Theorems on Redundancy Identification

Vishal J. Mehta,[*] Vishwani D. Agrawal and Michael L. Bushnell
Rutgers University, ECE Dept.
Piscataway, NJ 08854
mehta@caip.rutgers.edu, vishwani02@yahoo.com, bushnell@caip.rutgers.edu

Abstract – *There is a class of implication-based methods that identify logic redundancy from circuit topology and without any primary input assignment. These methods are less complex than automatic test pattern generation (ATPG) but identify only a subset of all redundancies. This paper provides new results to enlarge this subset. Contributions are a fixed-value theorem and two theorems on fanout stem unobservability. Our framework is an implication graph of signal controllabilities and observabilities represented as Boolean variables. Besides the conventional implication edges this graph also contains partial implications implemented by AND nodes. An analysis of the transitive closure (TC) of this graph provides many redundancies. Weaknesses of this procedure are in dealing with the effects of fixed-valued variables on TC and the lack of observability relations across fanouts. The fixed-value theorem adds unconditional edges from all variables to the fixed variable and then recomputes TC recursively until no new fixed nodes are found. The stem unobservability theorems determine the observability status of a fanout stem from its dominator set, which either has fixed values or is unobservable. Results are considerably improved from the previously reported implication-based identifiers. In the c5315 circuit we identify 58 out of 59 redundant faults. All 34 redundant faults of c6288 are identified. Besides, our procedure can classify faults according to the causes of their redundancy, namely, unexcitable, unobservable, or undrivable. For the future research, we provide examples of cases where the present method still fails.*

1. Introduction

Redundant logic in a digital circuit is often identified as untestable or redundant single stuck-at faults. Redundant faults in a combinational circuit are faults that no input patterns can detect [2]. Removal of such faults simplifies the circuit without changing its input-output function. Redundancy removal makes sure that the circuit does not have unnecessary logic gates, which increase the circuit size, and in turn increase the chip area, power consumption and propagation delay [17]. The other reason as to why redundancies should be identified is the reliability of the circuit, because in the presence of redundant faults some testable faults may be masked. There are three different approaches to redundancy identification: *automatic test pattern generation* (ATPG) [4, 5, 8, 12, 15, 16, 22, 23, 25, 26], testability analysis where depending on the type of method we use, i.e., approximate or exact, we can identify all or a subset of "probably" redundant faults [9, 19, 21, 24], and implication-based methods [1, 7, 11, 13, 17, 18].

ATPG methods can find all redundant faults but they have very high complexity. Their algorithm complexity is exponential in the circuit size. Testability analysis, often restricted to a linear complexity, can identify "hard to test" faults some of which may be redundant. Implication-based methods do not guarantee to find all redundant faults and their complexity is polynomial. Both testability analysis and implication-based algorithms do not target specific faults and are sometimes referred to as "fault-independent methods." These do not assign any values to the primary input variables and in that sense they differ from the ATPG algorithms.

Chakradhar *et al.* [3] define a Boolean false function to construct an implication graph from a given gate level netlist. Both true and false status of the signal line is represented by nodes in the implication graph. Pair-wise relationships of signal lines provide implication edges in the graph. Signal dependencies derived from the transitive closure are used to reduce ternary relations to binary relations that in turn dynamically update the transitive closure. When the ATPG problem is posed as Boolean satisfiability, the implication garph provides useful help in solving it. Recent work has produced several ATPG programs

[*] Student

that use Boolean satisfiability [4, 12, 16, 25, 26].

The implication graph and transitive closure have also been used in fault-independent approaches to redundancy identification [1]. Here observability of signals are represented as additional Boolean variables in the implication graph. Gaur *et al.* [6, 7] improved such a redundancy identification technique, using a subset of all partial implications derived from the higher order terms represented using anding nodes. Mehta *et al.* [17] implemented all the unimplemented direct and partial implications and this improved the results considerably. The nodes in the graph represent the true and false status of the signal lines and true and false status of the observability of those signal lines. The directed edges represent the Boolean implication relationships between these nodes. The motivation for the present work comes from the weaknesses of the existing technique. An analysis of the redundant fault found by an ATPG program that could not be identified by the implication method showed:

- The circuit topology forces some signals to fixed values. The implication graph procedures could not analyze the effect when a set of fixed signals influence the state of another signal.

