Optimizing Tests for Multiple Fault Models

Nitin Yogi* and Vishwani D. Agrawal
Auburn University, Department of Electrical and Computer Engineering, Auburn, AL 36849, USA
yoginit@auburn.edu, *vagrawal@eng.auburn.edu*

Abstract

We present a method for deriving minimal tests to cover multiple fault models. Integer linear programming (ILP) is used to select a minimum set of vectors without reducing the original coverage. Tests can be initially generated separately for different fault models. All tests are then simulated for each fault model of interest without fault dropping. The fault simulation data are converted into ILP constraints, whose solution guarantees the minimality of test length. As an illustration we have used stuck-at, transition, and I_{DDQ} (pseudo stuck-at) faults. Tests are generated and simulated using Mentor Graphics FastScan and then minimized by the AMPL-CPLEX ILP software. The illustration includes combinational, scan and non-scan sequential circuits. For scan circuits, either type, i.e., launch on shift (LOS) or launch on capture (LOC) can be used. In each case, a minimal set of vectors is obtained in which a small subset of vectors is identified for I_{DDQ} measurement. An interesting aspect of the ILP methods is the trade off they allow between the absolute minimality of test length and the number of I_{DDQ} measurements.

1. Introduction

Several years ago the results of a Sematech study [8] were published. Four types of tests, namely, scan-based stuck-at, scan-based delay, I_{DDQ}, and functional, were examined. A general conclusion was that none of the tests could be dropped. That study has been a subject for numerous discussions [9, 10]. With advances in technology, new fault modes are continuously emerging and the gap between the age-old stuck-at fault model and "realistic defects" continues to widen. New fault models are continuously emerging at recent conferences. On the other hand, the need to minimize test length and test time has never been greater because of complex system-on-chip (SOC) devices. This paper addresses the problem of combining tests that target several fault models into a single compact test.

Test minimization has been a widely researched area. However, most of the published methods will be difficult to apply to multiple fault models. Integer linear programming (ILP) is an effective method of test optimization. It gives global optimization and has been used for both combinational and sequential circuits [1, 13, 14] as well as for globally minimizing N-detect tests [3].

In this paper we develop ILP methods for multiple-fault models. Although applications of ILP have been reported for separately optimizing vectors for detecting stuck-at faults [1, 13, 14], N-detect stuck-at faults [3], and transition faults [15], to our knowledge a simultaneous ILP optimization for multiple fault models has not been attempted before.

2. Overview

Different fault models address different types of realistic defects that can occur at the wafer level. For example, stuck-at faults model some types of transistor short defects and transition delay faults model certain timing defects. None of the two are adequate in modeling each other's characteristics completely as was observed in one of our experiments. In our experiment, a comparison was performed between the transition delay fault coverage of stuck-at fault vectors which detected 100% of detectable stuck-at faults and stuck-at fault coverage of transition delay vectors which detected 100% of detectable transition delay faults for an ISCAS'89 benchmark circuit. In neither of the cases we observed a coverage of 100%. A similar observation was been made in [15]. This characteristic holds true for other fault models too. This necessitates the inclusion of test

*PhD student.

vectors over different fault models to improve the defect coverage. An easy method of achieving this is to concatenate all the test sets together (as was being experimented in [15]), however this rapidly increases the number of test vectors. Hence we need a method to find a minimized test set which caters to all the considered fault models.

In order to address this issue, we make use of the Integer Linear Programming (ILP) problem formulation. An ILP problem is a subset of a Linear Programming (LP) problem. A LP problem [11] is an optimization problem in which we need to minimize or maximize an objective function given a set of constraints. The objective function and constraints are both linear in nature. A LP problem formulation consists of three parts: a set of variables defined as real numbers, a set of constraints on those variables and an objective function consisting of those variables which needs to be minimized or maximized. On solving this problem, the solution consists of values assigned to the variables which satisfy the constraints and meet the objective function goals. An ILP problem is one, in which the variables are defined as integers as opposed to real numbers.

