas/74181.html.
- [11] Kelsey T. P., Saluja K. K. and Lee S. Y., 1993. An Efficient Algorithm for Sequential Circuit Test Generation. *IEEE Trans. on Computers*, 42 (11), 1361-1371.
- [12] Pomeranz I., Reddy L. N. and Reddy S. M., 1993. COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits. *IEEE Trans. CAD*, 12 (7), 1040-1049.
- [13] Pomeranz I. and Reddy S. M., 2004. Level of Similarity: A Metric for Fault Collapsing. *In: Proc. Design, Automation and Test in Europe (DATE) Conf., volume 1, Mar. 2004.* 56-61.
- [14] Prasad A. V. S. S., Agrawal V. D. and Atre M. V., 2002. A New Algorithm for Global Fault Collapsing into Equivalence and Dominance Sets. *In: Proc. Int. Test Conf., Oct. 2002.* 391-397.
- [15] Sandireddy R. K. K. R. and Agrawal V. D., 2005. Diagnostic and Detection Fault Collapsing for Multiple Output Circuits. *In: Proc. Design, Automation and Test in Europe (DATE) Conf., Mar. 2005.* 1014-1019.
- [16] Wang J. C. and Stabler E. P., 1995. Collective Test Generation and Test Set Compaction. *In: Proc. IEEE Int. Symp. Circ. and Syst. (ISCAS'95), volume 3, Apr.-May 1995.* 2008-2011.