- There was no generally known procedure to determine the observability status of a fanout stem even when the fanout branches were unobservable or had fixed values.

Other authors who have contributed to implication-based approaches are Iyer and Abramovici [13], Peng *et al.* [20], and Zhao *et al.* [27].

In Section 2, the analysis of fixed-value nodes is discussed. Stem unobservability theorems are discussed in Section 3. Limitations of our technique are given in Section 4. Results are discussed in Section 5. Section 6 presents the conclusion of this paper.

2. Fixed-Valued Nodes

A Boolean variable x is represented by two nodes x and $\overline{x}$ in the implication graph. Each node can be in a true of false state. When $x = 1$, node x is true and $\overline{x}$ is false, and vice-versa. The variable x assumes a fixed value 0 if there exists an edge $x \rightarrow \overline{x}$ in the implication graph. Similarly, x assumes a fixed value 1 when the edge $\overline{x} \rightarrow x$ exists. When several nodes have fixed-values, we have found some limitations of the previously described transitive closure technique [6] in identifying certain redundancies. We consider:

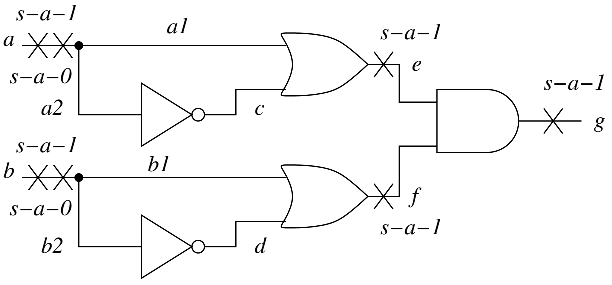

Figure 1: Example circuit with fixed-value nodes.

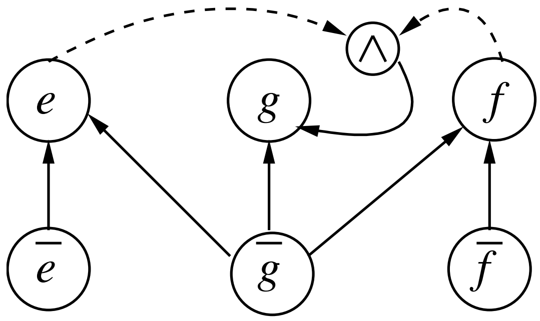

Figure 2: Implication graph showing how the primary output redundant fault was identified redundant using Theorem 2.1.

- A primary output AND or NOR gate with an unexcitable s-a-1 redundant fault on its output line (see g s-a-1 in Figures 1 and 3).

- A primary output OR or NAND gate with an unexcitable s-a-0 redundant fault on its output line.

The following theorem identifies the above mentioned redundancies.

Theorem 2.1 *(Fixed-value) If a Boolean variable in the implication graph is fixed to a true (false) value then there exist unconditional edges from all other nodes in the graph to the node representing the true (false) state of the fixed variable.*

Proof: Suppose there exists an edge from node a to $\overline{a}$ in the graph (where a and $\overline{a}$ represent true and complement values of signal line a in the circuit), then irrespective of the status of any other nodes in the graph, $\overline{a}$ will be unconditionally true. In other words, there exists an edge from all the nodes in the graph to the node $\overline{a}$. The status of signal line a will remain 0 even though node a is traversed in the graph traversal, because it will eventually lead to $\overline{a}$. ■

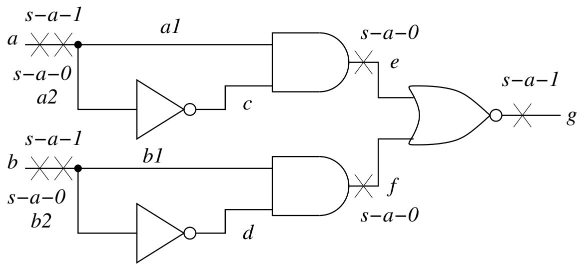

Figure 3: Example circuit showing the consistency of limitations of previous technique [6].

The example circuit shown in Figure 1 has 7 redundant faults. Previous work [6] could identify s-a-1 faults on lines e and f. They could not identify redundant faults: s-a-1 on line g, because of insufficient edges to identify the s-a-1 redundant fault on the primary output for the AND gate and s-a-0 and s-a-1 on lines a and b, due to a lack of stem unobservability relationships.

