Journal of Theoretical and Applied Information Technology
20th April 2013. Vol. 50 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN APPROACH TO GENERATE TESTS FOR MULTIPLE VICTIM LINES OF CROSSTALK FAULTS IN INTEGRATED CIRCUITS

PAN ZHONGLIANG, CHEN LING
Department of Electronics, School of Physics and Telecommunications Engineering,
South China Normal University, Guangzhou 510006, China.
E-mail: panz@scnu.edu.cn

ABSTRACT

The continuous progress in circuit technology offers the opportunity to design circuit chip with high integration density. The parasitic capacitances among the signal lines in circuits are increased; this results in the arising of circuit noises such as crosstalk. The coupling effects of a crosstalk occur between the aggressor lines and victim lines. If the strength of the aggressor line is large, then it may produce interferes for a lot of adjacent lines. Therefore, the type of the crosstalk fault consists of an aggressor line and multiple victim lines are investigated in this paper. A new test vector generation approach is presented for the crosstalk faults with multiple victim lines. The approach uses a lot of structure characteristics of the circuit under test, performs the partition of circuit cones, path selections, value implications of signal lines, etc., and the test vectors are produced by forward tracing and backtrack tracing. The experimental results for a lot of digital circuits show that the test vectors of the crosstalk faults with multiple victim lines can be generated by using the approach proposed in this paper, and the feasible fault coverage can be obtained.

Keywords: *Integrated circuits, crosstalk noise, multiple victim lines, test approach, test vector generation*.

1. INTRODUCTION

As the VLSI design technology enters deep-submicron (DSM) domain, the feature size and density of integrated circuits are decreased, while the signal speed increases, which results in the appearing of the noises. The noises may harm the functions of circuits.

The noises can be defined as any deviation of a node (signal line) voltage from nominal high or low values [1]. The crosstalk is one of dominant noises, it is the voltage induced on a node due to coupling among adjacent nets. The coupling effects can be divided into two following types: capacitive coupling and inductive coupling [2]. The crosstalk noises may result in unpredictable timing, for example, a crosstalk noise on a clock signal line of synchronous circuits can cause a wrong value to be clocked in a flip-flop, this can results in the circuit failure. Therefore, it is needed to perform the test of crosstalks to insure the reliability of circuits.

The most of the related work in the area of the test for crosstalk has concentrated on the following aspects: the coupling effects of crosstalks, the test algorithm for crosstalk, and the crosstalk in interconnect lines such as bus lines, the built in self-test (BIST) of the crosstalk, etc.

First of all, the coupling effects of crosstalks were investigated. Ferragina et al [3] analyzed the crosstalk effects due to the current pulses drawn from voltage supplies in analog-digital mixed-signal CMOS ICs, the simulations results demonstrated that the disturbances of switching currents, and the interconnection affects, can degrade the circuit performance. Roy et al [4] discussed the closed-form matrix rational-approximation method to model the delay and crosstalk noise of coupled RLC on-chip interconnects, the method for any rational order can obtain the approximations in terms of the predetermined coefficients and parameters. Jongsun et al [5] investigated the crosstalks between the adjacent cells in SRAM, and proposed a test method that a negative voltage stress was applied to bit lines for performing the detection of cell coupling. Mbairi [6] discussed the approach that uses the guard traces to reduce the crosstalk between differential transmission lines pair, the crosstalk effects of differential lines were treated with and without guard trace separation between the differential line pairs.

In the aspect of the test algorithm for crosstalk, Sunghoon et al [7] investigated multiple-aggressor crosstalk faults to maximize the noise of the victim line, and presented a test pattern generation method for delay faults considering crosstalk-induced delay effects, the method was based on the principle of the conventional automatic test pattern generation for delay faults. Hasan et al [8] discussed the glitches and delay faults being introduced by the crosstalk, and proposed a test generation and compaction algorithm for test patterns of crosstalk faults. The algorithm simultaneously considers the coupling capacitance, timing and functional incompatibilities between the victim and aggressor nets, and can produce the maximum crosstalk noise. Jianxun et al [9] investigated the crosstalk noise of programmable logic arrays (PLAs), used the characteristics of dynamic PLA crosstalk noise to design an automatic test pattern generation method, the method can detect the maximum crosstalk noises of product lines.

