A Reduced Complexity Algorithm for Minimizing *N*-Detect Tests

Kalyana R. Kantipudi and Vishwani D. Agrawal
Auburn University
Department of Electrical and Computer Engineering
Auburn, AL 36849, USA
kantikr@auburn.edu, vagrawal@eng.auburn.edu

Abstract – *We give a new recursive rounding linear programming (LP) solution to the problem of N-detect test minimzation. This is a polynomial-time solution that closely approximates the exact but NP-hard integer linear programming (ILP) solution. In ILP, a test is represented by a* $[0, 1]$ *integer variable and the sum of those variables is minimized. Constraints ensure that each fault has at least N tests with non-zero variables. Traditionally, the problem has been transformed to less complex LP by treating the variables as real numbers, regarded as probabilities with which they can be rounded off to* 0 *or* 1*. This is known as the randomized rounding method. In the new method, the LP is recursively used, each time rounding the largest variable to 1 and reducing the size of the LP. The method is found to converge to a solution in just a few LP runs and the result is usually better than that of randomized rounding. Experimental results include ISCAS85 benchmarks and a set of multiplier circuits. N-detect tests for* $N = 1,\ 5$ *and* 15 *are considered. Also, a* 10*-vector single-detect sequence for c6288 is given.*

1. Introduction

N-detect tests are stuck-at fault tests such that each fault is detected by at least N different test vectors. N-detect tests are of interest because they are found to be useful in improving the defect coverage [2, 3, 8, 18, 22]. Therefore, various test generation strategies have been developed [2, 14, 16, 19, 22] to derive and apply such tests. The main disadvantage of the N-detect tests is their size. Various optimization strategies based on the ideas of reverse order N-detection fault simulation [19] and integer linear programming [14] have been proposed. Basically, we must reduce the test application time by reducing the test set size.

2. Background

Single-detect test sets are widely used and favored in the industry. The closeness of stuck-at faults to actual defects, the ease of generating tests for those faults and the availability of efficient test generation strategies highly favor the stuck-at tests. The recent demands for improved defect coverage are based on experiments showing that the single-detect test sets can only achieve up to 360-1000 dpm [18]. N-detect tests are suggested as an alternative to single-detect tests because of their ability to improve the defect coverage and their ability to be easily accommodated into the normal test generation flow. N-detect tests can be generated using a normal single detection ATPG [2, 3, 12, 19, 22].

It is observed that detecting a fault multiple times under different excitation and observation conditions increases the probability of detecting the physical defects associated with that fault site [8]. The cost will be increased test set size. The improvement in the defect coverage of N-detect tests is often attributed to the increase in node-to-node bridge fault coverage [3]. Among the defect coverage metrics are excitation balance [5] and logic-proximity bridge coverage [21]. Though there are other types of tests with known defect coverage qualities [9], N-detects have gained acceptance in manufacturing environments and their minimization is a relevant problem.

2.1. *N*-Detect Test Minimization by Integer Linear Programming (ILP) – Previous Work

Suppose we have a set of k vectors that detects every fault at least N times. We use diagnostic fault simulation, i.e., fault simulation without fault dropping, to identify the vector subset T_j that detects a fault j, for all j. We assign an integer-valued variable $t_i \epsilon [0, 1]$ to ith vector such that $t_i = 1$ means that ith vector should be included in the minimal vector set and $t_i = 0$ means that ith vector should be discarded. The problem of finding the minimal N-detect set then reduces to assigning values to t_i's so as to [14]:

$$minimize \sum_{i=1}^{k} t_i \tag{1}$$

under following constraints:

$$\sum_{t_i \epsilon \{T_j\}} t_i \geq N_j, \forall \ faults \ j \tag{2}$$

where N_j is the multiplicity of detection for the jth fault. In general, N_j can be selected for individual faults based on some criticality criteria or on the capability of the initial vector set. For simplicity, we have assumed all N_j's to be equal. An integer linear program (ILP) solver [7] can then find the [0,1] values of the variables $\{t_i\}$ that define a minimal N-detect vector set. For $N = 1$, the ILP produces the minimal single-detect test set [6, 11, 17]. Integer programming has also been applied to sequential circuit test minimization [4].