Without Theorem 2.1, we could not identify s-a-1 on line g as a redundant fault. Figure 2 shows a part of the implication graph. The edges other than the three outgoing edges from $\overline{g}$ exist in the transitive closure. The faults e s-a-1 and f s-a-1 are identified to be redundant due to edges from nodes e and f in the transitive closure graph to nodes $\overline{e}$ and $\overline{f}$, respectively. These faults are classified as unexcitable redundant faults. As these nodes are fixed to logic value 1, according to Theorem 2.1, we implemented unconditional edges from all other nodes in the graph to nodes e and f. These include the two edges from g to e and f, respectively. The transitive closure is obtained by a graph traversal from each node to determine its reachability. Traversal proceeds through an AND node only when paths arrive at all inputs. In this case, therefore, when we start at $\overline{g}$, g is reached and the edge $\overline{g} \rightarrow g$ is added. This sets g to a fixed value 1 and identifies the s-a-1 fault on line g as redundant. The other four faults, a and b both s-a-0 and s-a-1, are not yet identified due to the absence of any false stem observability edges. These faults are identified as redundant with the theorems introduced in next section. As another example, in Figure 1, replace: The AND gate with a NOR gate and both OR gates with AND gates. This is shown in Figure 3. Using similar arguments and Theorem 2.1, we can identify g s-a-1 as a redundant fault. This example is consistent with the limitations identified (of the previous method [6]) earlier in this section.

3. Stem Unobservability Theorems

We define the *dominator set* of a fanout stem in a circuit as a minimal set of signals through which all forward paths originating at the stem must pass. Trivially, a dominator set for any signal can be found as the set of reachable primary outputs (POs). A dominator set consisting of a single signal is called a *unique dominator*. Our definition is consistent with the graph theory [10] and has also been used in an ATPG algorithm [14].

Theorem 3.1 *(Stem unobservability) A fanout stem is unobservable, if each signal in its dominator set assumes a constant value and:*

- *the fanout stem does not hold a constant value, or*
- *the fanout stem holds a constant value and, in spite of any local change in the stem signal, the dominator set values do not change.*

A *local change* of a signal only affects the portion of the circuit between that signal and POs.

Proof: From the definition of the dominator set, if the stem under consideration is unobservable at the dominator set then it will be unobservable at primary outputs as well. So:

$$f_i(X, x_p) = c \tag{1}$$

where X is the set of primary inputs, x_p is the fanout stem under consideration, f_i is the dominator set function, that is a function of X and x_p and $i \in [1, k]$, where k is the cardinality of the dominator set and c is a constant, which can assume a logic 0 or 1 value. Expanding Equation 1 using Shannon's expansion theorem [2], we get:

$$x_p f_i(X, 1) + \overline{x_p} f_i(X, 0) = c \tag{2}$$

We consider two cases:

- $x_p \neq$ constant, then Equation 2 implies:

$$f_i(X, 1) = f_i(X, 0) = c \tag{3}$$

Taking the Boolean difference of Equation 1, with respect to x_p, we get:

$$\frac{\partial f_i(X, x_p)}{\partial x_p} = f_i(X, 1) \oplus f_i(X, 0) \tag{4}$$

From Equation 3:

$$\frac{\partial f_i(X, x_p)}{\partial x_p} = c \oplus c = 0 \tag{5}$$

Thus, x_p is unobservable.

- x_p = constant. Substituting $x_p = 1$ in Equation 2, we get:

$$f_i(X,1) = c \textit{ and } f_i(X,0) = \textit{unknown} \quad (6)$$

and substituting $x_p = 0$ in Equation 2, we get:

$$f_i(X,1) = \textit{unknown and } f_i(X,0) = c \quad (7)$$

The Boolean difference of Equation 1, with respect to x_p gives:

$$\frac{\partial f_i(X, x_p)}{\partial x_p} = f_i(X,1) \oplus f_i(X,0) \neq 0 \quad (8)$$

In the special case, when $f_i(X,1) = f_i(X,0) = c$, *i.e.*, $f_i(X, x_p)$ remains unchanged when x_p assumes two different values, Equation 8 evaluates to 0 and the stem x_p becomes unobservable. ■

Theorem 3.2 *(Stem unobservability) A fanout stem is unobservable, if each signal in its dominator set is unobservable and:*