In the aspect of the crosstalk in the interconnect lines such as bus lines, Jaehoon et al [10] investigated the compact crosstalk test patterns for system on chip (SoC) and board level interconnects considering the physically effective aggressors, designed the 6λ test patterns for the multiple victim lines, where the λ was the effective distance among interconnect nets. Bengtsson et al [11] discussed the crosstalk-induced delay and glitch faults in network-on-chip interconnects, and presented a method for at-speed testing of crosstalk faults in asynchronous connection of chip, the method provided a complete mode for the detection of crosstalk-induced faults in on-chip communication infrastructure by using asynchronous handshaking protocols. Min et al [12] analyzed the glitch and crosstalk-induced delays in the SoC interconnect bus, and presented a pulse detector with an adjustable detection threshold, the detector can detect the glitches and crosstalk-induced delay.

In the aspect of built in self-test (BIST) for the crosstalk, Rudnicki et al [13] investigated the test-per-clock approach for the at-speed testing of crosstalk faults in long interconnects of SoC, used a linear feedback shift register (LFSR) to produce the test pattern of crosstalk faults, the LFSR consists of 2μ flip-flops, where the μ is the number of nets in a chip. Assadi et al [14] discussed the crosstalk noise in the FPGA, proposed a method to detect the crosstalk fault effects such as glitches and delays in FPGA, the method incorporated the test pattern generator to produce the test vectors and the analyzer to analyze the crosstalk faults. Der et al [15] investigated the crosstalk faults in embedded RAMs, gave a test algorithm and its BIST implementation for detecting the crosstalk faults on the address buses and data buses of RAMs. Jen et al [16] discussed the crosstalk faults in three-dimensional integrated circuits, and presented a BIST scheme for the post-bond test of through silicon via (TSV) with crosstalk faults.

Besides, the crosstalk effects with multiple aggressors were investigated. Sanyal et al [17] discussed the multiple aggressor crosstalk effects considering gate leakage, presented an automatic test pattern generation algorithm that uses 0-1 integer linear programming to maximize the cumulative voltage noise at a given victim net. Gope et al [18] investigated the crosstalk noises with multiple aggressors, gave a timing-driven test generator that can sensitize multiple aligned aggressors coupled to a delay-sensitive victim path, the test generator can detect the combination of a delay spot defect and crosstalk-induced slowdown. Ganeshpure et al [19] investigated the test approach for multiple aggressor crosstalk faults, presented a test pattern generation algorithm for multiple aggressor crosstalk faults considering the zero and unit delay, the algorithm can obtain the maximal aggressor excitation.

In this paper, a new test vector generation approach is presented for the crosstalk faults with multiple victim lines. The partition of circuit cones, path selections, forward tracing and backtrack tracing, etc, are used in this approach. This paper is organized as follows. Section 2 gives the model of the crosstalk faults with multiple victim lines. Section 3 presents the test vector generation approach based on circuit structure for the crosstalk faults. In Section 4, the experimental results for a lot of ISCAS'85 benchmark circuits are given. The Section 5 concludes this paper.

2. CROSSTALK WITH MULTIPLE VICTIM

The noise in VLSI circuit is the perturbation or interference on a signal line that causes the signal value to deviate from its original value. If the magnitude of noise is large enough, it can invert the logic value of signal line, and cause the malfunction of whole circuit. In general, the noise sources include inductive and capacitive crosstalk, power supply noise, thermal noise, flicker noise and shot noise, etc.

The crosstalk is one of the noises being induced by a signal line that interferes with another signal line. The crosstalk is produced primarily by the capacitive coupling and inductance coupling of some signal lines. A signal transition on a signal line can take effects on the adjacent signal lines. The crosstalk can cause many types of errors, such as circuit power swing, clock skew, signal delay, signal glitch, etc.

