# Redundancy Identification Using Transitive Closure

*Vishwani D. Agrawal*
Bell Labs, Lucent Technologies
Murray Hill, NJ 07974
*va@research.bell-labs.com*

*Michael L. Bushnell*
CAIP Center, Rutgers University
Piscataway, NJ 08855
*bushnell@caip.rutgers.edu*

*Qing Lin**
Sun Microsystems
Chelmsford, MA 01824
*qing@east.sun.com*

## Abstract

*We analyze all signals of a combinational circuit simultaneously for redundancy. The state of a signal is represented by two binary variables. The first variable is the logic value of the signal. The second variable is the observability status of the signal with respect to all primary outputs. Boolean equations specify local relationships of these variables in a manner similar to the neural network or Boolean satisfiability method. All pairwise terms appearing in these Boolean equations are used to construct an implication graph, for which the transitive closure graph is obtained. Any signal assignments or relations found from the transitive closure are substituted into higher-order terms of the Boolean equations, some of which reduce to pairwise terms. Such cases are iteratively included in the transitive closure until no more reductions are possible. In the final transitive closure, all signals are examined for the following conditions of redundancy: (1) If a signal and its complement imply each other (contradiction) then both stuck-at faults on that signal are redundant; (2) If one value implies the other value (fixation) then one of the stuck-at faults on that signal is redundant; (3) If the true observability status of a signal implies its own false observability status, then both stuck-at faults of that signal are redundant; (4) If a certain value of a signal implies the false observability status, then the corresponding stuck-at fault is redundant. Despite the apparent similarities with the transitive closure based ATPG, the present method is quite different. Here transitive closure is computed just once, and not recomputed or updated separately for each fault as required in ATPG. We give ISCAS '85 benchmark results. For c6288, we could identify 31 out of 33 redundancies. The percentage of identified redundancies was not always that high, but the algorithm has polynomial complexity and we discuss its limitations.*

## 1 Introduction

Redundant faults in a combinational circuit are faults that no input patterns can detect. They increase chip area and cause difficulty in test generation. Methods of redundancy identification can be classified as either ATPG-based or fault-independent. An *ATPG-based method* uses an exhaustive test pattern generator to determine whether or not a target fault has a test. A complete test generation thus produces a list of all redundant faults. Such results have been reported by several authors [1, 2, 3, 4, 5]. *Fault-independent methods* analyze circuit topology and function without targeting a specific fault. The analysis may involve controllability and observability of signals [6]. To control the complexity of analysis, approximations are normally used. These methods can, therefore, find *some* redundancies very quickly [7, 8]. In this paper, we give a new fault-independent method for redundancy identification. The following paragraph gives a very brief review of previous methods.

Menon *et al.* proposed an approach based on the analysis of reconvergent fanout structures [9], [10]. Their method requires repeatedly computing the controlling value paths associated with every reconvergent gate of every circuit stem. Iyer and Abramovici give another technique to identify redundancy by propagating *uncontrollability* values in combinational circuits [11]. They also extended the method to sequential circuits [12]. Agrawal and Chakradhar [13] proposed a new methodology for finding untestable faults in sequential circuits. They applied an ATPG program to a combinational model derived from the sequential circuit. The combinational model, though functionally not equivalent, preserves the untestable behavior of some faults. Pomeranz and Reddy [14] have further elaborated on that technique. Also, for sequential circuits, Liang *et al.* [15, 16] use symbolic simulation to identify flip-flops that cannot be initialized. That information is used to identify untestable faults.