- *the stem does not hold a constant value, or*
- *the stem holds a constant value and, in spite of any local change in the stem signal, the unobservable status of the dominator set remains unchanged.*

Proof: From the definition of the dominator set, it is sufficient to show that the stem under consideration is unobservable at the dominator set. Consider the primary output functions:

$$f_i(X, x_p, d_1, ..., d_n) \quad (9)$$

where, X is a set of primary inputs, x_p is the fanout stem under consideration, $d_1, ..., d_n$ are the dominator set signals for x_p, $i \in [1, k]$, where k is the cardinality of the PO set. Let the dominator function be $d_j(X, x_p)$, $j \in [1, n]$. Since dominator signal d_1 is unobservable:

$$\frac{\partial f_i}{\partial d_1} = f_i(X, x_p, 1, d_2, ..., d_n) \oplus f_i(X, x_p, 0, d_2, ..., d_n) = 0 \quad (10)$$

Similarly, $d_2, ..., d_n$ are unobservable. Thus, we get:

$$\frac{\partial f_i}{\partial d_n} = f_i(X, x_p) + f_i(X, \overline{x_p}) = 0 \quad (11)$$

Now the Boolean difference of the above equation with respect to x_p gives:

$$\frac{\partial f_i}{\partial x_p} = x_p f_i(X,1) + \overline{x_p} f_i(X,0) = 0 \quad (12)$$

We consider two cases:

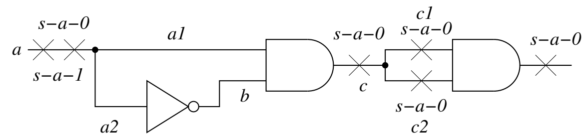

Figure 4: Example circuit illustrating the cases of stem observability and unobservability (all faults shown are redundant).

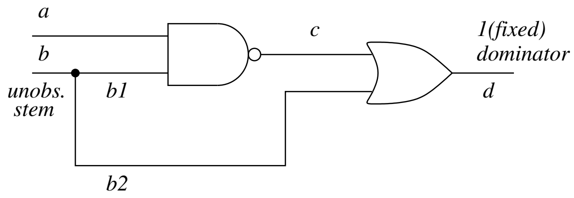

Figure 5: Example circuit justifying the implementation of Theorem 3.1.

- If $x_p \neq$ constant, then:

$$f_i(X,1) = f_i(X,0) = 0 \quad (13)$$

Thus, x_p is unobservable.

- If x_p = constant. For $x_p = 1$, from Equation 12 we get, $f_i(X,1) = 0$ and $f_i(X,0)$ = unknown and for $x_p = 0$, we get, $f_i(X,0) = 0$ and $f_i(X,1)$ = unknown. Taking the Boolean difference of Equation 9, with respect to x_p, is thus not equal to 0. Thus, x_p is observable. But in a special case when unknown becomes equal to 0, x_p becomes unobservable. ■

The fanout stem b, shown in Figure 6, is not fixed and the dominator d has a constant value of 0. Thus, Theorem 3.1 identifies fanout stem b as unobservable. The fanout stem d, shown in Figure 6, is fixed to 0 and the dominator set $\{e, f\}$ holds a constant value of {0,0}. Thus, Theorem 3.1, locally changes the value of d and checks the effect on the dominator set, which is found to be changed. Thus, Theorem 3.1 does not identify d as an unobservable stem. The fanout stem a, shown in the same figure, is not fixed and the dominator set signals $\{e, f\}$ are fixed to 0. Thus, Theorem 3.2 identifies a as an unobservable fanout stem. The fanout stem b, shown in Figure 5, is not fixed and the dominator d is fixed to 1. Thus, Theorem 3.1 identifies b as an unobservable fanout stem. The fanout stem c, shown in Figure 4, is fixed to 0 and the dominator set signals $\{c1, c2\}$ are unobservable. A change in the value of c changes the observability status of the dominator set. Thus, Theorem 3.2 does not identify c as an unobservable stem.