For two signal lines in a circuit, if the signal transition of 0 to 1(or 1 to 0) on a line produces coupling effects on another signal line, then the signal line is called aggressor line, the another signal line is called victim line. If the strength of the aggressor line is large, then it may produce interferes for a few of adjacent lines. In the following, we consider this type of crosstalk, i.e., a crosstalk fault consists of an aggressor line and multiple victim lines.

One of the main tasks of test generation for crosstalk faults is to produce the test vectors such that: when apply the test vectors to the circuit, results in that the correct circuit behavior and the faulty circuit behavior are different, i.e., the values of their primary output lines are not equivalent. A test vector of a crosstalk fault is a pair of circuit input vectors, i.e., it consists of two circuit input vectors. Let S=(S_1, S_2) is a test vector of a crosstalk fault, where the S_1 and S_2 are the input vectors of circuit. The S_1 is called the first vector; the S_2 is called the second vector.

An example of test vectors is shown for the C17 circuit in the Fig.1.

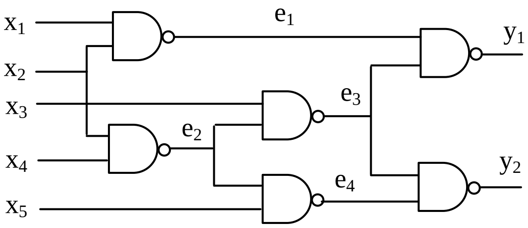

Fig.1 The C17 circuit.

Suppose there is a crosstalk fault in the C17 circuit, and the signal line e_1 is the aggressor line, the signal lines e_3 and e_4 are victim lines. The crosstalk fault is caused by a transition (0 to 1) in the line e_1 that produces the interferences (0 to 1) in the lines e_3 and e_4. The goal of test generation is to search for the circuit input vectors that can detect the given crosstalk fault. Here, the following vectors S_1=(x_1 x_2 x_3 x_4 x_5)=(1 1 1 0 1) and S_2 =(x_1 x_2 x_3 x_4 x_5)=(0 0 1 0 1) can be used to detect the crosstalk fault.

Apply the vectors S_1 and S_2 to the circuit primary inputs sequentially. If the circuit outputs are y_1=1 and y_2=1 for the S_1, the y_1=1 and y_2=1 for the S_2, then there is not crosstalk in the circuit. If the circuit outputs are y_1=1 and y_2=1 for the S_1, the y_1=0 and y_2=0 for the S_2, then there is the crosstalk in the circuit. Thus, the pair of vectors S_1 and S_2 can detect the crosstalk fault that the e_1 is the aggressor line and the e_3 and e_4 are victim lines.

3. TEST VECTOR GENERATION BASED ON CIRCUIT STRUCTURE

The test vector generation of crosstalk faults with multiple victim lines can be described as the process of determining the test vectors for a given fault in a given circuit. The procedure of the test generation includes following aspects: The activation of crosstalk faults, the selection of sensitizing path, fault propagation, etc.

For the circuit under test, let its primary input lines be x_1, x_2, ···, x_n, let its primary output lines be y_1, y_2, ···, y_L. Let the set of crosstalk faults in the circuit be **F**={f_1, f_2, ···, f_m}, where the number of faults is *m*. Each crosstalk fault f_i includes an aggressor line A_i and a lot of victim lines V_{i1}, V_{i2}, ···,V_{it}, where the number of victim lines is t for the aggressor line A_i.

The test generation procedure for the crosstalk faults with multiple victim lines consists of following steps:

Algorithm 1

Step 1. Put up the parameter k:=0; partition the circuit into L cones, the outputs of these cones are the circuit primary output lines y_1, y_2, ···, y_L, respectively, let these cones be D_1, D_2, ···, D_L.

Step 2. Put up the k:=k+1; Choose the crosstalk fault f_k from the set **F**.

Step 3. Activate the crosstalk fault f_k, i.e, set the aggressor line A_k of the f_k to the required transition, set the victim lines V_{k1}, V_{k2}, ···, V_{kt} to their value.

Step 4. Partition cones. For the circuit under test, partition the circuit into a lot of cones, where the number of cones is t+1. Let these cones are C_1, C_2, ···, C_{t+1}. The output of one cone is the aggressor line A_k, the outputs of another cones are V_{k1}, V_{k2}, ···, V_{kt}, respectively.