The present redundancy identification technique is based on transitive closure computation. The state of

*Formerly with Rutgers University, Piscataway, NJ 08855

1081 – 7735/96 $5.00 © 1996 IEEE
*Proceedings of ATS '96*

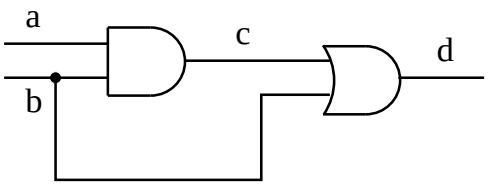



Figure 1: An example circuit

a signal in a combinational circuit is represented by two binary variables. The first is the binary value of the signal. The second gives the observability status of the signal at primary outputs. Boolean expressions are obtained for local relationships of state variables. Only the pairwise relations are represented in an implication graph for which we compute the transitive closure to find global dependencies of signals. Signal relations derived from transitive closure are then used to reduce higher-order relations of the Boolean expression to binary relations that in turn dynamically update the transitive closure. When no more reduction is possible, we examine all signals to identify several classes of redundancies. *In comparison to transitive closure-based ATPG, which must recompute or update transitive closure separately for each target fault, the present technique will compute transitive closure only once.* However, in general, only a subset of all redundant faults may be found.

## 2 Main Idea

We use the circuit of Figure 1 to illustrate the new idea. The signals, $a$, $b$, $c$ and $d$, are related through two Boolean gates. The basics of the minimum energy model used in this work are given in a recent book [17]. The gates are described by the disjunction of terms shown in Table 1. Thus, the expression $\bar{a}c + \bar{b}c + ab\bar{c}$ describes the AND gate. This expression becomes 0 only when signals $a, b$ and $c$ conform to the AND function. The conjunction of expressions for all gates gives the expression for the entire circuit.

In the present work, we define observability variables for signals. Thus, $O_a$ is a Boolean variable which is 1 when signal $a$ is observable at the output $d$. $O_{bc}$ is the observability variable for the fanout of $b$ that feeds into gate $c$. The relations of observability variables, which should also evaluate to 0, are shown in Table 1. Details of these expressions are given in the next section.

The pair-wise relations from Table 1 are represented in the implication graph of Figure 2. This graph contains two nodes for each variable. Thus, the true and false states of signal $a$ are represented by nodes $a$ and $\bar{a}$. Notice that a term $\bar{a}c$ is represented by two arcs in

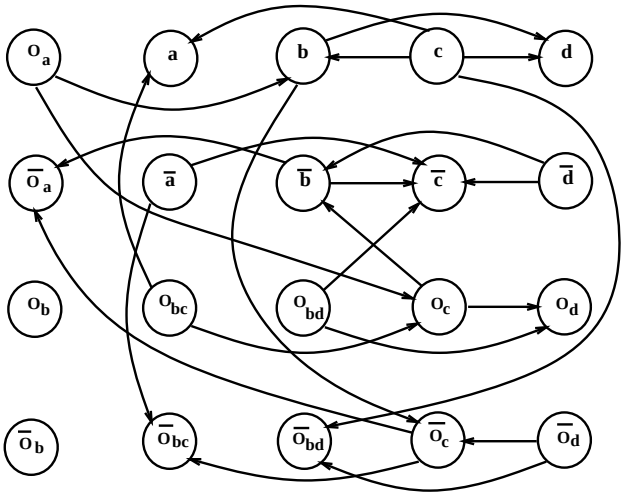



Figure 2: Implication graph of example in Figure 1

the graph. Given that $c$ is true, the term will be 0 only when $\bar{a}$ becomes false. Thus node $c$ implies $a$ as shown by an arc from $c$ to $a$. Similarly, $\bar{a}$ implies $\bar{c}$.

Next, we obtain the transitive closure for the implication graph in the same way as described by Chakradhar *et al.* [1]. In this process, any fixations or identifications are used to simplify the higher-order terms (see Table 1) for inclusion in the graph. Notice, that there are paths from $O_a$ to $\bar{O_a}$, e.g., $O_a$, $O_c$, $\bar{b}$, $\bar{O_a}$ and $O_a$, $b$, $\bar{O_c}$, $\bar{O_a}$. Thus, the transitive closure would contain an arc from $O_a$ to $\bar{O_a}$. This fixes $O_a = 0$, meaning that $a$ is not observable. Thus, both s-a-1 and s-a-0 faults on $a$ are redundant. Similarly, we find that the transitive closure will have an arc from $c$ to $\bar{O_c}$. This means that whenever $c = 1$, its observability becomes false. Hence, the fault $c$ s-a-0 cannot be tested. Also, the path from $b$ to $\bar{O_{bc}}$ will cause an edge between those two nodes, implying that $bc$ (fanout of $b$ into $c$) s-a-0 is redundant. Thus, four redundant faults are identified in this example.