3. Multiple-Fault Model Minimization Problem

Given a set of test vectors, which detects different types of faults (stuck-at, transition delay, I_{DDQ}, etc.), our goal is to find a minimized test set to detect all detected faults of the corresponding fault models. Here we consider combinational and scan-inserted sequential circuits only. Hence the test sets are combinational in nature and their order of application is immaterial.

Basic problem with multiple fault models is that a fault simulator can deal with only one model at a time. We therefore concatenate all tests and then separately simulate the entire set for each fault model, one at a time without fault dropping and obtain a fault dictionary i.e. information about the faults detected by each test vector. Many fault simulators provide this facility. Having obtained this fault dictionary, our aim is to pick the least number of test vectors which cover all the faults in the considered fault models. We formulate this problem as an integer linear programming model. In the following sections we introduce two ILP models for test set minimization. The first is called the "two-step ILP" model and the second the "combined ILP" model.

4. Two-Step ILP Model

As discussed earlier, our goal is to minimize a test set for multiple fault models. Different fault models have different characteristics and procedures for testing. For example, if test vectors are generated for I_{DDQ} faults, just the application of the vectors is not enough, but an extra step of I_{DDQ} measurement needs to be carried out. Since this I_{DDQ} measurement is an expensive and time consuming test, it is not performed for all vectors unlike the fault coverage measurement for stuck-at and transition delay test vectors. Hence the I_{DDQ} vectors and their corresponding measurements need to be given special attention. Based on this constraint, we propose a two-step ILP model. In the first step a global minimization for the total number of vectors is carried out for all the considered fault models and then in the second step the minimum number of I_{DDQ} measurements to be made are determined.

4.1 First ILP - Minimize Vectors

As described in the previous section we obtain the fault dictionary by concatenating all tests and then separately simulating the entire set for each fault model, one at a time without fault dropping. Using the fault dictionary information, for each fault (of any type), one constraint inequality is generated. Suppose, fault f_i is detected by test numbers j, m and q, then the corresponding constraint is,

$$t_j + t_m + t_q \geq 1 \tag{1}$$

where t_x is an integer variable for test number x that can take the value 0 (test number x is discarded) or 1 (test number x is selected). The above constraint means that at least one test is required to ensure the coverage of fault f_i. Because of this constraint, any test set solution that ILP provides will never drop any fault and the fault coverage is ensured. Notice that a test here can be a single vector (in our case for stuck-at faults and Iddq faults) or a vector pair (for transition delay faults).

A single constraint like inequality (1) is generated for each fault in the fault set, which contains all faults corresponding to all fault models of interest. Our objective function for this problem is defined as follows:

$$\text{Minimize} \sum_{all\ vectors\ x} t_x \tag{2}$$

The ILP then finds a solution under the specified constraints (as shown in equation (1)) and meeting the objective function (shown in equation (2)). The solution consists of values assigned to variable t_x (x for all vectors) which can be either 0 (test number x is discarded) or 1 (test number x is selected). The test sets with $t_x = 1$ are minimal to guarantee detection of each modeled fault by at least one vector.

4.2 Second ILP - Minimize I_{DDQ} Measurements

Once vectors are minimized, a second ILP run is used to find the minimal set of these vectors on which I_{DDQ} should be measured. For this ILP run, the tests consist of only those selected by the first ILP run. All t_x's whose values were 0 in the first ILP run are eliminated. Constraints like inequalities(1) are then specified only for the I_{DDQ} faults. The objective function to minimize is:

$$\text{Minimize} \sum_{selected\ vectors\ y} t_y \tag{3}$$

The non-zero t_y's in this ILP solution identify the minimal set of vectors for which I_{DDQ} needs to be measured.

5. Combined ILP Model

Since I_{DDQ} measurement is expensive, it is necessary that their number be kept to a minimum, sometimes even at the expense of slight increase in the total number of vectors to be applied. The "Two-step ILP" model discussed in the earlier section does not give the global minimum number of I_{DDQ} measurements as in it's first step itself it optimizes for the total number of vectors. To achieve the goal of reducing the I_{DDQ} measurements even further, we modify the ILP formulation of Section 4.