Step 5. Carry out the backtrack tracing operation and implication operation on the cones C_1, C_2, ···, C_{t+1} to obtain the first vector S_1.

Step 6. Locate the crosstalk signal lines A_k, V_{k1}, V_{k2}, ···, V_{kt} into the cones D_1, D_2, ···, D_L. Select the paths from the fault site to the output of cones. Carry out the forward tracing to sensitize the paths being selected, and perform the backward tracing to obtain the values of primary input lines, thus the second vector S_2 is obtained. The second vector S_2 can propagate the effect of the crosstalk fault to at least one of circuit primary outputs.

Step 7. If the first vector S_1 is not obtained in Step 5 or the second vector S_2 is not obtained in the Step 6, then the test vector of crosstalk fault f_k does not exist, else the test vector of the crosstalk fault f_k is S=(S_1, S_2).

Step 8. If the $k<m$ is true, then go to the Step 2, else stop the algorithm.

The detail implementation of the Algorithm 1 is given as follows.

In the Step 1 and Step 4 of the Algorithm 1, the method of partitioning cones is following. A cone in a circuit is defined by the set of all signal lines that can be found connected between a single output and the inputs that lead to the single output.

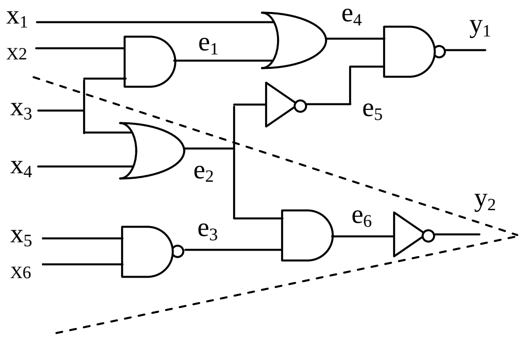

Fig.2 Cones in a circuit.

For example, the circuit shown in Fig.2 has two cones. The output of the first cone is y_1, the cone has nine nodes x_1, x_2, x_3, x_4, e_1, e_2, e_4, e_5 and y_1. The output of the second cone is y_2, the cone has eight nodes x_3, x_4, x_5, x_6, e_2, e_3, e_6 and y_2.

In the Step 3 of the Algorithm 1, the method of activating the crosstalk fault can be classified into following two cases:

Case 1: If the aggressor line A_k has a transition of 0 to 1, then when generate the first vector S_1, the value of the aggressor line A_k is set to 0, and the values of all victim lines V_{k1}, V_{k2}, ···, V_{kt} are set to 0; when generate the second vector S_2, the value of the aggressor line A_k is set to 1, and the values of all victim lines V_{k1}, V_{k2}, ···, V_{kt} are set to 0.

Case 2: If the aggressor line A_k has a transition of 1 to 0, then when generate the first vector S_1, the value of the aggressor line A_k is set to 1, and the values of all victim lines V_{k1}, V_{k2}, ···, V_{kt} are set to 1; when generate the second vector S_2, the value of the aggressor line A_k is set to 0, and the values of all victim lines V_{k1}, V_{k2}, ···, V_{kt} are set to 1.

In the Step 5 of the Algorithm 1, the first vector S_1 is obtained by backtrack tracing operation and implication operation. The implication operation is performed in the following mode: For an AND gate, if its output is 1, then its all inputs should be 1; if its output is 0, then at least one of its inputs should be 0. For an OR gate, if its output is 0, then its all inputs should be 0; if its output is 1, then at least one of its inputs should be 1.

The backtrack tracing operation is carried out in the following mode: The backtrack procedure traces the objective lines (A_k, V_{k1}, V_{k2}, ···, V_{kt}) back to one of the primary inputs and assigns a logic value to them. There are several possible paths from the objective lines to the primary inputs, and the selections of particular paths are made randomly. Because there maybe the conflicts in the assigning values to the signal lines, therefore in general, more than one backtrack process is needed to achieve the desired values on the objective lines (A_k, V_{k1}, V_{k2}, ···, V_{kt}).

