Automatic Test Pattern Generation for Maximal Circuit Noise in Multiple Aggressor Crosstalk Faults

Kunal P. Ganeshpure, Sandip Kundu
University of Massachusetts Amherst
{kganeshp, kundu}@ecs.umass.edu

Abstract

Decreasing process geometries and increasing operating frequencies have made VLSI circuits more susceptible to signal integrity related failures. Capacitive crosstalk is one of the causes of such kind of failures. Crosstalk fault results from switching of neighboring lines that are capacitively coupled. Long nets are more susceptible to crosstalk faults because they tend to have a higher coupling capacitance to overall capacitance ratio. A typical long net has multiple aggressors. In generating patterns to create maximal crosstalk noise, it may not be possible to activate all aggressors at the same time. Therefore, pattern generation must focus on activating a maximal subset of aggressors weighted by actual coupling capacitance values. This is a variant of max-satisfiability problem. Unlike a traditional max-satisfiability problem, here we must deal with signal propagation to an observable output. In this paper, we present a novel solution that combines 0-1 Integer Linear Program (ILP) with traditional stuck-at fault ATPG. The maximal aggressor activation is formulated as a linear programming problem while the fault effect propagation is treated as an ATPG problem. The problems are separated by min-cut circuit partitioning technique based on Kernighan-Lin-Fiduccia-Mattheyses (KLFM) method. This proposed technique was applied to ISCAS 85 benchmark circuits. Results indicated that 75-100% of the aggressors could be switched for generating crosstalk noise while satisfying requirement of sensitizing a path to the output.

I. Introduction

Increase in the transistor density and switching speed has led to an increasing number of signal integrity related failures in VLSI circuits [1]. Capacitive crosstalk is one of the major sources of signal integrity related failures. Crosstalk fault results from parasitic coupling between adjacent signal nets and is more common in nets that have weaker drivers relative to their adjacent peers [2].

This work was supported in part by grants from the Semiconductor Research Corporation and from the Intel Corporation.

Crosstalk fault effects can be classified into two types: crosstalk induced pulses and crosstalk induced delays. In the first case, the victim line remains in a static state, while one or more aggressor lines are switching. The amplitude and the width of the pulse depends, among other factors, on relative switching time of the aggressors, the amount of coupling capacitance and the relative transition times of the aggressors. In the second case of crosstalk faults, both the aggressor(s) and victim lines have simultaneous or near simultaneous transitions. If the aggressor and the victim lines transit in the opposite direction, then there will be an increase in transition delay for the victim. During crosstalk ATPG, patterns are chosen carefully to sensitize the victim node to an observable output. If the increase in transition delay at the victim node is significant, it can be detected by observing the output thus sensitized.

Current trends in integrated circuit design indicate that interconnect sidewall coupling capacitances can be significant and can create severe design and test problems. These problems are known to be aggravated by variations in the fabrication process [1].

If it were not for stringent area and performance requirements, an error due to crosstalk observed during validation could be eliminated by resizing drivers, re-routing signals, shielding interconnect lines with power distribution lines and other such redesign techniques. However, redesign may be very expensive in terms of design effort and its effectiveness may be offset by process variation. Thus, these problems need to be tested during manufacturing [6].

Crosstalk faults are observed more frequently for long nets. A long net may have multiple fan-outs and may be routed through multiple levels of interconnect metals. Thus, a typical long net is capacitively coupled with a multiple aggressors. Due to sharing of logic, it may not be possible to excite all aggressors while simultaneously sensitizing a victim net. From an ATPG point of view, the next best solution is to switch a set of aggressors that maximizes the switching of the total coupling capacitance. This is the problem thrust of this paper.

In this paper, we present a novel ATPG technique to generate patterns that will excite the worst case delay at the victim by switching maximal set of aggressors.

The rest of the paper is organized as follows: in section II we review previous work. Section III describes the problem statement. In section IV, the proposed Crosstalk ATPG algorithm is explained. This is followed by results for ISCAS85 benchmark circuits in section V. We conclude and propose future work in section VI.