We represent each vector x in the initial vector set by two {0,1} integer variables, t_x and an I_{DDQ} variable i_x. The variable t_x as before signifies whether test number x will be selected in the final minimized test set (either as an I_{DDQ} test or a non-I_{DDQ} test). The variable i_x is used to signify whether an I_{DDQ} measurement is performed on test number x. Also this means that variable i_x needs to be a subset of variable t_x.

The ILP constraints are generated as follows. If fault f_i is not an I_{DDQ} fault, then its constraint is exactly like inequality 1, i.e., no constraints are applied to the i variables of its tests. For an I_{DDQ} fault f_k, that is detectable by vectors u, v and w, the constraint inequalities are as follows:

$$t_u + t_v + t_w \geq 1 \tag{4}$$

$$i_u + i_v + i_w \geq 1 \tag{5}$$

$$t_u \geq i_u \tag{6}$$

$$t_v \geq i_v \tag{7}$$

$$t_w \geq i_w \tag{8}$$

The last three constraints mean that I_{DDQ} measurements may be conducted for some subset of all vectors with capability of detecting I_{DDQ} faults. The objective function is,

$$\text{Minimize} \sum_{all\ vectors\ x} (\ t_x +\ W \times i_x) \tag{9}$$

Here the weighting factor W specifies how strongly we wish to minimize the number of I_{DDQ} measurements. Setting $W = 0$ will lead to the result of Section 4.1, i.e., we will minimize the vector count without any regard for I_{DDQ} measurements. A second ILP run with objective function 3 may give more than the smallest possible number of I_{DDQ} measurements. Choosing a larger value of W will minimize the number of I_{DDQ} measurements but may lead to slightly increased number of vectors. This trade off is evident in the results explained in Section 7.

Though the ILP gives an optimum solution, it's time complexity is exponential and can become a bottleneck. Our results in Section 7 exhibit this characteristic. Hence we need to adopt a technique which will simplify the computation. A linear programming (LP) model is often used in place of integer linear programming (ILP) model to reduce the CPU time and obtain a result that may still be quite close to the optimum. The worst-case complexity of LP is polynomial-time while that of ILP is exponential. We have devised an "hybrid LP-ILP" method, outlined in Section 6, which is a variation of the published recursive-LP method for test minimization [4].

6. Hybrid LP-ILP Method

Observing the large time-complexity for some ILP runs, we developed an hybrid LP-ILP method. This method solves any ILP problem and can be effectively used for the models introduced in Sections 4 and 5. In

Table 1. Test vectors for stuck-at, I_{DDQ} and transition faults generated and minimized by FastScan.

Circuit (1)	Type of vectors (2)	Number of vectors		Fault coverage (%) (5)
		Unminimized (3)	Minimized (4)	
c1908	stuck-at	66	46	93.93
	I_{DDQ}	37	26	98.19
	transition	113	75	91.80
	total	216	147	-
c3540	stuck-at	167	130	96.00
	I_{DDQ}	53	45	99.09
	transition	299	229	96.55
	total	519	404	-
s1238	stuck-at	178	149	95.17
	I_{DDQ}	70	58	93.04
	LOS	293	234	96.75
	LOC	296	242	95.97
	total	837	683	-
s1488	stuck-at	136	114	100.00
	I_{DDQ}	55	45	98.20
	LOS	166	126	67.92
	LOC	200	166	91.57
	total	557	451	-
s5378	stuck-at	150	145	99.30
	I_{DDQ}	71	70	85.75
	LOS	319	293	98.31
	LOC	256	242	90.05
	total	796	750	-

LOS : Launch-on-Shift; LOC : Launch-on-Capture

spite of some similarities other methods [4], the hybrid LP-ILP differs from previous work.

Once formulated as an ILP, an LP model is obtained by changing the variables to real numbers in the same [0,1] range. The LP is solved and variables below 0.1 are then rounded to 0 and variables above 0.9 are rounded to 1. The rounded variables are now treated as constants, constraints are updated and the LP solution and rounding processes are repeated until no more variables can be rounded either to a 0 or a 1. Then a much reduced ILP is solved. The results in Section 7 reveal the reduced time complexity of this method.