We take a crosstalk fault that consist of aggressor line e_4 and victim lines e_5 and e_6 in the Fig.2 as an example to show the backtrack procedure. Suppose the aggressor line e_4 has a transition of 0 to 1. The objective values of lines e_4, e_5 and e_6 should all be 0 when the first vector S_1 is applied to the primary inputs (x_1, x_2, x_3, x_4, x_5, x_6) of the circuit.

(a) Firstly, the cone with output e_4 consists of lines x_1, x_2, x_3, e_1 and e_4. The value of e_4 is needed to be 0 only if x_1=0 and e_1=0. The e_1=0 implicates that at least one of the x_2 and x_3 should be 0.

(b) Secondly, the cone with output e_5 consists of lines x_3, x_4, e_2 and e_5. The value of e_5 is needed to be 0 only if the e_2=1. This implicates that at least one of the x_3 and x_4 should be 1.

(c) Thirdly, the cone with output e_6 consists of lines x_3, x_4, x_5, x_6, e_2, e_3 and e_6. The value of e_6 is needed to be 0 if at least one of the lines e_2 and e_3 is 0. Randomly select a path from the primary inputs to the line e_6. For instance, the path x_6–e_3–e_6 is chosen. The e_3 is 0 only if the x_5=1 and e_6=1.

The circuit input vectors that satisfy the above three aspects (a), (b) and (c) can be obtained. For example, (x_1 x_2 x_3 x_4 x_5 x_6)=(0 0 0 1 1 1). This vector can be as one of the first vectors S_1.

In the Step 6 of the Algorithm 1, the second vector S_2 is produced by using the operations of selecting paths, forward tracing, and backward tracing, etc. Here, we still take the crosstalk fault that consists of aggressor line e_4 and victim lines e_5 and e_6 in the Fig.2 as an example. Here the aggressor line e_4 has a transition of 0 to 1. The objective values of the lines e_4, e_5 and e_6 should be 1, 0 and 0 respectively when the second vector S_2 is applied to the primary inputs (x_1, x_2, x_3, x_4, x_5, x_6) of the normal circuit.

(d) Locate the lines e_4 and e_5 into the cone that its output is y_1, the cone consists of nodes x_1, x_2, x_3, x_4, e_1, e_2, e_4, e_5, and y_1. Select the sensitizing path x_2–e_1–e_4–y_1 for the line e_4, select the sensitizing path x_3–e_2–e_5–y_1 for the line e_5, and perform forward and backward tracing.

(e) Locate the line e_6 into the cone that its output is y_2, the cone consists of nodes x_3, x_4, x_5, x_6, e_2, e_3, e_6 and y_2. Select the sensitizing path x_6–e_3–e_6–y_2 for the line e_6, and carry out the forward tracing and backward tracing.

The circuit input vectors that satisfy the above two aspects (d) and (e) can be obtained. For example, (x_1 x_2 x_3 x_4 x_5 x_6)=(1 0 1 0 1 1). This vector can be as one of the second vectors S_2, because it can propagate the effect of the crosstalk fault to at least one of circuit primary outputs.

In the Step 7 of the Algorithm 1, for the crosstalk fault that consists of aggressor line e_4 and victim lines e_5 and e_6 in the Fig.2, the test vector S=(S_1, S_2)=((0 0 0 1 1 1), (1 0 1 0 1 1)). Apply the vectors S_1 and S_2 to the circuit primary inputs sequentially. If the circuit outputs are (y_1 y_2)=(1 1) for the S_1, and the (y_1 y_2)=(1 1) for the S_2, then there is not crosstalk in the circuit. If the circuit outputs are (y_1 y_2)=(1 1) for the S_1, and the (y_1 y_2)=(0 0) for the S_2, then there is the crosstalk in the circuit. Thus, the vector S can detect the crosstalk fault.

