Functional ATPG for Delay Faults *

S. Tragoudas M. Michael
Electrical and Computer Engineering Department
The University of Arizona
Tucson, AZ 85721

Abstract This paper presents a functional level ATPG tool for delay faults which handles all existing fault models. The tool generates patterns using either binary decision diagrams or boolean satisfiability. Experimental results are presented on the ISCAS'85 benchmarks.

1 Introduction

Delay testing focuses on timing defects, which could degrade the performance of the circuit–under–test (CUT). Functional test generation can be used to identify testability problems before an implementation is selected. In addition, the problem arises when a gate–level description of an implemented CUT is not available. This is the case in embedded core designs with intellectual property considerations.

A fault is a tuple (I, O, tI, tO), where I is a CUT input, O is a CUT output, tI is a rising (r) or falling (f) transition at I and tO is a r or f transition at O. [14, 9, 10] present fault models for functional ATPG. Under model M1, only one pair of test patterns must be generated per fault [14]. Under model M2, Δ different test patterns are needed per fault [9]. Δ is a positive integer, always in the low hundreds, and is given as an input parameter for each CUT [9]. [10] proposed that all possible patterns are generated for each fault but they showed later that Δ patterns per fault target all path delay faults in some gate–level implementation of the CUT [9].

Each generated pair of patterns may have single–input or multi–input transitions. In the case where logic hazards exist, only single–input transition (SIT) pairs of test patterns, referred to as SIT tests, can be considered for function–robust propagation. Results in [9] indicate that SIT tests are more effective than multi–input transition (MIT) tests. Whenever SIT patterns cannot be found, MIT tests must be generated.

Every generated test must function–robustly propagate a transition tI from input I as a tO transition to an output O. This happens when the value on O changes if and only if the value on I changes. We call this property the function robust propagation (FRP) property.

Given a pair of test patterns (T_1, T_2), let $\mathcal{V} = \{i_1, .., i_k\}$ denote the set of variables that are assigned different values in T_1 and T_2. Let f denote the function of output O. The FRP property for (T_1, T_2) is satisfied if and only if all patterns that satisfy conditions (i) through (iii) below evaluate f to $f(T_1)$.
(i) Variable I is set to its value in T_1.
(ii) All variables $\notin \mathcal{V}$ are set to their value in T_1, T_2.
(iii) The variables in $\mathcal{V}$ are assigned to any value.

Observe that any generated SIT test always satisfies the FRP property. For any generated MIT test, the ATPG tool must explicitly determine whether the FRP property is satisfied. Condition (iii) implies that a brute–force approach requires $\Theta(2^k)$ time which is prohibitive when k is large.

No systematic methodology for functional test generation for delay faults has been proposed in the literature. This paper presents an ATPG tool that generates pairs of patterns using either boolean satisfiability (SAT) or reduced ordered binary decision diagrams (ROBDDs). The choice depends on the fault model and whether SIT or MIT tests are generated. Section 2 outlines advantages and disadvantages of each framework (SAT vs ROBDD). Section 3 provides an experimental comparison of the two frameworks on the functions of the ISCAS'85 benchmarks. The ATPG tool, described in Section 4, distinguishes between model M1 or model M2 and combines the above frameworks. For each model, it initially generates SIT tests and then the appropriate number of MIT tests.

2 The Underlining Frameworks and Related Theory

Established frameworks such as SAT and ROBDDs can be utilized to solve the problem of functional delay fault test generation. This section lists the properties (advantages and disadvantages) of each framework. More details can be found in [12], and for SIT

*Partially supported by NSF grant CCR-9815229

0-7695-0104-4/99 $10.00 © 1999 IEEE

ATPG also in [13]. We distinguish between models M1 and M2 and also between SIT and MIT tests.

The problem of finding a set of test patterns for a given (I, O, tI, tO) tuple can be reduced to that of solving a CNF satisfiability formula which is constructed from the function f of output O. A different satisfiability formula is constructed for each fault (I, O, tI, tO). For example, when $tI = tO = r$ the CNF formula represents function $f_I \cdot \bar{f}_{\bar{I}}$, where f_I denotes the cofactor of f with respect to input I. Subsequently, efficient SAT solvers [7, 4] can be used to solve the constructed CNF satisfiability formula very fast. A test for the examined fault is generated based on the variable assignment in the satisfied formula.