The advantage of the method is that one computation of transitive closure finds a group of redundant faults. In comparison, an ATPG method will repeat the process for each redundant fault. On the negative side, not all redundant faults will be found by our method. On comparison with another fault-independent technique [11], we find that the sets of redundant faults identified by the two methods are not always the same.

Furthermore, we make an interesting observation from this example. If we trace paths starting at node $c$, we find that it implies $\bar{O_a}$, $\bar{O_{bc}}$ and $\bar{O_c}$. These implications will be represented by direct arcs form $c$ in the transitive closure graph. If we activate the $c$ s-a-0 fault,

Table 1: Boolean expressions for signals and observability variables

| Gate | Variables | Pair-wise terms | Higher-order terms |
|---|---|---|---|
| AND | $a, b, c$ | $\bar{a}c$ and $\bar{b}c$ | $ab\bar{c}$ |
| OR | $b, c, d$ | $c\bar{d}$ and $b\bar{d}$ | $\bar{b}\bar{c}d$ |
| AND | $O_a, O_{bc}, O_c$ | $O_a\bar{b}$, $O_a\bar{O}_c$, $O_{bc}\bar{a}$ and $O_{bc}\bar{O}_c$ | |
| OR | $O_{bd}, O_c, O_d$ | $O_{bd}c$, $O_{bd}\bar{O}_d$, $O_cb$ and $O_c\bar{O}_d$ | |

then lines $a$, $bc$ (fanout of $b$ into $c$) and $c$ become unobservable. A multiple fault consisting of $c$ s-a-0, $a$ s-a-1 and $bc$ s-a-1 is, therefore, redundant. Identification of multiple faults can be useful in logic minimization and needs further study.

## 3 Redundancy Identification Procedure

Redundant faults in combinational circuits can be classified into three classes, namely, *unexcitable*, *unpropagatable*, and *undrivable* redundant faults. A fault is *unexcitable* redundant if a value opposite to the faulty value cannot be achieved in the fault-free circuit, i.e., we cannot activate the fault. An *unpropagatable* redundant fault means that there is no sensitizable path from the fault site to a primary output. For a *undrivable* redundant fault, although both excitation and path sensitization conditions are possible, the two conditions are not satisfied by the same vector.

### 3.1 Observability Variables

As explained in the last section, we represent the state of a signal in the circuit by two binary variables. The first variable is the binary value of the signal. The second represents the observability status of the signal at primary outputs. We assign an observability variable, $O_x$, to every circuit line $x$. The variable assumes the logic value 1 only when $x$ is observable at a primary output. This is similar to the path sensitization variable used by Chakradhar *et al.* [1] for a specific fault. For a fanout stem and its fanout branches, the same path sensitization variable is used, but the respective observability variables are different. In our analysis, no specific fault or ATPG circuit is considered as is done by Chakradhar *et al.* [1].

### 3.2 Deriving Boolean Expressions

The Boolean expression is the logical OR of two types of constraints. The first is a functional constraint. It is the Boolean false function of logic gates in the circuit. It is derived in the same way as in the neural network or Boolean satisfiability methods [18]. The second is observability constraint based on the local relationships of observability.

Two kinds of observability constraints are added to the Boolean expression. They depend on the specific interconnection topology of gates and the gate types used in the circuit. First, when an input of a gate is observed at the gate output, all other inputs of the gate must have noncontrolling values. Second, an input of a gate is observable at a primary output (PO) only if the gate output is observable at PO. Boolean expressions specifying observability constraints for all gate types are derived as follows:

1. AND and NAND gates: The noncontrolling value for AND and NAND gates is 1. If the input $x_i$ has its observability true, then all other gate inputs must be 1. For the input $x_i$ of an $n$ input gate, we have:

$$O_{xi}\bar{x}_1 + O_{xi}\bar{x}_2 + ... + O_{xi}\bar{x}_{i-1} + O_{xi}\bar{x}_{i+1} + ... + O_{xi}\bar{x}_n + O_{xi}\bar{O}_y = 0 \quad (1)$$