Following theorems characterize the ILP method:

Theorem 1 [1]: *The size of the largest clique in the independence graph is a lower bound on the single-detect test set size.*

Here the independence graph is defined by faults as nodes and independence of two faults by an undirected edge between the corresponding nodes. Independence of two faults simply means that they cannot be detected by the same test.

Theorem 2 [14]: *A lower bound on the size of the N-detect test set is N times the size of the largest clique in the independence graph.*

Theorem 3 [14]: *When the minimization is performed over an exhaustive set of vectors of a combinational circuit, any ILP solution that satisfies expressions (1) and (2) is a minimum N-detect test.*

3. Linear Programming (LP) Methods

The complexity of the integer linear programming (ILP) is known to be exponential in terms of the number of variables. We illustrate this by a very simple three-vector three-fault (3V3F) example. Consider three faults, f_1, f_2 and f_3, and three vectors. We assign a binary integer t_i to vector i. The single-detection problem is specified as follows:

$$\text{Minimize} \quad t_1 + t_2 + t_3 \tag{3}$$

subject to constraints,

$$\begin{array}{lrcl} f_1: & t_1 + t_2 & \geq & 1 \\ f_2: & t_2 + t_3 & \geq & 1 \\ f_3: & t_3 + t_1 & \geq & 1 \end{array} \tag{4}$$

The solution space is shown in Figure 1. Any pair of vectors is an optimum solution for this problem and the three solutions are shown by black dots. Four points, marked as 0 or 1, do not satisfy the constraints 4 and one point, $t_1 = t_2 = t_3 = 1$, though it satisfies constraints is non-optimal. The

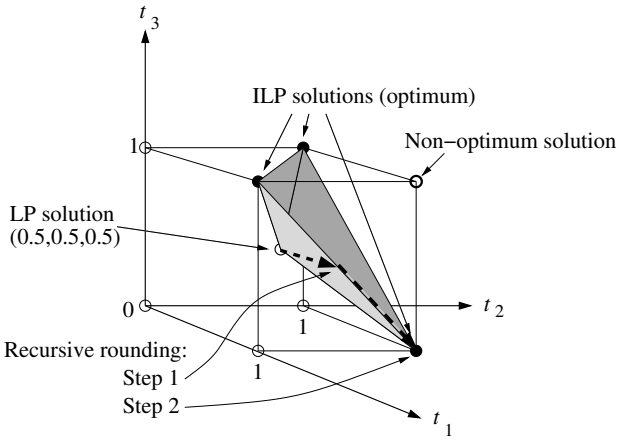

Figure 1. ILP and LP solution space for the three vector three fault (3V3F) example.

ILP basically must search among all vertices of the unit cube (a unit hypercube in general) to find one of the optimal solutions. Although a branch and bound solution can be implemented, the ILP search complexity remains exponential, as the number of vertices for n vectors is 2^n.

3.1. Randomized Rounding LP Solution

As observed by others [4], the above problem can be converted into a linear programming (LP) problem if we redefine the variables t's as real variables in the range [0.0,1.0]. The LP solution, which can be found in polynomial time and sometimes in linear time, lies in the interior of the hypercube. For the 3V3F example, the LP solution $t_1 = t_2 = t_3 = 0.5$ is shown in Figure 1. The problem now remains to convert it into an ILP solution.

Notice that the LP solution for the sum in Equation 3 is 1.5. This is known to be a lower bound for the exact ILP solution, which is 2 in this case. The literature [4, 20] gives a randomized rounding method. The real variables t's are treated as probabilities. We generate a random number x_i uniformly distributed over the range [0.0,1.0] for each variable t_i. If $t_i \geq x_i$ then t_i is rounded to 1, otherwise it is rounded to 0. If the rounded variables satisfy the constraints 4 then the rounded solution is accepted, otherwise, rounding is again performed starting from the original LP solution.