II. Previous Work

Crosstalk noise induced errors are a significant source of signal integrity problems in deep submicron technology [3]. Chen, Gupta and Breuer presented a crosstalk ATPG solution for single aggressor, single victim scenario [6]. Bai, Dey and Krstic proposed a heuristic solution for multiple aggressor crosstalk ATPG problem [4]. In their approach, an implication graph is constructed to determine a feasible set of aggressors (a set of aggressors that could be switched to cause maximum crosstalk given the Boolean constraints of the circuit) and then a modified version of PODEM is used to determine a pattern pair that satisfies both feasible aggressor set excitation and fault propagation. Lee, Nordquist and Abraham presented an ATPG technique for crosstalk induced glitches but did not consider crosstalk induced delay [5]. A mixed signal test generator was proposed by Chen, Gupta and Breuer in [6], which not only considered static signal values but also dynamic signals like transitions and glitches as the possible input signals. Timed test pattern generation for CMOS domino circuits has been proposed by Kundu and Blanton in [7]. Both [7] and [8] consider multiple aggressors but employ computationally expensive circuit level timing simulations. A Genetic Algorithm based test generation for crosstalk induced faults has been proposed by Krstic, Liou and Jiang in [9] while Chen and Keutzer proposed a SAT based method [10]. A Built in Self Test method to detect crosstalk faults has been proposed by Shimizu e.t. a.l. in [11].

Kundu e.t. a.l. proposed generalized fault model for multiple aggressor crosstalk faults in [1] but the ATPG aspect was not considered.

It can be seen that most of the above techniques do not consider the effect of multiple aggressors on the victim node.

III. Problem statement

The problem of generating pattern that results in maximal noise has two aspects:

Switching aggressor to cause maximal delay at victim: As the victim net is coupled with multiple aggressors, switching aggressors to create maximal delay at victim is a max-satisfiability problem. Max-satisfiability problem is NP-complete [21]. This is because we need to switch maximum set of aggressors with cognizance of Boolean relationship between aggressors.

Propagation of fault effect to the output: In addition to maximal noise creation, the pattern must propagate the fault effect at the victim net to an observable output.

It may not always be possible to switch all the aggressors in a desired fashion and at the same time propagate the fault effect to an output. So we seek to find the best that can be achieved.

The following example illustrates the above problems.

Example: In the circuit shown in Figure 1, the gate G0 drives victim net (V) while the coupled aggressor lines A1, A2, A3 and A4 are driven by gates G1, G2, G3 and G4 respectively. The numbers in the box associated with the aggressors indicate the coupling weight. Higher the coupling weight more is the delay impact on the victim. Total delay introduced is proportional to the sum of the coupling weights of all the aggressors switching in opposite direction of the victim (desired direction of switching).

If we follow a greedy approach by applying the pattern pair {0,0,1,↓,↓,1,1} at the inputs {*a,b,c,d,e,f,g*}, where the nodes *d* and *e* are transitioning from high to low, to greedily switch the aggressor A2 (with highest coupling weight of 100), it can be seen that the aggressor A3 (coupling weight = 20) will also switch in the desired direction producing a total coupling weight of 120 = (100+20). As both aggressors A2 and A3 couple to the net connected to the input of gates G6 and G5, they will experience slow-to-rise fault. We can then propagate the fault effect via gate G6 by setting the input *g* to 1.