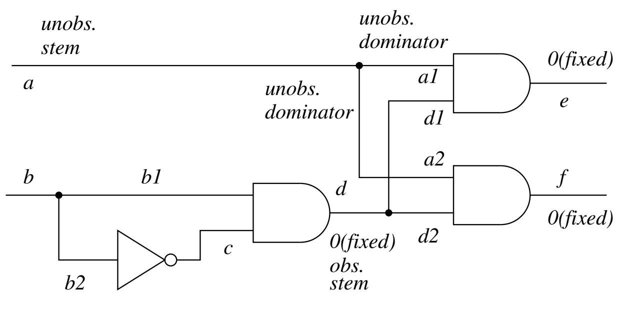

Figure 6: Example circuit justifying the implementation of Theorem 3.2.

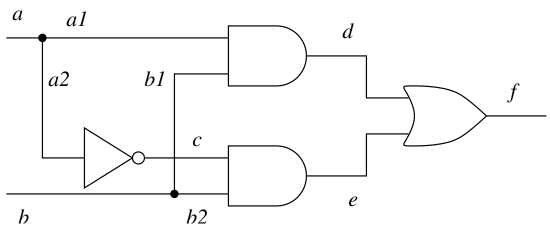

Figure 7: Example circuit showing the limitation of our method.

Theorem 3.2 is consistent with a lemma proposed by Iyer and Abramovici [13]. Their lemma is a special case of the Theorem 3.2 because the theorem does not require the uncontrollability indicator condition required by the lemma. After the observability status of a fanout stem is decided, an unobservability edge is implemented in the implication graph.

4. Limitations

The limitations we identified in Section 1, are only partially removed by the three theorems. If a node is fixed then we add unconditional edges to improve the reachability. However, if no node is fixed then we cannot traverse backwards from that node. That is, if we want to traverse from a 0 at the output of an AND gate backwards than we do not have implications to do so. Stem unobservability theorems cannot identify the observability status of the fanout stems a shown in Figure 7. The dominator is neither fixed nor unobservable. Thus, none of the theorems can find the unobservability status of the fanout stem a.

Figure 8 gives another circuit that illustrates a limitation. This circuit has six redundant faults as found by an ATPG program and marked in the figure. Two faults on c cannot be identified because no fixed-valued or unobservable dominator set is found. Faults a s-a-1, $c1$ s-a-1, and d s-a-1 are identified as

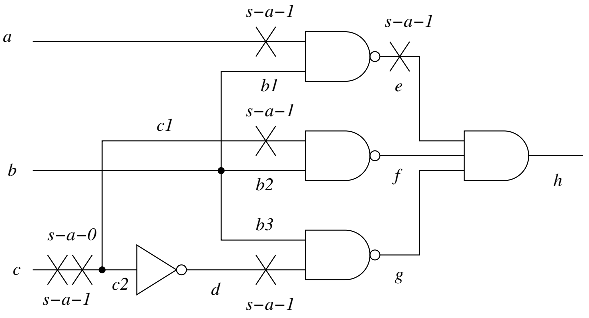

Figure 8: Another example of limitation.

redundant. The fault e s-a-1 is not identified because it is *undrivable*. This means that the fault can be excited and its site is observable, but the excitation and observation cannot be done simultaneously. We note that observation of e requires $f = g = 1$, which can only be achieved when $b = 0$. However, that sets $e = 1$. Thus, observation of the fault effect makes it impossible to excite it. In the implication graph this would be indicated be a directed edge from the true observability node (say O_e) to the node e. An extended implication method in which sets of signals are enumerated to check for essential assignments can identify such a redundancy at a higher computation cost [20]. The three redundant faults not identified in the circuit c432 are of this type.

5. Results

Table 1 shows the results obtained on ISCAS '85 and ISCAS '89 benchmark circuits. The first column shows the names of benchmark circuits for which results from various programs are compared with respect to the redundant faults identified and their respective CPU times. The next column shows the total number of collapsed single faults in the circuit. The next two columns show the redundant faults identified and CPU time in seconds for the ATPG tool TRAN [4] by Chakradhar *et al.*, which uses transitive closure for test generation. The next two columns lists the result of FIRE [13] by Iyer and Abramovici. The next two columns show the results of the transitive closure algorithm without partial implications by Agrawal *et al.* [1]. The next two columns show the result of the transitive closure algorithm with some of the partial implications by Gaur *et al.* [7]. The last two columns are the results of the new fault-independent algorithm, which uses transitive closure for redundancy identification. The sequential benchmark circuits were converted into combinational circuits by removing the flip-flops from the

Table 1: Combinationally redundant faults in ISCAS '85 and ISCAS '89 benchmark circuits.