For many problems, randomized rounding works efficiently. However, notice that for our 3V3F problem, $t_1 = t_2 = t_3 = 0.5$, and therefore, all nodes in Figure 1 are equally likely. Here, the randomized rounding is nothing but a random search. In general, we found that the LP solution for the test minimization problem contains a large number of equal values. That makes the search almost random. Since the number of tests is much larger than

the size of the minimal test set, we find ourselves conducting a random search in a very large solution space, which contains very few optimal and many non-optimal solutions (this last point is not illustrated by the small 3V3F example). As a result, it may require many iterations of rounding before all constraints are satisfied and even then the solution generally turns out to be non-optimal.

3.2. Recursive Rounding (Present Contribution)

After unsuccessful attempts at using randomized rounding for a solution for benchmark circuits, we devised a new *recursive rounding* procedure:

1. Obtain an LP solution. Stop if each t_i is either 0.0 or 1.0.
2. Round the largest t_i to 1 and fix its value to 1.0. If several t_i have the largest value, then arbitrarily set only one to 1.0. Go to Step 1.

Step 1 guarantees that any solution thus obtained satisfies all constraints. The maximum number of LP runs in Step 1 is bounded by the minimized test set size because each iteration selects at least one vector. Of course, an absolute optimality is not guaranteed.

For the 3V3F example, Step 1 gives $t_1 = t_2 = t_3 = 0.5$. In Step 2, we arbitrarily set $t_1 = 1.0$. Thus, the first and third constraints in 4 are satisfied. Repeating Step 1, we get either $t_2 = 1$, $t_3 = 0$ or $t_2 = 0$, $t_3 = 1$ or $t_2 = t_3 = 0.5$. In the last case, Step 2 sets $t_2 = 1.0$ and then Step 1 gives $t_3 = 0.0$. Thus, we always select two vectors, which is an optimum solution.

To understand recursive rounding, let us reexamine the 3V3F example in Figure 1. The LP solution and the optimal ILP solutions form a tetrahedron. Having found the apex (LP solution), which is an interior point, we wish to get to any one of the points at the base of the tetrahedron. The recursive rounding procedure involves successively projecting onto one of the faces (by setting a variable to 1.0) and thereby reducing the dimension of the solution space by one each time. When the process terminates, the LP solution is found at a corner of the reduced dimension hypercube. These LP solutions are shown by the bold dashed line arrows in Figure 1.

This small example clearly shows the effectiveness of recursive rounding, whose key feature is the guaranteed convergence to a solution in a small number (size of the minimal test set) LP runs. It is observed that in each run, many 1's are generated by the LP, thereby reducing the number of LP runs as well as the sizes of the repeated LP runs. For most large circuits the total time was dominated by the time taken by the first LP run. Although this example is too small to illustrate the limitations of the method, in practice we have found it to work well. In most cases, our ability to find the minimum test is restricted by Theorem 3, which says that an absolute minimality is guaranteed only if we start with the exhaustive vector set. The following result is easily derived:

Table 1. Optimized single-detect tests for ISCAS85 circuits.

Circ. name	Initial vect.	Lower bound	Rand round Vect.	Rand round CPU s	Rec. round Vect.	Rec. round CPU s	ILP Vect.	ILP CPU s
c432	608	34.7	66	1	36	2	35	4
c499	379	52.0	52	1	52	1	52	4
c880	1023	23.4	124	8	28	31	28	31h*
c1355	755	84.0	84	5	84	5	84	14
c1908	1055	106.0	109	7	107	8	107	29
c2670	959	84.0	96	9	84	9	84	49
c3540	1971	89.9	301	64	105	197	108	70m*
c5315	1079	62.7	223	121	72	464	74	70m*
c6288	243	10.2	92	33	18	78	18	6h*
c7552	2165	144.1	231	145	145	151	145	8035

* Incomplete run

$$\text{LP lower bound} \leq \text{ILP size} \leq \text{Recur. LP size} \quad (5)$$

Because ILP size is optimum, whenever the recursive LP test set size equals the LP lower bound, we have the optimum solution. Otherwise, the difference between the recursive solution and the lower bound provides the maximum possible deviation from optimality.