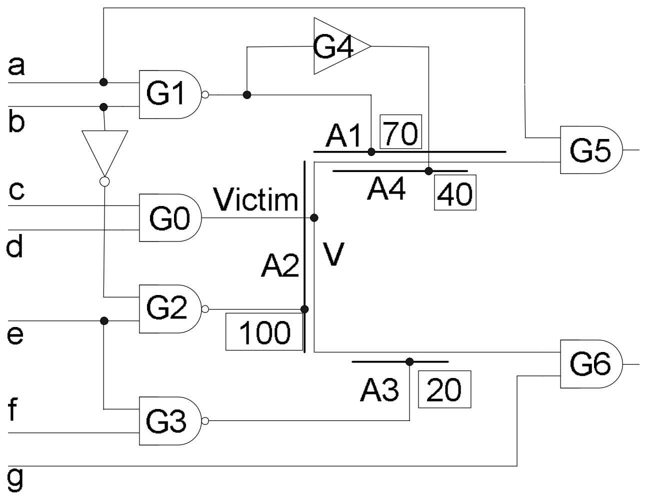

Figure 1: Example circuit showing aggressors and victims

Now consider the second input pattern pair {↓,1,1,↓,↓,1,1} which switches aggressors A1, A4 and A3 in the desired direction ({A1,A4,A3,V}= {↑,↑,↑,↓}) to produce a total coupling weight of 130. Thus, a greedy approach does not result in the maximal aggressor switching.

For this pattern the input of the gates G5 and G6 will be delayed. If G6 has large slack, then this pattern will not propagate the fault effect through G6, while the fault effect is squashed at gate G5 by a controlling side input. This example illustrates that the max-satisfiability problem of switching maximal aggressor weight is also connected to the propagation problem.

For the pattern {1,0,1,↓,↓,1,1} the aggressors A1, A4 and A3 switch in desired direction a total ({A2,A3 ,V}={↑,↑,↓}) coupling weight of 120. We can propagate the faulty effect through both the AND gates G5 and G6 as their side inputs are 1. Thus this pattern is better than the previous two. This illustrates the qualitative nature of multiple aggressor crosstalk fault test generation problem. ■

IV. Proposed Solution

Max-satisfiability is known to be an intractable problem. All intractable problems are solved by heuristic techniques. In this section we present a heuristic solution to our problem. In the proposed approach max-satisfiability and fault propagation problems are solved separately. Given a list of aggressors and victims, the steps that are followed are described next.

Step 1: Circuit Partitioning

Divide the circuit into two partitions such that the input logic cones of the aggressors and the victim belong to the left partition while the output logic cone of the victim belongs to the right partition.

With these constraints there are multiple ways to cut the circuit to create left and right partitions. The only requirement is that cut line should pass through the victim net. For efficiency reasons which will become apparent in the subsequent discussion, the cut line should pass through least number of circuit nodes. The cut-points represent outputs for the left partition, while they represent inputs for the right partition. Please note that the victim net itself is an output of the left partition while it is an input for the right partition.

Step 2: Fault Effect Propagation

Fault effect propagates through the right partition to a primary output. In this step, a stuck-at 0 or 1 value is placed on the victim line and a stuck-at fault ATPG is invoked to generate an input pattern that specifies requirements for the cut points.

Step 3: Maximal Noise Generation

In this step the left partition is targeted because it includes all the aggressors. The constraints derived from step 2 are placed on the outputs of the left partition. The aggressor weights are formulated into an ILP equation. It has been shown in the past that ILP equations can be formulated to represent Boolean function of logic gates [22]. Thus, both the logic circuit and the maximal aggressor weight equations are represented as ILP equations. Next, we use an ILP solver to solve simultaneous logic constraints at the outputs of the left partition and the maximal aggressor switching requirement. This represents the final test vector pair. The above steps are explained in more detail below.

A. Circuit Partitioning

The initial partition is obtained by cutting the circuit along the input to output level of the victim such that all the nodes with level lower than or equal to that of the victim belong to the left partition while the nodes with level higher than the victim are present in the right partition. Next the aggressors and their input cones that are part of the right partition are moved to the left partition.

Next the Kernighan-Lin-Fiduccia-Mattheyses (KLFM) algorithm [17] is applied to the resultant circuit to reduce the cut size. It has been observed in [16] that KLFM algorithm with a random initial cut gives fairly good result as compared to most other types of initial partitioning techniques. Reducing the cut size reduces the number of constraints in the ILP formulation which leads to better solution. In Table II we have compared results obtained with and without the application of the KLFM algorithm. Moreover, the table also shows the reduction in the cut size obtained form the KLFM algorithm.

