On More Efficient Combinational ATPG using Functional Learning

Rajarshi Mukherjee[†] Jawahar Jain[‡] Masahiro Fujita[‡] Jacob A. Abraham[†] Donald S. Fussell[†]
†Computer Engineering Research Center
University of Texas
Austin, TX 78713
‡Fujitsu Laboratories of America
3350 Scott Blvd., Bldg. #34, Santa Clara
CA 95054-3104

Abstract

Learning techniques like SOCRATES and recursive learning have greatly enhanced the technology of FAN-based ATPG. In this paper we present a test generation methodology for combinational circuits using functional learning, discuss application of novel functional information to enhance ATPG and present ATPG results on ISCAS 85 benchmark circuits. The test generation methodology combines the use of structural (topology) based analysis methods with the function representation techniques (such as BDDs).

1 Introduction

Traditional decision-tree based ATPG techniques like PODEM [6] and FAN [4] have been greatly enhanced by learning techniques like SOCRATES [15, 16] and recursive learning [10] that allow a more efficient pruning of the search space by identifying signal assignments (necessary assignments) in the circuit that cannot be obtained by a simple direct simulation of the value assignments. However, both SOCRATES and recursive learning can learn only unique Boolean implications, referred to as *constant-value implications.* Functional learning (FL) [11, 12, 8], on the other hand, can learn both constant-value implications and complex functional relations, for instance, the disjunction of the values at gates g and h is always true.

In this paper we present preliminary results of an FL-based ATPG tool. We intend to show that FL is a powerful tool for analysis during ATPG and can help speed up test generation and redundancy identification. We discuss methods to use novel functional information for more efficient test generation and show that use of OBDDs does not create any bottleneck in space requirement during ATPG. The present ATPG tool uses the BDD package of the Carnegie Mellon University [2].

The rest of the paper is organized as follows: in section 2 we illustrate the principle of FL. Some theoretical aspects of FL are presented in section 3. Section 4 presents the overall strategy for ATPG. Application of functional relations for efficient pruning of search space has been investigated in section 5. The results of ATPG on ISCAS 85 benchmark circuits are presented in section 6. Section 7 presents the conclusions.

2 Functional Learning : an overview

This section illustrates the basic concept of FL. Consider the circuit shown in Fig. 1 . In this figure lowercase letters are used to name the wires in the circuit and the uppercase letters are used to refer to the Boolean functions realized at the wires. Consider the wire 23 to be unjustified to a 1. Let the OBDD for this wire be built in terms of the pseudo inputs a, b and c. Let the OBDD of the wire 23 be denoted as **G**. Let the OBDD for the wire 16 built in terms of the same set of pseudo inputs be denoted as **H**. The two OBDDs are shown in Fig. 2. Now, if we take an AND of $\overline{G}$ and **H** we find that the result is the OBDD **H**. This means that when **G** is a Boolean 1, **H** is a Boolean 0. Hence, it is learned that when the wire 23 carries a value of Boolean 1, the wire 16 must carry a value of Boolean 0. Then by forward implication it is learned that for this value condition in the circuit, wire 24 must be a Boolean 1. Hence from our initial condition that wire 23 is a Boolean 1, we learn that Boolean 0 on wire 16 and a Boolean 1 on wire 24 are *necessary conditions.*

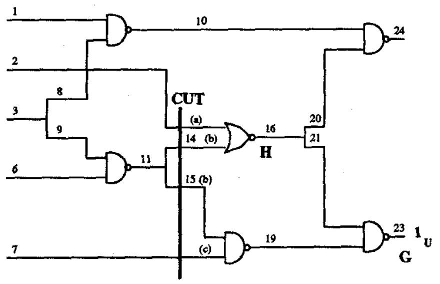

Figure 1: *Functional learning*

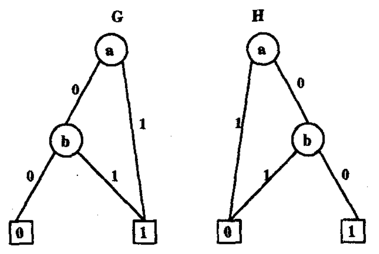

Figure 2: *OBDDs for wire 23 and 16*

3 Theoretical aspects of FL

Here we discuss some theoretical properties of FL, highlighting the features that make it a more powerful learning technique than other constant-value learning methods. The proofs of the theorems are omitted due to space constraint. Let us consider two OBDDs G and H.