4. Results

4.1. Single-Detect Tests

Our first results evaluate the relative merits of recursive rounding against randomized rounding [4, 20] and ILP. Table 1 gives optimized test set sizes for single-detection. The initial vectors in the second column were obtained from an ATPG program [15]. Enough vectors were generated so that every fault was covered by at least five vectors. This required between 5 to 8 complete test sets from the ATPG. The industrial methods mentioned in earlier papers [2, 3, 12, 22] can also be used to generate a similar test set for further minimization.

As stated before, the minimum value of the sum of variables provided by the LP is a lower bound (sometimes unattainable) on the size of the absolute minimum test set. This is given in the third column of Table 1. The next six columns give optimized test set sizes and CPU times (Sun Ultra-5) for randomized rounding, recursive rounding, and ILP, respectively. In some cases, ILP runs did not complete.

Table 2. Single-detect test optimization for multipliers.

Mult. size	Initial vect.	Lower bound	Rec. round. Vect.	Rec. round. CPU s	ILP (*incomplete) Vect.	ILP (*incomplete) CPU s
3	64	5.5	6	0.13	6	0.25
4	256	6.0	7	3	6	6
5	1024	6.0	8	10	7	56
6	1024	6.1	8	18	8	9704
8	1024	6.4	10	45	10	1007*
10	1024	6.9	12	112	12	1011*
12	1024	8.3	14	173	17	1016*
14	1024	8.2	14	428	15	1022*
16	1024	8.4	16	742	*	*

Those are shown by asterisk (*) in the last column. A simpler form of randomized rounding as described by Hochbaum [11] was implemented. We notice that recursive rounding solutions are almost the same as ILP. For c3540 and c5315, ILP solution was suboptimal because the program did not complete. Recursive rounding found better solutions. Randomized rounding was suboptimal in several cases although its CPU time was always the smallest. As stated in the previous section, the recursive LP may iterate in the worst case as many times as the size of the optimized vectors. However, the CPU times in columns 5 and 7 show that not to be the case, considering that randomized rounding does not iterate.

To study the complexity of the recursive rounding we used array multipliers of increasing sizes. As shown in Table 2 the initial vector sets were exhaustive for multipliers up to five bits. For larger multipliers 1,024 random vectors that could detect all faults were used. ILP could not complete in several cases and its test sets above 12-bits were larger than those obtained by recursive rounding. Figure 2 shows the time complexities and the minimized test set sizes. The exponential complexity of ILP is evident. CPU time limits had to be used and the ILP solutions became worse than those of recursive rounding.

4.2. *N*-Detect Tests

Table 3 gives optimized 5-detect test set sizes for ISCAS85 benchmarls. The initial vectors in the second column are the same as those used in the previous subsection for single-detection. Here again, we had problems with excessive CPU times for the ILP. The time complexity of recursive rounding is much lower. The test set sizes are either equal to those of ILP (optimum) or are very close. The usefulness of determining how close the recursive solution by the relation (5) is to the optimun can be verified. For c7552, even without knowing the ILP solution, we can say that the recursive rounding solution is optimal. For c6288, the lower bound is 53 (rounded

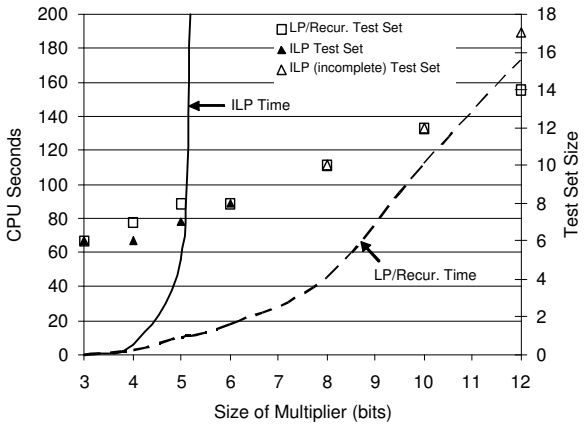

Figure 2. Quality and Complexity of recursive LP and ILP solutions for multipliers.