4. EXPERIMENTAL RESULTS

We have implemented the test vector generation approach in this paper in C++ language for the crosstalk faults with multiple victim lines, and have carried out a lot of experiments for digital circuits such as the ISCAS'85 benchmark circuits. These experiments are performed on a personal computer with 3.0GHz and 256M main memory. The numbers of signal lines in these ISCAS'85 benchmark are shown in the Table 1.

Table 1 The results for ISCAS'85 benchmark circuits.

Circuit	Inputs	Outputs	Lines	Twofc	Threefc
C432	36	7	432	100%	100%
C499	41	32	499	100%	100%
C880	60	26	880	99%	97%
C1355	41	32	1355	100%	98%
C1908	33	25	1980	100%	100%
C2670	233	140	2670	100%	96%
C6288	32	32	6288	98%	97%
C7552	207	108	7552	97%	95%

In the Table 1, the column "Circuit" denotes the names of benchmark circuits. The columns "Inputs" and "Outputs" show the numbers of primary inputs and primary outputs in the circuits, respectively. The column "Lines" denotes the total number of signal lines in a circuit.

In these experiments, we randomly choose 400 crosstalk faults with two victim lines and with three victim lines respectively from the ISCAS'85 benchmark circuits. The Algorithm 1 is used to generate the test vectors of these crosstalk faults being chosen. The steps in the Algorithm 1 are implemented by the activation of crosstalk faults, the partitions of circuit cones, the selections of sensitizing paths, the propagation of fault effects, forward and backtrack tracing operations, etc.

The experimental results are shown in Table 1. The column "Twofc" show the fault coverage for the crosstalk faults with two victim lines. The column "Threefc" show the fault coverage for the crosstalk faults with three victim lines. The fault coverage (F_C) is calculated by the following equation $Fc=(N_0/N_1)\times 100\%$. The N_1 is the total number of crosstalk faults considered, the N_0 is the number of crosstalk faults detected by using the Algorithm 1.

The experimental results in the Table 1 show that the Algorithm 1 in this paper can produce the test vectors of the crosstalk faults with multiple victim lines. For some circuits such as the C432, C499 and C1908, the 100% fault coverage can be obtained.

For some circuits such as C880, C1355, C6288 and C7552, the high fault coverage that is more than 90% can also be obtained. In these experiments, for the crosstalk faults being not testable, i.e. their test vectors do not exist, the Algorithm 1 is still used to produce the test vectors for the crosstalk faults, i.e. we do not judge the faults being testable or not testable before the Algorithm1 is performed.

These experiments also show that a portion of the time needed in the Algorithm 1 is due to the forward tracing and backtrack tracing operations. This portion of the time needed can be decreased by using the partition of circuit cones in Algorithm 1. Besides, another portion of the time needed in the Algorithm 1 is due to the sensitizing path selection. The total computation time needed can be cut down when the suitable paths are chosen. Therefore, it is valuable to investigated for the method that can obtain the most suitable paths for the test generation of crosstalk faults.

5. CONCLUSIONS

The crosstalk effects are induced between the adjacent signal lines owing to both the scaling of VLSI circuit and the increase in switching speed. It is necessary to carry out test for the crosstalk fault because the crosstalk may induce pulses and delay. In this paper, the test for the crosstalk faults with multiple victim lines is investigated, a test vector generation approach for the detection of this crosstalk faults is presented. Further work needs to be done in the future such as the accurate analysis of the coupling effects among the aggressor line and the multiple victim lines.

ACKNOWLEDGEMENTS

This work were supported by National Natural Science Foundation of China (No.61072028), the Project of Department of Education of Guangdong Province (No.2012KJCX 0040), Guangdong Province & Chinese Ministry of Education Cooperation Project of Industry, Education and Academy (No.2009B090300339).

REFERENCES:

[1] U.Gupta, N.Ranganathan.“A utilitarian approach to variation aware delay, power, and crosstalk noise optimization”. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol.19, no.9, 2011, pp.1723-1726.

[2] S.Sayil, V.K.Boorla. “Single event crosstalk prediction in nanometer technologies”. *Analog Integrated Circuits and Signal Processing*, vol.72, no.1, 2012, pp.205-214.