The proposed framework guarantees that the ATPG process for SIT tests under the M1 model is exact, i.e., 100 % fault coverage is obtained for non redundant faults. More precisely, we have shown [12]:

Theorem 2.1 *An (I, O, tI, tO) test is found if and only if the respective CNF formula is satisfied.*

In contrast, the SAT based framework is not exact for SIT tests under the M2 model or for MIT tests under either model. In particular, under the M2 model, the tool cannot guarantee that Δ tests (SIT or MIT) per fault will be generated, even if they exist, and under the M1 model that an MIT test will be generated. However, we have shown that [12]:

Theorem 2.2 *No MIT test exists for fault (I, O, tI, tO) if at least one of two constructed CNF formulas is unsolvable.*

A serious drawback of the framework is that it uses a brute–force approach for determining whether an MIT candidate test (T_1, T_2) satisfies the FRP property. This requires $\Theta(2^k)$ evaluations of function f, where k is the number of variables that have different values in T_1 and T_2. Thus, the framework is not very appealing for MIT test generation.

The second framework for generating tests is based on ROBDDs. The problem of finding the number of SIT tests for a given (I, O, tI, tO) fault amounts to constructing an appropriate ROBDD and counting the number of paths in the ROBDD that terminate at its terminal node 1. For example, when $tI = tO = r$ the ROBDD represents function $f_I \cdot \bar{f}_{\bar{I}}$, where f_I denotes the cofactor of f with respect to input I. The ROBDD based framework is theoretically more powerful than the SAT based framework when generating SIT tests [12]:

Theorem 2.3 *The number of paths that correspond to SIT tests for fault (I, O, tI, tO) can be found in linear time to the size of the constructed ROBDD.*

Thus, the ROBDD based framework is exact for SIT ATPG under both M1 and M2 fault models. The drawback of the framework is that space limitations prevent generating the appropriate ROBDDs for certain faults. When this occurs a non exact (approximate) approach is used that has reduced space complexity. Under the latter approach, the constructed ROBDD represents function f. Thus, in practice the ROBDD based framework may be inferior to the SAT based framework for SIT test generation, especially under model M1.

We define an MIT candidate test to be a pair of test patterns with MIT which has not been examined for the FRP property. An important advantage of the ROBDD framework is that the FRP property of an MIT candidate test (T_1, T_2) can be determined in linear time to the size of the ROBDD. Let $P_{I,O,tI,tO}$ denote the set of all paths for which the respective variables variables, including I, have in T_2 the same value as in T_1.

Theorem 2.4 *The MIT candidate test (T_1, T_2) satisfies the FRP property for fault (I, O, tI, tO) if and only if all paths in $P_{I,O,tI,tO}$ lead to terminal $f(T_1)$.*

Neither the SAT or the ROBDD based framework are exact for MIT test generation. [1] However, the above property of the ROBDD based framework makes it more appealing than the SAT based framework for MIT test generation.

3 Experimental comparison

We compared the two frameworks on a 270MHz SUNW, Ultra–5 workstation. We experimented on the functions of the ISCAS'85 benchmarks. We used the solver of [4] for the SAT based framework and the ROBDD package of [11] for the ROBDD based framework. Our focus here is on comparing the two frameworks for SIT and MIT ATPG under both M1 and M2 models, in terms of fault coverage and time performance. We do not provide any results on the actual fault coverage obtained when the test sets generated are applied to gate–level path delay faults since this was shown in [9] (in order to demonstrate the validity of the two models) and it is beyond the scope of this work.

For large multiplier circuits, such as c6288, even the SAT framework has memory problems because the constructed CNF formula is very big. For such CUTs the approximate ROBDD approach also fails to constructs ROBDDs. In such cases, the ATPG tool preassigns a fraction of the input variables to either the true or false value (using a random assignment) in order to simplify function f so that the two presented frameworks can apply [12].