Theorem 3.1 a. *If $G = 1 \Rightarrow H = 1$ then $G \bigwedge H = G$. Conversely, if $G \bigwedge H = G$, then $G = 1 \Rightarrow H = 1$. By the law of contrapositum $H = 0 \Rightarrow G = 0$.*

b. *If $G = 1 \Rightarrow H = 0$ then $\overline{G} \bigwedge H = H$. The converse is also true. That is if $\overline{G} \bigwedge H = H$, then $H = 1 \Rightarrow G = 0$. By the law of contrapositum $G = 1 \Rightarrow H = 0$.*

Conditions for other constant-value learnings can be similarly stated; for such details, and a proof of the above theorem, please refer to [11].

Through the use of OBDDs FL is able to efficiently capture and manipulate more information about the given circuit than is possible by the structural learning techniques. This fact is demonstrated below through an example that compares FL with RL.

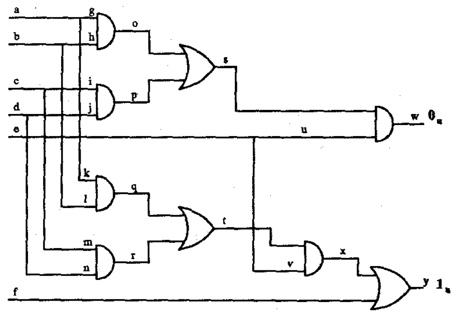

Figure 3: *Circuit to demonstrate power of functional learning*

Consider the circuit shown in Fig. 3. This example has been taken from [10]. The two initial value conditions in the circuit are wire $w = 0$ and wire $y = 1$. Both the wires are unjustified. Now, as shown in [10], if we apply recursive learning to the circuit then we learn that $x = 0$ and $f = 1$ are necessary conditions.

Now, by examining the structure of the circuit we see that the wires s and t realize the same function. So, definitely for any condition of value assignments in the circuit, wires s and t must carry the same Boolean value. This value may not be a *necessary condition.* As dictated by the *theorem* 3.1, FL determines the OBDD at each wire in terms of a certain set of internal variables in the circuit and imposes on this OBDD the restrictions imposed on the circuit by the current situation of value assignments. Hence, FL determines the restricted functions realized at the wires in a circuit. Structural methods, on the other hand, cannot maintain this information. Using this information, it can be easily determined that wires s and t realize identical functions because identical restricted OBDDs are associated with them. Although no unique value is learned at the wires s and t, we are able to derive information about the set of values that can be assigned to the wires s and t. This information can be useful while making optional assignments during test generation using FAN.

Assume that G and H are two Boolean functions at gates g and h in a digital circuit. Let $b, c \in \{0, 1\}$. We denote $L(b)_G$ as the set of all learnings in the circuit for function $G = b$, and similarly $L(c)_H$ as the set of all learnings in the circuit for $H = c$. Assume $G = b \Rightarrow H = c$. Then it can be seen that:

Observation 3.1 $L(c)_H \subseteq L(b)_G$. *By the law of contrapositum it follows that* $L(\overline{b})_G \subseteq L(\overline{c})_H$.

Assume that $S_f(L)$ is the set of conditions learned when FL operates at a distance L from the gate at which learning is taking place. It can be proved that :

Theorem 3.2 $S_f(L) \supseteq S_f(L-1) \cup S_f(L-2) \cup \ldots \cup S_f(1)$

For an intuition behind this theorem please refer to [12]. Using these theorems groups of wires in the circuit can be identified such that no explicit learning operations are necessary on them.

4 Test generation strategy

Test generation proceeds in four phases : phase 1 - random pattern test generation to eliminate easy to detect faults; phase 2 - FAN with a small backtrack limit and with ordinary heuristics; phase 3 - FAN using heuristics orthogonal to the ones used in phase 2; phase 4 - remaining hard faults targeted with FAN and FL. In phase 4 whenever there is a conflict during the FAN algorithm, instead of performing a backtrack, the last decision on the decision stack is assigned a value 'X' and the learning routine is called. If it is learned that the last decision on the decision stack has to be reversed then the reversal is done and FAN is continued. Otherwise, the learning level is increased and learning is carried out with higher precision. FL aborts if a preset maximum learning level is exceeded.