2. OR and NOR gates: The noncontrolling value for OR and NOR gates is 0. Therefore, for input $x_i$, we have:

$$O_{xi}x_1 + O_{xi}x_2 + ... + O_{xi}x_{i-1} + O_{xi}x_{i+1} + ... + O_{xi}x_n + O_{xi}\bar{O}_y = 0 \quad (2)$$

3. BUFFER, NOT, XOR and XNOR gates: These gates do not have any controlling value. We thus have:

$$O_{xi}\bar{O}_y = 0 \quad (3)$$

### 3.3 Identifying Redundant Faults

For the disjunction of Boolean expressions of all controllability and observability variables, an implication graph is constructed. The transitive closure of the implication graph is then computed. Any signal assignments or relations found from the transitive closure are

Table 2: Redundancies in ISCAS-85 circuits found by FIRE and TC-RID

| Circuit | Total Red. Flts. | Ident. Red. Flts. | | Sparc-2 CPU Sec. | |
|---|---|---|---|---|---|
| | | FIRE [11] | TC-RID | FIRE [11] | TC-RID |
| c1908 | 7 | 4 | 2 | 1.3 | 2.5 |
| c2670 | 115 | 29 | 23 | 1.1 | 2.1 |
| c3540 | 105 | 93 | 54 | 5.7 | 12.7 |
| c5315 | 59 | 20 | 20 | 2.1 | 4.5 |
| c6288 | 33 | 33 | 31 | 1.0 | 2.2 |
| c7552 | 131 | 30 | 32 | 3.5 | 7.6 |

substituted in the higher-order relations of state variables, some of which reduce to binary relations. This step is repeatedly executed until no more reduction of terms is possible. In the final transitive closure, we examine all signals for the following dependencies:

1. A signal state and its complement imply each other (*contradiction*).
   If both arcs $(x, \bar{x})$ and $(\bar{x}, x)$ exist, then $x$ can neither be 1 nor 0. Thus, the function of the circuit is independent of the signal $x$, and both stuck-at faults on $x$ are trivially redundant.

2. One value of a signal implies its other value (*fixation*).
   If only arc $(x, \bar{x})$ occurs and there is no other arc between literals $x$ and $\bar{x}$, signal $x$ is fixed at the value 0 in the fault-free circuit. Therefore, fault $s-a-0$ on line $x$ is unexcitable redundant. Similarly, if the transitive closure only consists of arc $(\bar{x}, x)$, $s-a-1$ fault on line $x$ is unexcitable redundant.

3. The true status of observability variable implies its own false observability status.
   If the transitive closure includes arc $(O_x, \bar{O}_x)$ , then the observability variable $O_x$ is fixed at the value 0. Therefore, line $x$ can not be observed from primary outputs. Hence, both $s-a-0$ and $s-a-1$ faults on line $x$ are unpropagatable redundant.

4. A Certain value of a signal implies its false observability status.
   If arc $(x, \bar{O}_x)$ or $(\bar{x}, \bar{O}_x)$ occurs, it implies that whenever $x$ assumes the logic value 1 or 0, it becomes unobservable at primary outputs, causing the $s-a-0$ or $s-a-1$ fault on $x$ to become untestable. Since the conditions of fault excitation and path sensitization can not be satisfied by the same vector, the fault is undrivable redundant.

### 3.4 Algorithm

The algorithm is outlined below:

1. Assign a signal value variable and an observability variable to each circuit line.

2. Derive Boolean expressions representing the local relationships of the state variables.

3. Compute the transitive closure.

4. In the final transitive closure, for each line:
   Observe conditions for redundancy identification:

   [a] If a contradiction exists, identify both faults as trivially redundant.

   [b] If one value of the line implies its other value, then one of the stuck-at faults on that line is redundant.