Table 3. Optimized 5-detect tests for ISCAS85 circuits.

Circuit name	Initial vect.	Lower bound	LP/rec. round. Vect.	LP/rec. round. CPU s	ILP (optimum) Vect.	ILP (optimum) CPU s
c432	608	196.4	197	1.0	197	1.0
c499	379	260.0	260	1.2	260	2.3
c880	1023	126.0	128	14.0	127	881.8
c1355	755	420.0	420	3.2	420	4.4
c1908	1055	543.0	543	4.6	543	6.9
c2670	959	477.0	477	4.7	477	7.2
c3540	1971	467.3	477	72.0	471	20008.5
c5315	1079	374.3	377	18.0	376	40.7
c6288	243	52.5	57	39.0	56	56000.0
c7552	2165	841.0	841	52.0	841	114.3

to integer) and the recursive rounding solution of 57, according to (5), is within 4 vectors from the optimum. As is evident from columns 4 and 6, this deviation from the optimum does not diverge when the circuits become larger.

Table 4 gives the result of 15-detect tests. Although the unoptimized vectors are generated, same as before, by Atalanta [15], in some cases we required many more vectors to have at least 15 detections for every fault. These numbers are given in column 2. Many ILP solutions (c880, c3540, c5315 and c6288) were obtained by setting time limits, while recursive rounding LP always converged, giving test set sizes within one vector of ILP. For c3540 it was better than the time-limited ILP. The last three columns compare the 15-detect results from the literature [16]. The lower bound (L.B.) is simply a number that is 15 times the minimum reported for single-detection [10].

CPU times (ILP and [16]) are for Sun Ultra-5, except those with † (Ultra-10) and ‡ (Sun Fire-280R-900MHz-Dual-Processor). Once again we see that the new LP/recursive rounding result is extremely close to the optimum (ILP) and its time

Table 4. Optimized sizes of 15-detect tests for ISCAS85 benchmark circuits.

Circuit name	Initial vectors	Lower bound	LP/recur. rounding (this paper)		ILP [14]		Heuristic [16]		L.B. [10]
			Vectors	CPU s	Vectors	CPU s	Vectors	CPU s	
c432	14882	429.5	430	83.5	430	444.8	505	292.1	405
c499	1850	780.0	780	17.8	780	24.9	793	153.2	780
c880	4976	318.9	322	94.5	321	521.4	338	229.6	195
c1355	2341	1260.0	1260	41.2	1260	52.1	1274	5674.6	1260
c1908	6609	1590.0	1590	150.4	1590	191.0	1648	1563.9	1590
c2670	8767	1248.0	1248	380.6	1248	607.8†	962	9357.6	660
c3540	4782	1400.5	1407	239.6	1411	1223.7	-	-	1200
c5315	4318	921.9	924	494.3	924	1368.4†	-	-	555
c6288	731	130.1	134	250.5	134	1206.3	144	1813.8	90
c7552	6995	2370.0	2371	359.1	2370	346.1‡	-	-	975

does not increase as rapidly with the increasing size of the circuit.

5. Finding Minimal Tests for c6288

In this section, we give some minimal single-detect test results on the ISCAS85 benchmark c6288. This circuit is of interest because there exists a huge difference between its theoretical lower bound of six and its practically achieved test set of size 12 [10]. As shown in Figure 3, c6288 is a 15×16 matrix of full-adders (FA) and half-adders (HA). There is prior work on finding minimum test sets of regular array structures that deals with minimization of test sets of ripple carry adders using the minimum test sets of their building blocks [13]. Initially, the minimum test sets for the 1-bit adders, used to build the ripple carry adder, are derived. As the carry-out of an adder passes on as the carry-in of the next adder, the test sets are replicated such that carry-out output bit of the adder is matched to the carry-in input bit of the next adder. We observe that faults of the ith cell propagated to its sum output are immediately detected. Faults propagated to the carry output are detected at the $i+1$ sum output irrespective of the input states of that block. Unfortunately, c6288 is not a completely regular as the ripple carry adder. So, we partitioned the circuit into regular modules, found a minimum test set for a module and tried to replicate it to get the entire vector set for the circuit. Unlike the ripple-carry adders, the outputs of the modules which are fed as inputs to the other modules are not able to propagate to their outputs. So we formulated an experiment in which lower order multipliers of the same architecture (Figure 3) are used along with the linear programming techniques to find minimum test sets.