The details of this algorithm are shown in the flow diagram (Fig. 4). Fig. 4(b) shows the internal details of the block for FAN with FL in Fig. 4(a). Note that in Fig. 4(a) phase 2 and phase 3 of the algorithm have been grouped together in the block "FAN with ordinary heuristics".

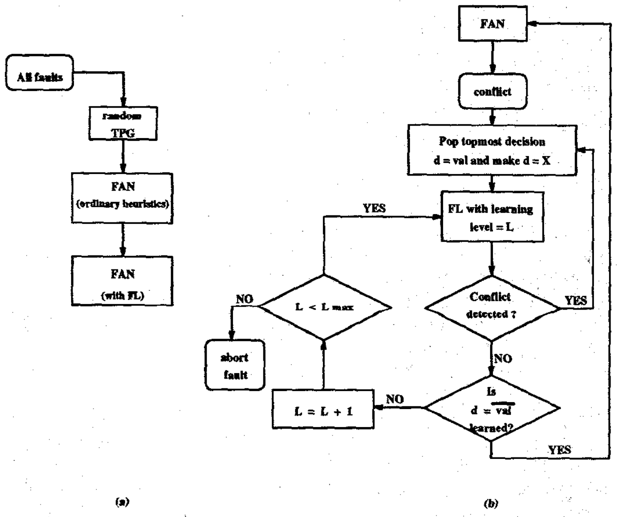
All faults
random TPG
FAN (ordinary heuristics)
FAN (with FL)
FAN
conflict
Pop topmost decision d = val and make d = X
FL with learning level = L
Conflict detected ?
YES
NO
Is d = $\overline{val}$ learned?
YES
NO
L = L + 1
L < L max
YES
NO
abort fault
(a)
(b)

Figure 4: *Flow diagram for ATPG*

5 Application of functional relations for more efficient pruning of search space

FL can be used to derive functional relations based on a situation of value assignments in a circuit. These relations, if used effectively, allow us to detect conflicts earlier, thus reducing time between backtracks. It can also help speed up redundancy identification.

Note learning $a = 0 \Rightarrow b = 1$ can be written as as a tautology expression (invariant) $(\overline{a} \wedge b) \equiv \overline{a}$. The restriction conditions derived from the learning relations can be used to greatly simplify problems of BDD construction as well as in reasoning about feasible test search space.

Consider $F_i(\lambda)$ to represent the BDD for some output F_i of circuits C expressed in terms of a decomposition set $\lambda = \{\psi_i, \ldots, \psi_m\}$, that is, a cutset λ of intermediate gates $\{\psi_i, \ldots, \psi_m\}$. [1] A canonical expression for $F(x_1, \ldots, x_n)$ for F_i in terms of primary input variables $x_1, \ldots, x_n$ can be obtained by composing each ψ_i in $F_i(\lambda)$. Let $R = \{\tau_1, \ldots, \tau_k\}$ be k restriction conditions discovered between points (gates) that are on or ahead of the cutset λ.

It is easy to prove:

Theorem 5.1 *$F_i(x_1, \ldots, x_n)$ is not satisfiable if $F_i(\lambda)$ or $F_i(\lambda) \wedge \tau_i$, $1 \leq i \leq k$ is not satisfiable.*

The above theorem states that any restriction condition conjuncted with $F_i(\lambda)$ cannot change the functionality of F_i. Obviously we can replace τ_i with a conjunction of any plurality of restriction conditions in R. The above exercise enables us to develop a useful lemma for ATPG.

In the above context, consider testing a fault g_{s-a-v}, where $v \in \{0, 1\}$. Let under some value assignment in the circuit a set R of learning relations, that is restriction conditions, be derived. Let G be a set of gates $\{g_1, \ldots, g_h\}$ with some corresponding value assignments. Each value assignment can obviously be expressed as a BDD in terms of the cut λ.