Since we round variables on both sides, i.e., to 0 and to 1, the procedure can lead to an intermediate solution that may not satisfy all remaining constraints. In such a case, the problem can be solved by the previously proposed recursive-LP algorithm [4], which guarantees a solution. We believe that, whenever a solution is possible, the hybrid LP-ILP will be faster than the recursive-LP.

7. Results

We applied our optimization techniques to two combinational and three scan inserted sequential circuits. We used three different fault models. They were stuck-at, transition and I_{DDQ} faults. We generated initial test vectors for each fault model using the commercial ATPG tool, Mentor Graphics FastScan [7]. For I_{DDQ} faults, the pseudo stuck-at model [6] was used because it is supported in FastScan.

Table 1 gives the details of the test vectors generated for the circuits. Column (1) gives the circuit name. Column (2) specifies the fault model for which test vectors in column (3) were generated. Stuck-at and I_{DDQ}

Table 2. Multiple fault model test optimization by ILP methods.

Circuit	Type of vectors	FastScan tests		Two-step model		Combined model					
		Original no. of vectors	Optimized no. of vectors	No. of vect.	CPU sec.	$W = 0.1$		$W = 1$		$W = 10$	
						No. of vect.	CPU sec.	No. of vect.	CPU sec.	No. of vect.	CPU sec.
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
c1908	All	216	147	79	8.97	79	9.3	81	9.4	84	9.9
	I_{DDQ}	37	26	24	1.50	24		22		20	
c3540	All	519	404	225	41.35	225	5044.2*	226	5046.7*	247	5046.9*
	I_{DDQ}	53	45	39	694.41	40		41		37	
s1238	All	837	683	203	20.59	203	97.1	205	63.5	212	243.1
	I_{DDQ}	70	58	51	1.78	45		43		40	
s1488	All	557	451	175	17.11	175	76.0	176	254.0	187	694.9
	I_{DDQ}	55	45	40	2.37	39		38		33	
s5378	All	796	750	320	147.75	320	2313.9	326	5154.2*	353	5160.9*
	I_{DDQ}	71	70	84	14.21	78		73		64	

* Incomplete optimization; CPU time limit of 5000 seconds exceeded.

vectors were single pattern tests and transition delay vectors were two-pattern tests. Each of those vector-pairs was considered a single entity in the ILP model. For combinational circuits, the transition delay tests were simple two patterns tests applied one after another, however, for scan-inserted sequential circuits we obtained launch-on-shift (LOS) and launch-on-capture (LOC) patterns [5, 12]. Column (3) gives the number of vectors that FastScan generated for each type of test vector set and column (4) gives the number of vectors obtained after minimization without reducing the fault coverage using tools provided in FastScan. Column (5) gives the fault coverage of each test set for the corresponding fault model. The numbers in the "total" rows are the sum of the three previous rows. Further minimization of these by repeated fault simulation was not attempted.

The results of minimization over multiple fault models using the ILP methods are given in Table 2. For comparison, the results of single fault model minimization from the "total" rows and columns (3) and (4) of Table 1 are reproduced in columns (3) and (4) of Table 2 in rows marked as "All".

We used the test vectors generated by FastScan (reported in column (3) of Table 2) as our initial test set. We simulated the combined vector set of the three fault models without fault dropping and obtained the fault dictionary information giving the complete fault detection data for all vectors. This fault dictionary was then used in the formulations of ILP as mentioned in previous sections.

For our experiments we used the Sun Sparc Ultra-10 machine with 4.0 GB of RAM shared among four CPUs. We used AMPL-CPLEX package for ILP [2]. While executing AMPL we specified a CPU time limit of 5000 seconds. The results for various circuits and their vector sets are tabulated in Table 2. Vectors for each circuit are shown in two rows. FastScan separately generated vectors for each fault model. I_{DDQ} vectors and "All" vectors (including I_{DDQ} vectors) are shown in column (3). Column (4) shows the sum of numbers of vectors obtained by using the minimization tools of FastScan separately for each vector set with its corresponding fault model.