Using exhaustive vector sets, the minimal single-detect required 6 vectors for four-bit multiplier and 7 vectors for six-bit multiplier. Two sets were generated for the four-bit circuit. These test sets are carefully duplicated to create nearly 900 different

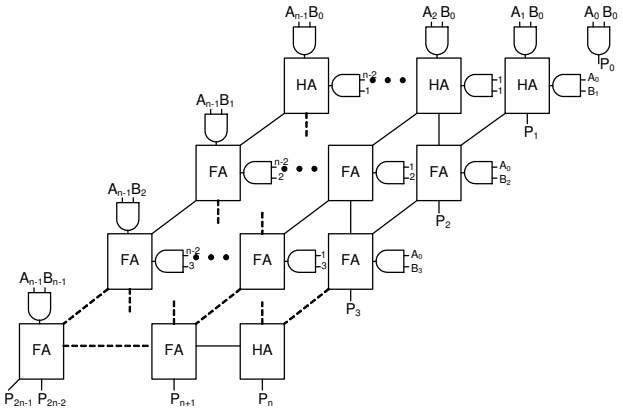

Figure 3. Structure of an n-bit multiplier.

test vectors for c6288, which are minimized using ILP (CPU time over one day on Sun Ultra-5) to obtain a test set of 10 vectors. This, we believe, is the lowest ever achieved for the circuit c6288. The vector sets are given in Table 5. Recursive rounding found a 12 vector set in 301 seconds.

6. Conclusion

We have shown that test minimization for single and N-detection can be efficiently done (in polynomial-time) by the new procedure of linear programming and recursive rounding. The quality of this result is almost the same as that of integer linear program (ILP), which is capable of exact minimality but has exponential computing complexity. In practice, the quality of the ILP method is compromised due to limits on the CPU time. A recently published [12] heuristic method solves this problem as a set covering problem using polynomial-time reduction techniques, such as, essentiality, row dominance and column dominance, adopted from logic synthesis. A greedy heuristic breaks cyclic choices of vectors. The method takes polynomial time and the result is comparable to the ILP solution. However, a comparison with the presented polynomial-time LP/recursive rounding method is not yet available. We do not know whether the 10-vector set we give

Table 5. Single-detect tests for multipliers.

Vector No.	Four-bit multiplier	
	Test set 1	Test set 2
1	0011 0110	1111 1100
2	0111 1101	1111 0011
3	1010 1111	1101 1111
4	1101 1111	1010 1111
5	1110 1100	0111 0110
6	1111 0011	0110 1101

Vector No.	Six-bit multiplier
1	001101 110111
2	011011 001011
3	011011 110100
4	100100 101110
5	110110 110101
6	110111 011011
7	111111 111110

Vector No.	Sixteen-bit multiplier, c6288
1	1101101101101101101111111111111
2	0110110110110110111111111111111
3	0000000000000000010111111111111
4	1011011011011011101111111111111
5	1111111111111111101010101010101
6	1111111111111111011010101010101
7	0011111111111101110101010101011
8	0011111111111101101010101010101
9	1110110110110110001011111111111
10	1101101101101100101010101010101

for c6288 is the ultimate minimum and the search may continue.

References

[1] S. B. Akers, C. Joseph, and B. Krishnamurthy, "On the Role of Independent Fault Sets in the Generation of Minimal Test Sets," in *Proc. International Test Conf.*, 1987, pp. 1100–1107.