   [c] If the true status of observability variable implies its own false observability status, then both stuck-at faults on the line are redundant.

   [d] If a certain value of the line implies its false observability status, then the corresponding stuck-at fault is redundant;

The algorithm is implemented in a C language program TC-RID, which runs on Sun Sparc-2 670/MP server.

## 4 Results

The results are shown in Table 2, which also gives the data for the method of Iyer and Abramovici, called FIRE [11]. Both methods are fault-independent, but solve the redundancy identification problem in very different ways. FIRE performs uncontrollability and unobservability propagation around fanout and reconvergence regions. We map the redundancy identification problem into the Boolean domain and use transitive closure to obtain global signal dependencies.

Table 3: Redundancy classes of identified redundant faults

| Circuit | Tot. Red. Flts. | Tot. Red. Flts. Ident. | Unexcitable Red. Flts. | Unpropagatable Red. Flts. | Undrivable Red. Flts |
|---|---|---|---|---|---|
| c432 | 4 | 0 | 0 | 0 | 0 |
| c499 | 8 | 0 | 0 | 0 | 0 |
| c880 | 0 | 0 | 0 | 0 | 0 |
| c1355 | 8 | 0 | 0 | 0 | 0 |
| c1908 | 7 | 2 | 0 | 0 | 2 |
| c2670 | 115 | 23 | 3 | 3 | 17 |
| c3540 | 105 | 54 | 0 | 8 | 46 |
| c5315 | 59 | 20 | 1 | 1 | 18 |
| c6288 | 33 | 31 | 15 | 16 | 0 |
| c7552 | 131 | 32 | 2 | 0 | 30 |

Iyer and Abramovici's implication of uncontrollability and unobservability is a linear process. Such implication may be performed separately on all fanout and reconvergent fanout points. The entire process is thus $O(n^2)$, where $n$ is the number of lines in the circuit. The complexity of the conventional transitive closure algorithm is $O(n^3)$ in the worst case. However, as pointed out by Chakradhar *et al.* [1], several factors contribute to an almost linear run time complexity of the transitive closure ATPG method. Our present implementation is not running at its best.

Table 2 shows that our results are similar to Iyer and Abramovici in both CPU time and the number of redundant faults identified. In the present implementation, our program takes twice as long. The number of redundant faults is also slightly different. In cases like c2670, where many redundant faults are not found by either method, it will be interesting to determine whether the two methods identify different sets of faults.

The new method also classifies redundant faults. The results are shown in Table 3. The number of *Tot. Red. Flts. Ident.* is broken down into three categories shown in the last three columns. No *trivially* redundant faults were found in these circuits. Column *Unexcitable Red. Flts.* is the number of redundant faults because certain signal values were impossible at fault sites. Column *Unpropagatable Red. Flts.* gives the number of redundant faults due to no sensitizable path from the fault site to a primary output. The faults for which a simultaneous activation and propagation was not possible are shown in Column *Undrivable Red. Flts.*

For c6288, we identify 31 out of 33 redundant faults and FIRE [11] identifies all. The two remaining faults are fanout stem faults and point to a limitation of our method. These faults are both unpropagatable redundant. Our technique can not identify this kind of unpropagatable fault because fault effects with different parities cancel out at reconvergent gates. This points to a direction for future improvement in the method. The observability variable and the Boolean expression may be enhanced to represent different parities of the observability variable.

## 5 Conclusion

In this paper, we present a one-pass redundancy identification method for combinational circuits using transitive closure. A Boolean expression is used to specify local relationships of binary signal value variables and observability variables. A graph algorithm using transitive closure then obtains global variable dependencies and identifies redundancies.

Preliminary results on ISCAS '85 benchmark circuits show that this approach efficiently identifies and classifies redundancy. For c6288, we identify 31 out of 33 redundancies. The speedup for this circuit is 166 times over an ATPG program. This method can be easily incorporated in a transitive closure based energy minimization test generator. The faults not identified by the one-pass procedure will require input signal assignments, backtracks, etc., as is normally done by the ATPG program.