Columns (5) and (6) show the results of the two-step ILP model of Section 4 in terms of the number of vectors and the CPU time. This shows a reduction in I_{DDQ} measurements but as pointed out before, further reduction is possible. Columns (7) through (12) show the results of ILP of Section 5, with increasing emphasis on I_{DDQ} by setting $W = 0.1$, 1 and 10, respectively, in the objective function 9. The trade off between the number of I_{DDQ} measurements and test length is clearly observed.

Reducing Run Times: The results of minimization by the hybrid LP-ILP method are given in Table 3. The columns of the table have the same meaning as th0se in Table 2. The number of minimized vectors and the CPU time complexity can easily be compared between these tables. For the two-step model, the pure

Table 3. Multiple fault model test optimization by hybrid LP-ILP method.

Circuit	Type of vectors	FastScan tests		Two-step model		Combined model					
		Original no. of vectors	Optimized no. of vectors	No. of vect.	CPU sec.	$W = 0.1$		$W = 1$		$W = 10$	
						No. of vect.	CPU sec.	No. of vect.	CPU sec.	No. of vect.	CPU sec.
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
c1908	All	216	147	79	17.50	79	17.88	81	23.74	84	23.23
	I_{DDQ}	37	26	24	2.32	24		23		20	
c3540	All	519	404	225	78.13	225	166.6	226	188.6	248	516.23
	I_{DDQ}	53	45	40	43.92	41		39		34	
s1238	All	837	683	203	37.91	203	48.36	203	61.64	215	63.12
	I_{DDQ}	70	58	51	3.4	46		46		40	
s1488	All	557	451	175	33.91	175	38.33	176	50.33	187	59.99
	I_{DDQ}	55	45	40	5.51	39		39		34	
s5378	All	796	750	320	306.43	320	528.89	326	617.14	353	793.22
	I_{DDQ}	71	70	84	32.63	80		72		63	

ILP method performed slightly better in terms of both; the number of minimized vectors and the CPU time, as compared to the hybrid LP-ILP method. However for the combined model, the hybrid LP-ILP method achieves close to optimum solution in terms of the number of minimized vectors, with an order of magnitude reduction in CPU time as compared to the pure ILP method. In cases of circuits like c3540 and s5378 where ILP quit early because of exceeding the CPU time limit, the hybrid LP-ILP method gave better results. For some circuits like s1238 and s1488, the hybrid LP-ILP method gave a close to optimum solution, if not the optimum.

When the hybrid LP-ILP method is used, the optimality of the solution achieved can be judged by examining the value of the objective function of equation 9 obtained in the first run of the LP. This value (usually rounded to the next higher integer) gives a lower bound on the absolute optimum solution as would be given by the ILP. Table 4 gives a comparison between this lower bound number, pure ILP method and hybrid LP-ILP method for the combined model. Column (1) gives the circuit name and column (2) gives the different cases of weights in the combined model. Columns (3) through (5) give the lower bound and the results of the exact ILP and hybrid LP-ILP methods. In most of the cases we observe that the value of the result obtained by the hybrid LP-ILP method either equals or is close to the exact optimum obtained by pure LP and this fact is indicated by the closeness of the hybrid LP-ILP solution to the lower bound. The pure ILP gives non-optimum solution in some cases where it had to quit due to CPU time limit. In those cases the hybrid LP-ILP method performs better.

8. Conclusion

The results show an effective reduction in both the total number of vectors and the number of I_{DDQ} measurements required as compared to the original test set. For test set minimization with multiple fault models, the problem arises because fault simulators lack the capability of simultaneously simulating more than one type of fault. The ILP technique allows us to combine the results of separate fault simulations. Besides, ILP gives a global optimization of the test set. The ILP methods also provide useful trade offs by varying the emphasis between total test length and the number of I_{DDQ} measurements. Using a hybrid LP-ILP method to solve an ILP problem significantly reduces the time complexity, while achieving a close to optimum solution.

References

[1] P. Drineas and Y. Makris, "Independent Test Sequence Compaction through Integer Programming," in *Proc. Int. Conf. Computer Design*, 2003, pp. 380–386.