[2] M. E. Amyeen, S. Venkataraman, A. Ojha, and S. Lee, "Evaluation of the Quality of N-Detect Scan ATPG Patterns on a Processor," in *Proc. International Test Conf.*, 2004, pp. 669–678.

[3] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, P. Krishnamurthy, N. Tamarapali, K.-H. Tsai, and J. Rajski, "Impact of Multiple-Detect Test Patterns on Product Quality," in *Proc. International Test Conf.*, 2003, pp. 1031–1040.

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

[5] J. Dworak, "An Analysis of Defect Detection and Site Observation Counts for Weighted Random Patterns and Compact Test Pattern Sets," in *Proc. North Atlantic Test Workshop*, 2006, pp. 183–190.

[6] P. F. Flores, H. C. Neto, and J. P. Marques-Silva, "An Exact Solution to the Minimum Size Test Pattern Problem," *ACM Trans. Design Autom. Electronic Sys.*, vol. 6, no. 4, pp. 629–644, oct 2001.

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

[8] M. R. Grimaila, S. Lee, J. Dworak, K. M. Butler, B. Stewart, H. Balachandran, B. Houchins, V. Mathur, J. Park, L.-C. Wang, and M. R. Mercer, "REDO–Random Excitation and Deterministic Observation – First Commercial Experiment," in *Proc. IEEE VLSI Test Symp.*, 1999, pp. 268–274.

[9] R. Guo, S. Mitra, E. Amyeen, J. Lee, S. Sivaraj, and S. Venkataraman, "Evaluation of Test Metrics: Stuck-at, Bridge Coverage Estimate and Gate Exhaustive," in *Proc. 24th IEEE VLSI Test Symp.*, 2006, pp. 66–77.

[10] I. Hamzaoglu and J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits," *IEEE Trans. on CAD*, vol. 19, no. 8, pp. 957–963, Aug. 2000.

[11] D. S. Hochbaum, "An Optimal Test Compression Procedure for Combinational Circuits," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 10, pp. 1294–1299, oct 1996.

[12] Y. Huang, "On N-Detect Pattern Set Optimization," in *Proc. 7th International Symp. on Quality Electronic Design (ISQED'06)*, Mar. 2006, pp. 445–450.

[13] S. Kajihara and T. Sasao, "On the Adders with Minimum Tests," in *Proc. Asian Test Symp.*, 1997, pp. 10–15.

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

[15] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Tech. Report 12-93, Dept. of Elec. Eng., Virginia Poly. Inst. and St. Univ., Blacksburg, Virginia, 1993.

[16] S. Lee, B. Cobb, J. Dworak, M. R. Grimaila, and M. R. Mercer, "A New ATPG Algorithm to Limit Test Set Size and Achieve Multiple Detections of all Faults," in *Proc. Design Automation and Test in Europe Conf.*, 2002, pp. 268–274.

[17] J. P. Marques-Silva, "Integer Programming Models for Optimization Problems in Test Generation," in *Proc. IEEE Asia-South Pacific Design Automation Conf.*, 1998, pp. 481–487.

[18] E. J. McCluskey and C. W. Tseng, "Stuck-Fault Tests vs. Actual Defects," in *Proc. International Test Conf.*, 2000, pp. 336–343.

[19] I. Pomeranz and S. M. Reddy, "Forming N-Detection Test Sets from One-Detection Test Sets Without Test Generation," in *Proc. International Test Conf.*, 2005, pp. 527–535.

[20] P. Raghavan and C. D. Thompson, "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[21] E. N. Tran, V. Kasulasrinivas, and S. Chakravarty, "Silicon Evaluation of Logic Proximity Bridge Patterns," in *Proc. 24th IEEE VLSI Test Symp.*, 2006, pp. 78–83.

[22] S. Venkataraman, S. Sivaraj, E. Amyeen, S. Lee, A. Ojha, and R. Guo, "An Experimental Study of N-Detect Scan ATPG Patterns on a Processor," in *Proc. IEEE VLSI Test Symp.*, 2004, pp. 23–29.