Circuit	Total faults	Redundant faults identified and run time on Sun workstations (a: Sparc 5; b: Sparc 2)									
		TRAN [4]		*FIRE* [13]		*TC* [1]		$TC_{w\_partial}$ [7]		*Our Algo.*	
		Redun. faults	CPU s (a)	Redun. faults	CPU s (b)	Redun. faults	CPU s (a)	Redun. faults	CPU s (a)	Redun. faults	CPU s (a)
c432	524	4	0.8	-	-	0	0.1	0	0.2	1	0.2
c499	758	8	1.8	-	-	0	0.2	0	0.2	0	1.3
c880	942	0	4.0	-	-	0	0.3	0	0.4	0	0.4
c1355	1574	8	11.0	-	-	0	0.7	0	0.9	0	1.9
c1908	1879	7	13.0	6	1.8	2	0.8	2	0.9	2	3.2
c2670	2747	115	95.2	29	1.5	23	1.3	25	1.5	82	4.0
c3540	3428	131	24.9	93	11.9	54	5.9	74	6.2	111	16.2
c5315	5350	59	32.3	20	2.8	20	3.4	32	3.4	58	3.9
c6288	7744	34	38.0	33	1.3	31	1.0	31	1.8	34	7.2
c7552	7550	131	308.0	30	4.7	32	5.2	34	5.8	55	11.5
s349	350	2	0.3	2	0.2	1	0.1	2	0.2	2	0.2
s444	474	14	0.4	11	0.2	2	0.1	8	0.2	10	0.3
s713	581	38	3.1	32	0.1	29	0.3	35	0.3	38	0.6
s1238	1355	69	17.4	6	1.9	4	0.5	6	0.6	20	2.6
s1423	1515	14	8.5	5	0.3	4	0.6	8	0.7	12	1.0
s1494	1506	12	3.7	1	1.1	1	0.7	1	0.8	2	8.8
s5378	4603	40	73.0	34	3.7	20	2.5	22	3.0	27	8.3
s9234	6927	452	803.7	165	20.6	16	9.8	135	11.2	233	106.0
s13207	9815	151	806.5	55	23.2	9	11.2	60	13.6	77	158.8

circuit and treating flip-flop outputs as primary inputs and flip-flop inputs as primary outputs. There are no aborted faults in TRAN [4] for all circuits in Table 1. We use benchmark circuit c5315 to compare results for all these different programs with our work. The ATPG tool TRAN [4] identifies all the 59 redundant faults in the circuit without aborting any fault in CPU time of 32.3 sec. FIRE [13] identifies 20 redundant faults with a CPU time of 2.8 sec. Next, the transitive closure algorithm without partial implication [1] identifies 20 redundant faults with CPU time of 3.4 sec. Next, the transitive closure algorithm with some partial implications [7] identifies 32 redundant faults with CPU time of 3.4 sec. Finally, our algorithm with the fixed-value theorem and stem unobservability theorems implemented identifies 58 redundant faults in 3.9 sec. Thus, in the benchmark circuit c5315 our algorithm identifies more redundant faults then any of the fault independent algorithms. The redundant faults identified by our work are much greater than all the other implication-based methods described in the result table with comparable CPU time. The results are also comparable to the ATPG tool TRAN because for the benchmark circuits c3540, c5315, c6288, s349, s444, s713, s1423 and s9234 our work provides either all or close to all redundant faults. We do recognize that there are ATPG programs [26] that are faster than TRAN, but then our implementation is only experimental with potential performance improvements possible.

An analysis on the benchmark circuit c432 reveals that due to the inability to take decision (*i.e.,* traverse backwards from the false node of the output of the AND gate, shown in Section 2) we can not identify 3 out of 4 redundant faults (*i.e.,* we identify only 1 out of 4 redundant faults.)

6. Conclusion

The fixed-value theorem and stem unobservability theorems enhance the transitive closure technique to identify redundant faults. Its other applications, not discussed here, include the identification of the equivalent faults and equivalence checking of two given circuits. Future research may find new ways of taking decisions in the implication graph to traverse from the false status of the output signal of an AND gate. Basically, our technique has a lower complexity than the exponential complexity of an ATPG algorithm and identifies many, though not all, redundant faults.

References

[1] V. D. Agrawal, M. L. Bushnell, and Q. Lin, "Redundancy Identification Using Transitive Closure," in *Proc. of the 5th Asian Test Symp.*, November 1996, pp. 4–9.