B. Stuck-at Fault ATPG

Stuck at fault ATPG was performed on the right half of the circuit using a publicly available ATPG tool, ATALANTA [19] to determine the input node values of the right partition that propagate the fault effect to an output. There may be multiple ways to propagate a fault effect to a primary output. Slack based heuristics to guide fault propagation through longest paths have been suggested in [23]. The input pattern generated by ATPG becomes a constraint for the left partition. The left partition may not be able to satisfy these constraints or may satisfy them at the expense of compromising on maximal aggressor weight. Thus, a single test vector to constrain the left partition may not be optimal. Our solution to this problem is heuristic and the results show that almost always, a single propagation pattern is satisfied by the left partition with very high percentage of the total aggressor weight switched in the desired direction.

C. ILP formulation

In order to obtain the worst case crosstalk condition in the circuit, ILP formulation is done for the circuit in the left partition by writing the ILP equations for the logic gates [22]. The ILP equations of the gates are formed by using the clausal description of the function of the gates given in

[18]. For example for a AND gate with inputs *a*, *b* and output *c*, we can describe all 4 input-output combinations as shown in Figure 2.

$$\begin{aligned}&\bar{a} \Rightarrow \bar{c} \text{ or } a+(1-c)\geq 1\\ &\bar{b} \Rightarrow \bar{c} \text{ or } b+(1-c)\geq 1\\ &ab \Rightarrow c \text{ or } (1-a)+(1-b)+c\geq 1\\ &a,b,c\in[0,1]\end{aligned}$$

Figure 2: ILP equations for an AND gate

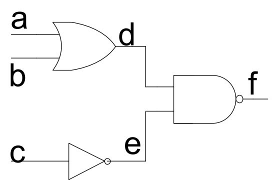

Figure 3: Circuit with internal nodes names.

For the circuit shown in Figure 3 above the complete set of ILP equations are shown in Figure 4 below.

$$\left.\begin{aligned}d+f&\geq 1\\ e+f&\geq 1\\ (1-d)+(1-e)+(1-f)&\geq 1\end{aligned}\right\}\text{For NAND gate}$$

$$c+e=1 \quad \text{For NOT gate}$$

$$\left.\begin{aligned}(1-a)+d&\geq 1\\ (1-b)+d&\geq 1\\ b+a+(1-d)&\geq 1\end{aligned}\right\}\text{For OR gate}$$

$$a,b,c,d,e,f\in[0,1]$$

Figure 4: ILP equations for circuit in the Figure 3

For the ILP formulation, the circuit is duplicated into two copies, say *A* and *B*, each representing one time frame corresponding to the first and the second patterns respectively. All the nodes are renamed by adding the prefix '*a*' for the circuit corresponding to the first pattern and prefix '*b*' for the one corresponding to the second pattern. Now ILP equations are formed for each of the copies of the circuit. ILP equations are only formed for the nodes in the input logic cones of the aggressors, victims and the cut nodes that are *not* assigned don't care value by the ATPG.

1) Formation of Objective Function

The delay of the victim net depends on the direction of the transition of the aggressor net. By using the Elmore delay equation we can determine the delay [20].

$$T_{p,k} = gC_w(0.38R_w + 0.69R_D)$$

Where

$T_{p,k}$ is the propagation delay of the victim net.

$C_w = c_w l_w : c_w$ is the capacitance per unit length of the wire and l_w is the wire length

$R_w = r_w l_w$; r_w is the resistance per unit length of wire

R_D is the equivalent resistance of the driver

g *is the correction factor*

The correction factor *g*, which determines the delay introduced by crosstalk, is a function of the capacitance ratio $r = c_i/c_w$ and the activities on the wire. Here c_i is the coupling capacitance per unit length. The value of g for a set of transitions is given in Table I below. Here the victim *vic* is capacitively coupled with an aggressor *agg*.

Table I: Values of g for different transition at aggressor and victim

vic	*agg*	*g*
Up	Up	1
Up	No	$1+r$
Up	Down	$1+2r$

The objective function in the ILP formulation calculates *g* depending on the transition on the aggressors and victim according to the Table I above. The expression for *g* is of the form

$$g = (1 + k * r)$$

Where k is an integer constant.