[2] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming.* South San Francisco, California: The Scientific Press, 1993.

Table 4. Comparing solutions: hybrid LP-ILP lower bound, ILP optimum and hybrid LP-ILP.

Circuit	Weight W	minimized (vectors + $W \times I_{DDQ}$ measurements)		
		hybrid LP-ILP lower bound	ILP solution	hybrid LP-ILP solution
c1908	0.1	81.4	81.4	81.4
	1	102.25	103	104
	10	276.5	284	284
c3540	0.1	227.94	229*	229.1
	1	257.82	267*	265
	10	499.97	617*	588
s1238	0.1	207.47	207.5	207.6
	1	247.64	248	249
	10	606.04	612	615
s1488	0.1	178	178.9	178.9
	1	211.74	214	215
	10	506.89	517	527
s5378	0.1	326.76	327.8	328
	1	392.28	399*	398
	10	910.68	993*	983

* Incomplete optimization; CPU time limit of 5000 seconds exceeded.

[3] K. R. Kantipudi and V. D. Agrawal, "On the Size and Generation of Minimal N-Detection Tests," in *Proc. 19th Int. Conf. VLSI Design*, Jan. 2006, pp. 425–430.

[4] K. R. Kantipudi and V. D. Agrawal, "A Reduced Complexity Algorithm for Minimizing N-Detect Tests," in *Proc. 20th International Conf. VLSI Design*, Jan. 2007, pp. 492–497.

[5] R. Madge, B. R. Benware, and W. R. Daasch, "Obtaining High Defect Coverage for Frequency-Dependent Defects in Complex ASICs," *IEEE Design & Test of Computers*, vol. 20, no. 5, pp. 46–53, Sept.-Oct. 2003.

[6] P. C. Maxwell and R. C. Aitken, "I_{DDQ} Testing as a Component of a Test Suite: The Need for Several Fault Coverage Metrics," *Journal Electronic Testing: Theory and Applications*, vol. 3, no. 4, pp. 305–316, Dec. 1992.

[7] Mentor Graphics, *FastScan and FlexTest Reference Manual*, 2004.

[8] P. Nigh, W. Needham, K. Butler, P. Maxwell, and R. Aitken, "An Experimental Study Comparing the Relative Effectiveness of Functional, Scan, I_{DDQ}, and Delay-Fault Testing," in *Proc. 15th IEEE VLSI Test Symp.*, April-May 1997, pp. 459–464.

[9] P. Nigh, W. Needham, K. Butler, P. Maxwell, R. Aitken, and W. Maly, "So What Is an Optimal Test Mix? A Discussion of the SEMATECH Methods Experiment," in *Proc. Int. Test Conf.*, Nov. 1997, pp. 1037–1038.

[10] P. Nigh, D. Vallett, A. Patel, J. Wright, F. Motika, D. Forlenza, R. Kurtulik, and W. Chong, "Failure Analysis of Timing and IDDq-only Failures from the SEMATECH Test Methods Experiment," in *Proc. Int. Test Conf.*, Sept. 1999, pp. 1152–1161.

[11] J. Noyes and E. W. Weisstein. Linear Programming. From MathWorld–A Wolfram Web Resource. *http://mathworld.wolfram.com/LinearProgramming.html*.

[12] L. Xijiiang, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson, and N. Tamarapalli, "High-frequency, at-speed scan testing," *IEEE Design & Test of Computers*, vol. 20, no. 5, pp. 17–25, Sept-Oct. 2003.

[13] N. Yogi and V. D. Agrawal, "High-Level Test Generation for Gate-Level Fault Coverage," in *Proc. 15th IEEE North Atlantic Test Workshop*, May 2006, pp. 65–74.

[14] N. Yogi and V. D. Agrawal, "Spectral RTL Test Generation for Microprocessors," in *Proc. 20th International Conf. VLSI Design*, Jan. 2007, pp. 473–478.

[15] N. Yogi and V. D. Agrawal, "Transition Delay Fault Testing of Microprocessors by Spectral Method," in *Proc. 39th Southwestern Symp. on System Theory*, Mar. 2007, pp. 283–287.