[1] An exact algorithm has been implemented in [12] for the ROBDD framework but is impractical due to memory limitations.

We first evaluate the performance of the proposed tools for SIT ATPG for model M1. Table 1 gives results for the ROBDD–based ATPG algorithms (columns 3 and 4) as well as the SAT based framework (columns 5 and 6). Column 2 lists the number of faults in each circuit. The ROBDDs were generated using the CUDD Decision Diagram Package [11] and the constructed CNF formulas were solved using the tool in [4]. Column 3 gives the number of generated tests using the exact ROBDD approach. In parenthesis, we list the number of generated tests obtained by an enhanced approach that uses the approximate ROBDD approach when the exact approach fails due to memory problems. Column 4 gives time performance results for the two alternatives. The approximate ROBDD approach has been implemented so that a fault is aborted when no pair of paths has been found within 5 minutes. Columns 5 and 6 give the generated tests and time performance of the SAT based framework. The results clearly show that the SAT based framework is superior.

Table 2 lists information similar to Table 1 for MIT tests under model M1. Both frameworks are non exact. A fault is aborted when no pair of patterns has been found within 5 minutes. The ROBDD based framework outperforms the SAT based framework.

Table 3 provides an experimental comparison for SIT ATPG under model M2. Note that the exact ROBDD-based solution can return the number of SIT tests for every fault. Therefore, its time performance is the same to that for the M1 model. We list results for the enhanced ROBDD based scheme that uses the approximate (non exact) approach for those faults where the exact approach fails due to memory problems. Columns 2 and 4 list the average number of SIT patterns that are generated for each fault. In both frameworks, a fault is aborted when an approximate scheme does not find a pair of patterns within 5 minutes. The results indicate that the ROBDD based framework performs better than the SAT based framework. They also indicate that there exist many SIT tests per fault.

For MIT test generation under model M2 the SAT based framework has poor performance. This is due to the brute–force scenario of generating more than one MIT candidate tests coupled with the brute–force time consuming approach that determines whether the candidate test satisfies the FRP property. Thus, Table 4 lists results only for the ROBDD based framework.

4 The proposed ATPG tool

The proposed ATPG tool accepts as a input parameter the fault model (M1 or M2). If the M1 model is selected it attempts to generate an SIT test for each fault using the SAT based framework. The experimental results of the previous section show that the majority of the faults have an SIT test. For those faults no further action is required. If the CNF formula cannot be constructed, some variables are preassigned as described in Section 2. For those faults that do not have an SIT test, an MIT candidate test is generated using the ROBDD based framework and it is tested for the FRP property using again the ROBDD based framework.

If the M2 model is selected, Δ tests per fault need to be generated. (The ATPG tool accepts the value of Δ as an input parameter.) It uses the ROBDD based framework to determine whether fault (I, O, tI, tO) has Δ or more SIT tests. For each fault that has less than Δ SIT tests, the appropriate amount of MIT tests are generated using the ROBDD based framework. Each generated candidate MIT test must examined for the FRP property using the ROBDD based framework.

References

[1] D. Bhattacharya, P. Agrawal, D. Agrawal, "Test Generation for Path Delay Faults using Binary Decision Diagrams", *IEEE Trans. on Computers*, vol. 44, pp. 434–447.

[2] K. Brace, R. Ruddel, R.Bryant, "Efficient Implementation of a BDD Package", *in Proc. 1990 Design Automation Conference*, pp.40–45.

[3] R. Bryant, "Graph–based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, Vol.C–35, No.8, August 1986, pp.677–691.

[4] J. Crawfort, *NTAB:Propositional Satisfiability Checker*, CIRL, The University of Oregon, 1996.

[5] S. Devadas, K. Keutzer, S. Malik, A. Wang, "Computation of floating mode delay in combinational circuits: Practice and Implementation", *IEEE Transactions on Computer–Aided–Design*, 12(12):1924–1936, December 1993.

[6] R. Drechsler, "BiTes: A BDD based Test Pattern Generator for Strong Robust Path Delay Faults", *in Proc. 1994 EDTC*, pp. 322–327.