To determine *k* we define Boolean variables *AV* and *V* for each of the aggressor victim pairs. *AV* is true if both the aggressor and victim are switching in the opposite directions while *V* is true if the victim is switching while the aggressor is not switching. The following example explains how to determine *AV* and *V*.

Example: Consider a pair of capacitively coupled nets having names *agg* and *vic*. Net *agg* is the aggressor and *vic* is the victim. After duplication of circuit the nodes *agg* and *vic* are renamed to a_{agg} , a_{vic} in the copy *A* for first time frame and b_{agg} , b_{vic} in the copy *B* for second time frame.

By definition, *AV* is true when *agg* and *vic* switch in opposite directions. To make *agg* and *vic* switch, $(a_{agg} \oplus b_{agg}) = 1$ and $(a_{vic} \oplus b_{vic}) = 1$ must be satisfied. And for them to switch in opposite direction $\overline{(a_{agg} \oplus b_{vic})} = 1$ must be satisfied. Hence *AV* is:

$$AV = (a_{agg} \oplus b_{agg}) \bullet (a_{vic} \oplus b_{vic}) \bullet \overline{(a_{agg} \oplus b_{vic})}$$

Similarly V is true when *vic* switches and *agg* does not. To make *vic* switch $(a_{vic} \oplus b_{vic}) = 1$ must be satisfied while for *agg* to remain constant $\overline{(a_{agg} \oplus b_{agg})} = 1$ must be satisfied. Hence V is:

$$V = (a_{vic} \oplus b_{vic}) \bullet \overline{(a_{agg} \oplus b_{agg})}$$

The XOR, NOT and AND functions in the above Boolean expressions are converted to the corresponding ILP equations.

Now from the Table I we can see that:

$k = 2$ if *agg* and *vic* are switching in opposite direction

$k = 1$ if only *vic* is switching

$k = 0$ otherwise

Thus we can represent k in terms of A and AV

$$k = (2 * AV + V)$$

The objective function becomes

$$Obj = 1 + [(2 * AV + V) * r]$$

For n number of aggressors we need to determine the equations for AV_i and V_i for each of the n aggressor victim pairs. Then the resultant objective function is:

$$Obj = \sum_i [(2 * AV_i + V_i) * r_i)] + 1$$

V. Results

The crosstalk ATPG was run on all ISCAS 85 benchmark circuits to obtain the results described below. In order to run ATPG, a crosstalk fault list is needed. The crosstalk fault list should be created by RC extraction from the physical layout. However, for the purpose of this paper we created the fault list by random selection of nets. The aggressor and victim nets are selected randomly such that every node in the circuit has equal probability of getting selected. The maximum number or aggressors per victim is limited to 7. Moreover the coupling capacitance ratio r_i for each aggressor is also selected randomly between 1 and 0. Actual data from RC extraction can be used to create the fault list; however the focus of this paper is the second half of this problem, namely the ATPG process so we did not work on the overall methodology segment of this solution.

The ATPG result is presented in Table II. To illustrate the efficiency of the solution, it compares the actual weight that is switched against the maximum possible weight that could be switched. It can be seen that for most circuits the proposed solution is able to switch maximum weight.

Table II also compares the results obtained with and without using KLFM min-cut algorithm. The dashes in the table represent that no solution was found due to conflicting requirements between excitation and propagation.

