A Delay Testing considering Sub-Aggressor Effects

Min Joo Lee
Department of Electrical and Electronic Engineering,
Yonsei University
Seoul, Korea
mansemin@soc.yonsei.ac.kr

Sungho Kang
Department of Electrical and Electronic Engineering,
Yonsei University
Seoul, Korea
shkang@yonsei.ac.kr

Abstract - *Crosstalk issues in deep sub-micron (DSM) causes severe design validation and test problems. Therefore Crosstalk issues should be tested for high speed circuits or large circuits. This paper proposes an aggressor maximization algorithm to allocate worst case noise or delay to the on-path based on robust delay test generation method. In proposed algorithm we focus on the effect when the off-paths have a glitch. Experimental results in ITC'99 benchmark circuits showed that the proposed technique should be applied to circuits regardless of complexity. Our technique also allows considering any number of coupling capacitors along the target path and off-paths..*

Keywords: sub-aggressor, multiple aggressors, crosstalk, robust, ATPG.

1 Introduction

With dense interconnect, low supply voltage, fast clock frequency and large coupling-to-ground capacitance ratio, crosstalk noises are becoming perilous to ignore in the deep submicron era[1]. Various crosstalk analysis models were proposed in [1], [2], [3], [4], [5], [6], [7]. The simplest approaches to the problem make use of the single crosstalk fault model[4], [7]. In these models, a single parasitic capacitance is supposed to be present in the circuit. The difference between single crosstalk fault models is how to allocate worst crosstalk induced delay or noise on the victim line.

Definition 1. A Victim line is called the line affected by another lines if a parasitic capacitance is present between those lines.

Definition 2. An aggressor line is called the line affecting another line(s) if a parasitic capacitance is present between those lines.

To find allocation path from victim line to any primary output, researchers choose a path using a path delay fault model [1][3][4][5]. To select a critical path, by making a library using static timing analysis or by a path delay fault consider only a single coupling capacitor.

Multiple aggressors can be utilized to allocate more crosstalk induced delay or noise. The problem is how to decide multiple aggressors or how to generate test patterns for multiple aggressors. The test generation for arbitrary occurrence of coupling capacitors has not dealt yet. In this paper, we propose an ATPG (Automated Test Pattern Generator) method for dealing specific multiple aggressors in the circuit. The proposed method divide the situation by the cone-based multiple aggressors.

In the proposed approach, for multiple aggressors, we define a set of sub-aggressors and a set of sub-victims and reduce the problem of finding test patterns for the given crosstalk target to that of finding the corresponding sub-aggressors.

Definition 3. Sub-aggressor is defined as the line affecting the sub-victim lines.

Definition 4. Sub-victim is defined as the line affected by sub-aggressor(s).

When multiple coupling capacitors exist in the circuit, we can divide the coupling capacitor into a dominate coupling capacitor and other coupling capacitors by their coupling capacitor values. The coupling capacitors, except a dominant coupling capacitor, can be considered as sub-coupling capacitors.

This paper is organized as follows. Section 2 gives a background of motivation of multiple aggressors when the on-path gate is the sub-victim's. In Section 3, we show the proposed an algorithm to maximize crosstalk induced delay or noise to the on-path. Section 4 shows the experimental results. Section 5 concludes this paper with several observations and suggestions for future research.

2 Motivation

In general, the quality of the ATPG is decided by maximal noises or delay induced test vectors. To generate maximal noises or delay, they proposed an algorithm for testing crosstalk induced delay faults [6]. They chose a critical path for activating crosstalk induced delay faults, and then generating the aggressors set which can be affect on-path. The problem is when there is no single significant crosstalk fault site which makes crosstalk induced delay faults. By basing this on a path delay model, the selected path can be a fault path although the selected path is not affected by crosstalk induced delay or noise. To avoid this problem, we assumed we had a dominant coupling capacitor in the circuit as for the single aggressor test methods [4], [7].

Based on path delay model, we cannot consider the effect of the off-path except [2]. They showed the increase of noise volume compared to without off-path considerations. Although they first considered off-path effects, the proposed method can not apply to the APTG due to lots of exception situations and impractical approach.