[2] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Boston, MA: Kluwer Academic Publications, 2000.

[3] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Neural Net and Boolean Satisfiability Models of Logic Circuits," *IEEE Design and Test of Computers*, vol. 7, no. 5, pp. 54–57, October 1990.

[4] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 7, pp. 1015–1028, July 1993.

[5] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," in *Proc. of the International Fault-Tolerant Computing Symp.*, June 1983, pp. 98–105.

[6] V. Gaur, "A New Transitive Closure Algorithm to Identify Redundancies in Logic Circuits," Master's thesis, Rutgers University, January 2002.

[7] V. Gaur, V. D. Agrawal, and M. L. Bushnell, "A New Transitive Closure Algorithm with Applications to Redundancy Identification," in *Proc. of the 1st International Workshop on Electronic, Design and Test Applications (DELTA'02)*, January 2002, pp. 496–500.

[8] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Computers*, vol. C-30, no. 3, pp. 215–222, March 1981.

[9] L. H. Goldstein, "Controllbility/Observability Analysis of Digital Circuits," *IEEE Trans. on Circuits and Systems*, vol. CAS-26, no. 9, pp. 685–693, September 1979.

[10] F. Harary, *Graph Theory.* Reading, MA: Addison-Wesley, 1972.

[11] M. Harihara and P. R. Menon, "Identification of Undetectable Faults in Combinational Circuits," in *Proc. of the International Conf. on Computer Design*, October 1989, pp. 290–293.

[12] M. Henftling, H. Wittmann, and K. J. Antreich, "A Formal Non-Heuristic ATPG Approach," in *Proc. of the European Design Automation Conf.*, September 1995, pp. 248–253.

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

[14] T. Kirkland and M. R. Mercer, "A Topological Search Algorithm for ATPG," in *Proc. of the* 24^{th} *Design Automation Conf.*, June-July 1987, pp. 502–508.

[15] W. Kunz and D. K. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits," in *Proc. of the IEEE International Test Conf.*, September 1992, pp. 816–825.

[16] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 1, pp. 4–15, January 1992.

[17] V. J. Mehta, K. K. Dave, V. D. Agrawal, and M. L. Bushnell, "A Fault-Independent Transitive Closure Algorithm for Redundancy Identification," in *Proc. of the* 16^{th} *International Conf. VLSI Design*, January 2003, pp. 149–154.

[18] P. R. Menon and H. Ahuja, "Redundancy Removal and Simplification of Combinational Circuits," in *Proc. of the* 10^{th} *IEEE VLSI Test Symp.*, April 1992, pp. 268–273.

[19] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. on Computers*, vol. C-24, no. 6, pp. 668–670, June 1975.

[20] Q. Peng, M. Abramovici, and J. Savir, "MUST: Multiple-stem Analysis for Identifying Sequentially Untestable Faults," in *in Proc. of the International Test Conf.*, September 2000, pp. 839–846.

[21] I. M. Ratiu, A. Sangiovanni-Vincentelli, and D. O. Pederson, "VICTOR: A Fast VLSI Testability Analysis Program," in *Proc. of the IEEE International Test Conference*, November 1982, pp. 397–401.

[22] J. P. Roth, "Diagonosis of Automata Failures: A Calculus and a Method," *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, July 1966.

[23] M. H. Schulz, E. Trischler, and T. M. Serfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. on Computer-Aided Design*, vol. CAD-7, no. 1, pp. 126–137, January 1988.

[24] S. C. Seth and V. D. Agrawal, "A New Model for Computation of Probabilistic Testability in Combinational Circuits," *Integration, the VLSI Journal*, vol. 7, no. 1, pp. 49–75, April 1989.

[25] J. P. M. Silva and K. A. Sakallah, "Grasp - A new Search Algorithm for Satisfiability," in *Proc. of the International Conf. on Computer-Aided Design*, November 1996, pp. 220–227.

[26] P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Combinational Test Generation Using Satisfiability," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 9, pp. 1167–1176, September 1996.

[27] J. K. Zhao, E. M. Rudnick, and J. H. Patel, "Static Logic Implication with Application to Redundancy Identification," in *Proc. of the* 15^{th} *IEEE VLSI Test Symp.*, April 1997, pp. 288–293.