Table II: Results for ISCAS85 benchmarks circuits

Ckt. name	# of aggs	Max. Case Wt.	Without KLFM			With KLFM		
			Cut size	Actual Wt.	% of max	Cut size	Actual Wt.	% of max
C17	4	4.35	2	3.25	74.71	2	3.25	74.71
	4	5.39	3	4.10	76.06	3	4.10	76.06
	2	1.87	3	1.87	100	3	1.87	100
C432	6	5.24	31	4.30	79.33	16	4.75	90.64
	6	6.99	57	6.71	96.00	46	6.20	88.68
	4	4.57	56	4.57	100	43	3.98	88.8
C499	7	5.98	62	5.71	95.48	62	5.71	95.48
	4	5.62	46	5.62	100	44	5.62	100
	6	6.17	55	5.86	94.9	44	5.86	94.9
C880	7	5.15	25	5.15	100	14	5.15	100
	7	10.89	47	9.93	91.18	36	9.93	91.18
	6	5.84	80	5.84	100	55	5.84	100
C1355	7	6.42	63	6.29	98.1	42	6.29	98.1
	7	6.22	73	-	-	53	6.22	100
	7	6.97	74	6.25	89.67	54	6.25	89.67
C1908	7	6.09	94	4.69	76.89	60	4.69	76.89
	7	7.07	135	5.74	81.18	85	5.74	81.18
	7	8.87	113	8.87	100	71	8.87	100
C2670	7	5.42	78	-	-	29	5.02	92.62
	7	6.98	212	6.98	100	132	6.98	100
	7	6.76	188	6.76	100	92	6.76	100
C3540	7	8.17	68	8.17	100	39	8.17	100
	7	5.49	139	5.00	100	76	5.00	100
	7	6.10	185	5.38	88.1	117	5.38	88.1
C5315	7	7.73	140	7.73	100	68	7.73	100
	5	5.25	358	5.25	100	206	5.25	100
	7	5.34	197	4.47	83.70	193	4.47	83.70
C6288	7	8.14	70	-	-	54	7.41	91.90
	7	5.83	156	5.83	100	73	-	-
	6	5.04	91	-	-	56	-	-
C7552	7	6.61	263	6.61	100	122	6.61	100
	7	9.95	424	9.95	100	192	9.95	100
	7	7.16	338	7.16	100	140	7.16	100

The advantage obtained from KLFM algorithm is evident for the circuits c1355 (row number 2 of c1355 circuit), c2670 (row number 1 of c2670 circuit) and c6288 (row number 1 of c6288 circuit). In all the above cases a solution is only obtained when KLFM algorithm is used to reduce the cut size thus reducing the number of constraints.

ATPG was solved using ATLANTA [19] and the ILP problem was solved using GLPK, a GNU Linear Programming Kit [24].

To obtain the above results, the crosstalk ATPG was run on a Dell PowerEdge 2800 server with 2.8GHz Dual Core Intel Xeon Processor, 2MB L2 cache and 2GB RAM. A workload consisting of all the circuits excluding c6288 ran in less than 30 minutes.

For the circuit c6288 the solution could not be found as it exceeded the 1 hour time limit for the ILP solver. c6288, a 32 bit multiplier, is a known nemesis for many ATPG tools as it has very large number of re-converging paths.

VI. Conclusion and Future Work

In this paper we presented a novel ATPG technique to generate a two pattern test for multiple aggressor crosstalk faults. The problem of maximizing the effect of the aggressor on the victim is a max-satisfiability problem which is known to be intractable. All intractable problems are solved by heuristic techniques. Our heuristic technique is based on partitioning the circuit into two parts. One partition is used for fault effect propagation using a standard ATPG while the other part is solved by an ILP solver for max-satisfiability. All results are validated using circuit simulation. The results show that the proposed approach is highly efficient for multiple aggressor crosstalk fault ATPG. The results could be improved further by deploying slack based heuristic solution to propagate fault effect through a path with least slack. This requires integration of the solution with static timing analysis. Moreover, our ATPG does not consider crosstalk induced glitches. A modification in the ILP formulation will be required to accommodate crosstalk induced glitches into our present setup. This is a topic of future investigation.

VII. References

[1] S. T. Zachariah , Y.Chang , S. Kundu , C. Tirumurti, "On Modeling Crosstalk Faults", *Proceedings of the conference on Design, Automation and Test in Europe*, March 03-07, 2003, p.10490

[2] Y. Chen, S. K. Gupta, and M. A. Breuer, "Analytic models for crosstalk delay and pulse analysis for non-ideal inputs," *in Proc. Int. Test Conf., Washington, DC, 1997*, pp. 809–818.