[7] J. P. Marques Silva, K. A. Sakallah, "GRASP–A New Search Algorithm for Satisfiability", *IEEE Transactions on Computers*, 1996, pp.220–227.

[8] P. C. McGeer, A. Saldanha, R. K. Brayton, A. Sangiovanni–Vincentelli, "Delay models and exact timing analysis. In T. Sasao, editor *Logic Synthesis and Optimization*, pages 167–189, Kluwer Academic Publishers, 1993.

[9] I. Pomeranz, S. M. Reddy, "Functional Test Generation for Delay Faults in Combinational Circuits", *in Proc. 1995 Intl. Conf. on Computer–Aided–Design*, Nov. 1995, pp.687–694.

[10] I. Pomeranz, S. M. Reddy, "On Testing Delay Faults in Macro–based Combinational Circuits", *in Proc. 1994 Intl. Conf. on Computer–Aided–Design*, Nov. 1994, pp.332–339.

[11] F. Somenzi, *CUDD:CU Decision Diagram Package*, Release 2.2.0, Department of Electrical and Computer Engineering, The University of Colorado at Boulder, 1998.

[12] S. Tragoudas, M. Michael, "ATPG Tools for Delay Faults at the Functional Level", Technical Report CENG–TR–98–118, ECE Dept., The University of Arizona.

[13] S. Tragoudas, M. Michael, "ATPG Tools for Delay Faults at the Functional Level", Design Automation and Test in Europe Conference (DATE'99), March 1999, to appear.

[14] B. Underwood, W. O. Law, S. Kang, H. Konuk, "Fastpath: A Path-Delay Test Generator for Standard Scan Designs", *in Proc. 1994 Intl. Test Conf.*, Oct. 1994, pp.154–163.

Table 1: SIT ATPG for model M1.

Circuit	Num Flts	ROBDD		SAT	
		Num tests found	Time (hours)	Num tests found	Time (hours)
c432	832	801 (804)	0.03 (0.52)	807	0.06
c499	4329	3915 (3915)	3.42 (10.21)	3917	5.8
c880	1368	1320 (1321)	0.12 (1.21)	1325	0.27
c1908	3224	3148 (3153)	1.65 (3.96)	3159	2.03
c2670	4018	3955 (3956)	1.98 (4.41)	3958	2.65
c3540	2730	2483 (2493)	3.1 (8.45)	2510	4.26
c5315	10718	10655 (10662)	6.52 (16.23)	10675	8.68
c6288	2337	1655 (2085)	9.62 (17.84)	2110	7.41
c7552	13932	13811 (13815)	9.14 (20.88)	13820	11.76

Table 2: MIT ATPG for model M1.

Circuit	Num Flts	ROBDD		SAT	
		Num tests found	Time (hours)	Num tests found	Time (hours)
c432	832	775	0.45	705	4.13
c499	4329	3679	6.49	1824	> 24
c880	1368	1293	1.18	1055	8.39
c1908	3224	2992	2.25	2513	18.21
c2670	4018	3966	2.82	2003	> 24
c3540	2730	2382	5.17	1158	> 24
c5315	10718	10139	8.33	2805	> 24
c6288	2337	1865	8.46	899	> 24
c7552	13932	13263	11.34	2919	> 24

Table 3: SIT ATPG for model M2.

Circuit	ROBDD-based		SAT-based	
	Avg. Δ per fault	Time (hours)	Avg. Δ per fault	Time (hours)
c432	290	0.06	198	0.63
c499	315	4.47	200	7.70
c880	401	0.25	278	1.40
c1908	353	2.24	191	4.33
c2670	489	3.12	275	5.69
c3540	258	4.81	134	7.89
c5315	427	8.29	259	13.11
c6288	231	18.33	156	> 24
c7552	375	11.10	221	17.45

Table 4: ROBDD based MIT ATPG for model M2.

Circuit	Avg. Δ per Flt	Time (hours)
c432	257	0.62
c499	249	8.91
c880	212	1.95
c1908	297	3.54
c2670	431	4.95
c3540	207	7.53
c5315	354	11.40
c6288	192	12.22
c7552	333	14.31