The usage of transitive closure to identify redundancy has several advantages. First, transitive closure has been shown to be a fast and practical method in test generation [1, 19, 20]. Techniques used to speed up transitive closure computation are applicable to our method. Second, transitive closure can derive many logical conclusions implied by binary relations. Recent work shows that the efficiency of the method can be enhanced by incorporating higher order expressions in

the implication graph [20]. Third, the transitive closure is easily parallelized. Moreover, combinational circuit redundancy identification has been applied to redundancy identification in sequential circuits [13], we can use this theory to extend our one-pass transitive closure method for redundancy identification in combinational circuits to sequential circuits. As pointed out at the end of Section 2, the presented technique may have further application to the identification of redundant multiple faults.

**Acknowledgment:** Research reported here was supported by the Semiconductor Research Corporation under Contract 95-DJ-706 and by the CAIP Research Center, an Advanced Technology Center of the New Jersey Commission on Science and Technology at Rutgers University, New Jersey.

## References

[1] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Trans. Computer Aided Design*, vol. 12, pp. 1015–1028, July 1993.

[2] X. Chen and M. L. Bushnell, *Efficient Branch and Bound Search with Application to Computer-Aided Design.* Boston: Kluwer Academic Publishers, 1996.

[3] H. Cox and J. Rajski, "On Necessary and Nonconflicting Assignments in Algorithmic Test Pattern Generation," *IEEE Transactions on Computer Aided Design*, vol. 13, pp. 515–530, April 1994.

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

[5] M. H. Schulz and E. Auth, "Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification," *IEEE Transactions on Computer-Aided Design*, vol. 8, pp. 811–815, July 1989.

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

[7] M. Abramovici, D. T. Miller, and R. K. Roy, "Dynamic Redundancy Identification in Automatic Test Generation," *IEEE Transactions on Computer-Aided Design*, vol. 11, pp. 404–407, March 1992.

[8] M. Abramovici and M. A. Iyer, "One-Pass Redundancy Identification and Removal," in *Proceedings of the IEEE International Test Conference*, pp. 807–815, September 1992.

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

[10] P. R. Menon and H. Ahuja, "Redundancy Removal and Simplification of Combinational Circuits," in *Proceedings of the 10th IEEE VLSI Test Symposium*, pp. 268–273, April 1992.

[11] M. A. Iyer and M. Abramovici, "Low-Cost Redundancy Identification for Combinational Circuits," in *Proc. 7th International Conf. on VLSI design*, pp. 315–317, January 1994.

[12] M. A. Iyer and M. Abramovici, "Sequentially Untestable Faults Identified Without Search," in *Proceedings of the IEEE International Test Conference*, pp. 259–265, October 1994.

[13] V. D. Agrawal and S. T. Chakradhar, "Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits," *IEEE Transactions on Computer Aided Design*, vol. 14, pp. 1155–1160, September 1995.

[14] I. Pomeranz and S. M. Reddy, "On Identifying Undetectable and Redundant Faults in Synchronous Sequential Circuits," in *Proceedings of the 12th IEEE VLSI Test Symposium*, pp. 8–14, April 1994.

[15] H. C. Liang, C. L. Lee, and J. E. Chen, "A Sequentially Redundant Fault Identification Scheme and Its Application to Test Generation," in *Proceedings of the Third Asian Test Symposium*, pp. 57–62, November 1994.

[16] H. C. Liang, C. L. Lee, and J. E. Chen, "Identifying Untestable Faults in Sequential Circuits," *IEEE Design & Test of Computers*, vol. 12, pp. 14–23, Fall 1995.

[17] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, *Neural Models and Algorithms for Digital Testing.* Boston: Kluwer Academic Publishers, 1991.

[18] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Neural Net And Boolean Satisfiability Models Of Logic Circuits," *IEEE Design & Test of Computers*, vol. 7, pp. 54–57, October 1990.

[19] S. T. Chakradhar, M. A. Iyer, and V. D. Agrawal, "Energy Model for Delay Testing," *IEEE Transactions on Computer-Aided Design*, vol. 14, pp. 728–739, June 1995.

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