[3] C. Werner, R. Göttsche, A. Wörner, Ulrich Ramacher: "Crosstalk noise in future digital CMOS circuits" *DATE 2001:* 331-335

[4] Xiaoliang Bai, Sujit Dey, Angela Krstic, "HyAC: A Hybrid Structural SAT Based ATPG for Crosstalk." *ITC 2003:* 112-121

[5] K. T. Lee, C. Nordquist and J. A. Abraham "Automatic test pattern generation for crosstalk glitches in digital circuits", *Proc. VLSI Test Symposium, 1998*, pp. 34-39.

[6] W. Y. Chen, S. K. Gupta and M. A. Breuer, "Test generation in VLSI circuits for crosstalk noise", *Proc. Int'l Test Conf.*, 1998, pp. 641–650.

[7] Rahul Kundu, R. D. (Shawn) Blanton, "Timed Test Generation Crosstalk Switch Failures in Domino CMOS Circuits", *VTS 2002:* 379-388

[8] Bipul Chandra Paul, Kaushik Roy, "Testing Crosstalk Induced Delay Faults in Static CMOS Circuits Through Dynamic Timing Analysis." *ITC 2002*: 384-390

[9] Angela Krstic, Jing-Jia Liou, Yi-Min Jiang, Kwang-Ting Cheng: "Delay testing considering crosstalk-induced effects." *ITC 2001*: 558-567

[10] Pinhong Chen, Kurt Keutzer: "Towards true crosstalk noise analysis." *ICCAD 1999*: 132-138W

[11] Kazuya Shimizu, Noriyoshi Itazaki, Kozo Kinoshita, "Built-in Self-Test for crosstalk faults in a digital VLSI." *Systems and Computers in Japan 33(13):* 35-47 (2002)

[12] D. S. Gao, A. T. Yang and S. M. Kang, "Modeling and simulation of interconnection delays and crosstalk in high-speed integrated circuits", *IEEE Trans. on Circuits and Systems*, Vol. 37, pp.1-9, January 1990.

[13] A. K. Goel and Y. R. Huang, "Modeling of crosstalk among the GaAs VLSI connections", *IEE Proc. Part G*, Vol. 136, pp.361-368, 1989.

[14] W. Y. Chen, S. K. Gupta and M. A. Breuer, "Test generation for crosstalk-induced delay in integrated circuits", Proc. *Int'l Test Conf.*, 1999, pp. 191-200.

[15] F. Moll and A. Rubio, "Methodology of detection of spurious signals in VLSI circuits", *Proc. Europe Test Conference*, 1993, pp. 491-496.

[16] S. Hauck, G. Borriello, "An evaluation of bipartitioning techniques", *Proceedings of the 16th Conference on Advanced Research in VLSI (ARVLSI'95),* p.383, March 27-29, 1995.

[17] C. M. Fiduccia, R. M. Mattheyses, "A Linear-Time Heuristic for Improved Network Partitions", *Design Automation Conference,* pp. 241-247, 1982.

[18] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability", *IEEE Trans. Computer-Aided Design*, Vol. 11, No. 1, Jan. 1992, pp. 4-15.

[19] H. K. Lee and D. S. Ha, "Atalanta: an Efficient ATPG for Combinational Circuits", *Technical Report, 93-12,* Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993.

[20] J. M. Rabaey, A. Chandrakasan, B. Nikolic, "Digital Integrated Circuits: A Design Perspective. 2nd Edition", *Prentice Hall 2003*, pp. 446-452.

[21] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", *New York: W. H. Freeman, 1979.*

[22] R. Fortet, "Applications de l'algebre de Boole en recherche operationelle", *Revue Francaise de Recherche Operationelle*, vol. 4, pp. 17-26, 1960.

[23] W. N. Li, S. M. Reddy, S. K. Sahni, "On path selection in combinational logic circuits", *IEEE Trans. on CAD of Integrated Circuits and Systems* 8(1): 56-63 (1989)

[24] http://www.gnu.org/software/glpk/