Theorem 5.2 *A value assignment $b \in \{0, 1\}$ to gate p will lead to conflict if BDD for $p = b$ is orthogonal to BDD $g_i = c$ for any value assignment $g_i = c$ in G.*

This leads to the following corollary:

Corollary 5.1 *A value assignment b to gate p will lead to conflict if BDD for $p = b$ is orthogonal to BDD $g_i = c \wedge \tau_j$ for any value assignment $(g_i = c) \in G$, $\tau_j \in R$ in G.*

In applying this theorem we must note the following facts

1) Learning can precede testing and hence a set of gates which can constitute λ in different regions of circuit can be easily precomputed.

2) It is possible to reduce the set of learning conditions which can be beneficial to be analyzed in context of the above presented corollary as discussed below.

Let λ be some cut through the circuit. Let R_c be the set of all learning conditions that were discovered using BDD operations between graphs encoded through a set of cuts $\kappa_{cut} = \kappa_1, \ldots, \kappa_m$, where each κ_i is a cut for some intermediate gates. Also let each $\{\kappa_i \in \kappa_{cut}\}$ be such that κ_i is covered by λ. We define an intermediate cut κ_i to be covered by cut λ if there exists no path from primary inputs to a gate in κ_i which does not pass through a gate in λ.

Theorem 5.3 *No learning condition invariant made from the learning relationships in set R_c can help minimize the BDD size of $F_i(\lambda)$ through a Boolean operation between the learning condition invariant and $F_i(\lambda)$.*

Intuitively, in the above theorem we are stating that under the above mentioned conditions, BDDs of numerous conditions in R_c will actually reduce to a Boolean 1, and need not be built at all. This can drastically reduce the total number of invariants that must be examined.

6 Results

In this section the results of ATPG on the ISCAS 85 benchmark circuits are presented. A *100 % fault coverage* (including test vector generation for testable faults and identification of the redundant faults) has been obtained for all the circuits reported. In addition, the sizes of the OBDDs that had to be built were extremely small, thus making the use of OBDDs in the ATPG framework viable. In this work very simple depth-first ordering heuristics have been used for ordering the OBDD variables. More sophisticated ordering techniques like [7, 14] would require smaller BDDs to be built and hence make the application of FL during ATPG even more efficient. Also note, that the results have been obtained from an unoptimized program written in C. Therefore, the times reported are not the fastest that could possibly be obtained.

We present the results of ATPG on the suite of ISCAS benchmark circuits in Table 1. The experiments were conducted on a sparc station 10 and the times reported are in seconds. The results indicate that FL is a very powerful tool for redundancy identification and creates no space bottleneck as only very small BDDs have to be built.

The first column gives the circuit under consideration. The second column lists the total number of faults in the circuit that were targeted by the ATPG tool. The third column gives the time spent for random pattern test generation (RTPG). The fourth column lists the number of faults caught by RTPG. The fifth column gives the time spent for completing pass 1. The column 6 gives the number of faults that were proved to be redundant (R) and the number of faults that were aborted (A) after the completion of pass 1. This set of statistics will henceforth be referred to as *fault status*. The faults that were aborted by pass 1 are now targeted during pass 2. The column 7 gives the time for pass 2. Column 8 lists the faults status after pass 2. The faults aborted by pass 2 are next targeted in pass 3. The time for pass 3 is listed in the column 9. The fault status after pass 3 is listed in column 10. The maximum BDD size (max BDD size) that was required in the *owc* during FL is listed in Column 11. The column 12 lists the total test generation time for the ATPG tool.

7 Conclusions

In this paper we have discussed application of FL to combinational ATPG, showing how novel functional in-

[1] Any decomposition set of internal gates $\{\psi_i, \ldots, \psi_m\}$ for some output F must necessarily be a cutset for the circuit representation of F and vice-versa.