[3] V.Ferragina, N.Ghittori, G.Torelli, G.Boselli, G.Trucco. “Analysis and measurement of crosstalk effects on mixed-signal CMOS ICs with different mounting technologies”. *IEEE Trans. on Instrumentation and Measurement*, vol.59, no.8, 2010, pp.2015-2025.

[4] S.Roy, A.Dounavis. “Closed-form delay and crosstalk models for RLC on-chip interconnects using a matrix rational approximation”. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.28, no.10, 2009, pp.1481-1492.

[5] B.Jongsun, B.Sanghyeon, P.Sungju. “Characterizing the capacitive crosstalk in SRAM cells using negative bit-line voltage stress”. *IEEE Trans. on Instrumentation and Measurement*, vol.61, no.12, 2012, pp.3259-3272.

[6] F.D.Mbairi, W.P.Siebert, H.Hesselbom. “On the problem of using guard traces for high frequency differential lines crosstalk reduction”. *IEEE Trans. on Components and Packaging Technologies*, vol.30, no.1, 2007, pp.67-74.

[7] C.Sunghoon, K.Taejin, K.Sungho. “ATPG-XP: test generation for maximal crosstalk-induced faults”. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.28, no.9, 2009, pp.1401-1413.

[8] S.Hasan, A.K.Palit, W.Anheier. “Test pattern generation and compaction for crosstalk induced glitches and delay faults”. *Proceedings of International Conference on VLSI Design*, Bangalore (India), Jan.03-07, 2010, pp.345-350.

[9] L.Jianxun, W.B.Jone, S.R.Das. “Crosstalk test pattern *generation* for dynamic programmable logic arrays”. *IEEE Trans. on Instrumentation and Measurement*, vol.55, no.4, 2006, pp.1288-1302.

[10] S.Jaehoon, H.Juhee, Y.Hyunbean, J.Taejin, P.Sungju. “Highly compact interconnect test patterns for crosstalk and static faults”. *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol.56, no.5, 2009, pp.419-423.

[11] T.Bengtsson, S.Kumar, R.J.Ubar, A.Jutman. “Test methods for crosstalk-induced delay and glitch faults in network-on-chip interconnects implementing asynchronous communication protocols”. *IET Computers & Digital Techniques*, vol.2, no.6, 2008, pp.445-460.

[12] L.K.Min, L.Chung-Len, S.Chauchin, J.E.Chen. “A unified detection scheme for crosstalk effects in interconnection bus”. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol.17, no.2, 2009, pp.306-311.

[13] T.Rudnicki, T.Garbolino, K.Gucwa. “Effective BIST for crosstalk faults in interconnects”. *Proceedings of IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, Liberec(Czech Republic), April 15-17, 2009, pp.164-169.

[14] W.K.Assadi, S.Kakarla. “A BIST technique for crosstalk noise detection in FPGAs”. *Proceedings of IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, Boston(USA), Oct.01-03, 2008, pp.167-175.

[15] Y.J.Der, L.J.Fu, T.T.Wei. “Testing crosstalk faults of data and address buses in embedded RAMs”. *Proceedings of International Symposium on VLSI Design Automation and Test*, Hsinchu (Taiwan), April 25-27, 2007, pp.1-4.

[16] H.Y.Jen, L.J.Fu, C.C.Wei. “Post-bond test techniques for TSVs with crosstalk faults in 3D Ics”. *Proceedings of International Symposium on VLSI Design Automation and Test*, Hsinchu (Taiwan), April 23-25, 2012, pp.1-4.

[17] A.Sanyal, K.Ganeshpure, S.Kundu. “Test pattern *generation* for multiple aggressor crosstalk effects considering gate leakage loading in presence of gate delay”. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol.20, no.3, 2012, pp.424-436.

[18] D.Gope, D.M.Walker. “Maximizing crosstalk-induced slowdown during path delay test”. *Proceedings of International Conference on Computer Design*, Montreal(Canada), Sept.30-Oct.03, 2012, pp.159-166.

[19] K.Ganeshpure, S.Kundu. “On ATPG for multiple aggressor crosstalk faults”. *IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems*, vol.29, no.5, 2010, pp.774-787.