Table 1: Results of ATPG on ISCAS 85 circuits

circuit	target	RTPG	flts	pass 1	flts	pass 2	flts	pass 3	flts	max BDD size	total time
c432	524	3.8	511	1.84	1R, 3A	0.55	3A	3.23	3R	79	9.42
c499	758	4.88	729	5.27	8R, 0A	-	-	-	-	-	10.15
c880	942	10.62	896	2.89	0R, 0A	-	-	-	-	-	13.69
c1355	1574	3.5	691	345.7	8R, 76A	16.0	-	-	-	-	349.2
c1908	1879	12.6	1648	347.46	7R, 15A	6.23	2A	3.11	2R	6	369.4
c3540	3428	356.19	3125	94.55	137R, 0A	-	-	-	-	-	451.5
c5315	5350	176.0	5007	367.32	58R, 4A	2.87	1R	-	-	-	546.19
c6288	7740	102.96	7708	4.75	34R, 0A	-	-	-	-	-	110.13

formation can be used to speed up detection of conflicts and redundant faults. Based on the theoretical and experimental results obtained in this publication and elsewhere [12, 8] we believe that FL is a powerful method to learn indirect implications. Thus, a proper application of functional learning technique has a potential to speed up detection of conflicts and redundancy identification. The resulting test generation methodology combines the use of structural (topology) based analysis methods with the BDD based function representation techniques. Due to the use of compact function representations techniques, the testing scheme discussed may be highly suitable for the ATPG problems which require traversing a very large search space. In addition, since usually only relatively small BDDs are required, FL does not create a space bottleneck.

References

[1] Bryant R. E., "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers, vol. C-35, no. 8, Aug. 1986, pp. 667-691.

[2] Brace K. S., Rudell R. L., Bryant R. E., "Efficient Implementation of a BDD Package", Proc. 27th Design Automation Conf., 1990, pp. 40-45.

[3] Fujiwara H., Toida S., "The Complexity of Fault Detection Problems for Combinational Logic Circuits", IEEE Transactions on computers, vol. C-31, June 1982, pp. 555-560.

[4] Fujiwara H., Shimono T., "On the Acceleration of Test Generation Algorithms", Proc. 13th Int. Symp. Fault Tolerant Computing, 1983, pp. 98-105.

[5] R. K. Gaede, M. R. Mercer and K. M. Butler, "CATAPULT: Concurrent automatic testing allowing parallelization and using limited topology", Proc. 25th DAC, June 1988, pp. 597-600.

[6] Goel P., "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE Transactions on Computers, vol. C-30, Mar. 1981, pp. 215-222.

[7] Jain J., Bitner J., Moundanos D., Fussell D. S., Abraham J. A., "A New Scheme to Compute Variable Order for Binary Decision Diagrams", Fourth Great Lakes Symposium on VLSI, March 1994, pp. 105-108.

[8] Jain J., Mukherjee R., Fujita M., "Advanced Verification Techniques Based on Learning", 32nd Design Automation Conf., June 1995, pp. 420-426.

[9] Jain J., Mukherjee R., Fujita M., "Verification Techniques Based on Functional Learning", Technical Report No. FLA-CPS95-01, Fujitsu Laboratories of America, 1995.

[10] Kunz W., Pradhan D. K., "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits", Proc. Int. Test Conf., 1992, pp. 816-825.

[11] Mukherjee R., Jain J., Pradhan D. K., "Functional Learning: A new approach to learning in digital circuits", Proc. IEEE VLSI Test Symp., April 1994, pp. 122-127.

[12] Mukherjee R., Jain J., Fujita M., "VERIFUL : VERIfication using FUnctional Learning", Proc. European Design and Test Conf., March 1995, pp. 444-448.

[13] Mukherjee R., Jain J., Fujita M., Abraham J. A., Fussell D. S., "Observations on Verification Techniques Based On Learning", Intl. Workshop on Logic Synthesis, May 1995, section 6, pp. 31-40.

[14] Rudell R., "Dynamic Variable Ordering for Ordered Binary Decision Diagrams", Proc. ICCAD, 1993, pp. 42-47.

[15] Schulz M., Trischler E., Safert T., "SOCRATES: A highly efficient automatic test pattern generation system", IEEE Transactions on CAD, vol. 7, Jan. 1988, pp. 126-137.

[16] Schulz M., Auth E., "Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification", IEEE Transactions on Computer-Aided Design, vol. 8, NO. 7, July 1989, pp. 811-816.

[17] Shiple T. R., Hojati R., Brayton R. K., "Heuristic minimization of BDDs using don't cares", DAC, 1994, pp. 225-231.

[18] T. Stanion and D. Bhattacharya, "TSUNAMI: A path oriented scheme for algebraic test generation", Proc. ITC, 1993.