3 The aggressor maximization

In proposed method, we chose the first victim line by a significant coupling capacitor, which can be part of a critical path or not. Even the selected path is not a critical path by the traditional method, it can be an authentic critical path with off-path effects in the proposed algorithm. We considered multiple aggressors not lying on-path but multiple aggressors on off-path inputs. As discussed in [2], we now consider load capacitance and driving capacitance of gate for maximum or worst noise case. To apply this to the ATPG, we first searched for new sub-aggressors for the maximize crosstalk induced delay or noise to on-path

3.1 Target identification

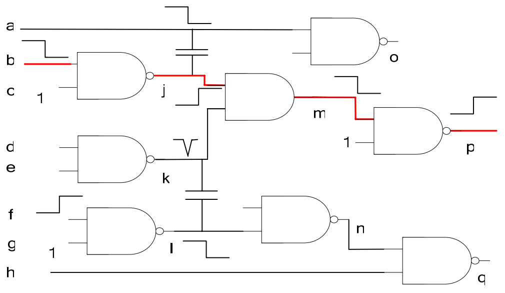

Figure 1. An example of sub-aggressors

We take up an example circuit to show new sub aggressors. In Figure 1, the on-path is marked as a red line. When victim line is decided on, the aggressor line is from a to o. Suppose that the off-path input k has the static value 1 for the glitch. In this condition, we can define a new sub-aggressor line l. If the line l has falling transition, the line k will have a glitch, and it will cause more noise to the on-path. When considering sub-aggressors, the problems are how many sub-aggressors have to be accounted for and how to find appropriate sub-aggressors set. Also, there can be another problem as explained in Figure 2.

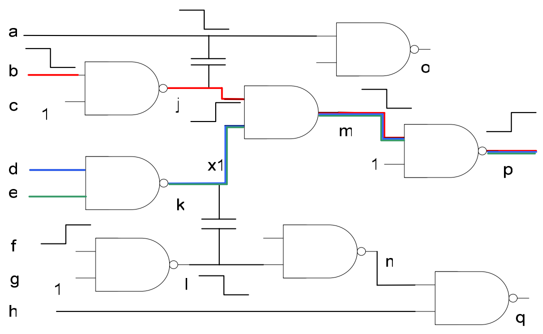

Figure 2. Exception condition in sub-aggressors

In Figure2, we supposed that off-path k just has the value x1, which can be a falling transition or static value 1. Assume the k's state is a rising transition, and sub-aggressor l a has falling transition, it causes a speed-down delay fault. With these conditions, one or more crosstalk-induced delay faults may occur in the circuit between line k and line l, the on-path can be d-k-m-p or e-k-m-p or b-j-m-p the same as for only one coupling capacitor in the circuit. Even with generated test vectors for the on-path b-j-m-p, we can not distinguish generated test vectors for the on-path d-k-m-p. During the ATPG process, they should have only one critical path including primary victim line. However, the problem that sub-victim line has a transition ; we cannot guarantee which crosstalk-induced delay faults are detected from the generated test vectors. To eliminate this problem, sub-victim lines should be static values such as S1, or S0 in this paper

3.2 Considering Cone-based multiple sub-aggressors

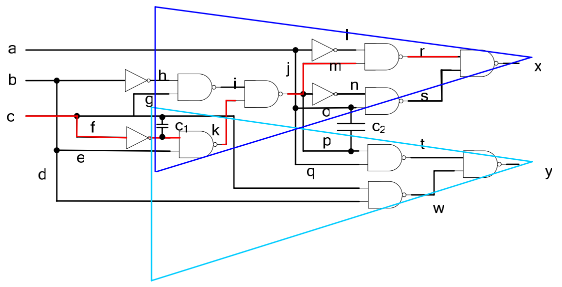

Figure 3. Selected Cone in example circuit

In an example circuit of figure 3, we can think about two cones. There is the possibility of each cone having many sub-aggressors. The proposed approach will divided sub-aggressors by a cone. The set of sub-aggressors are called A_1, A_2 ... A_N. (N is the number of cones) In figure 3, we have sub-aggressors set A_1 and A_2. To find candidate sub-aggressors in the A_1 sub-aggressors, suppose we have a coupling capacitor, C_1 in an example circuits. First, we decided which cone used for searching candidate sub-aggressors. In this example, we show when we choose A_1. Then we should decide on the propagation path from victim line to a primary output in the first cone. We prefer to use selective method through gates as much as possible for maximizing on-path gates because we settle off-inputs as sub-victims. When we have more precise timing information, the on-path can be changed without conflicting with the proposed method. The detailed method will be discussed in the next section. Using on-path c-f-k-j-l-r-x with another coupling capacitor C_2, the on-path should be c-f-k-n-s-x because of the constraint of first cone. Actually, if we choose on-path c-f-k-n-s-x for the first time, there is no need to change the on-path, even if we have to choose other on-path along the cone. For the ATPG, the change of on-path can be handled by allocating appropriate sub-aggressors. The problem is overlapping of handling multiple coupling capacitors i.e. C_1 and C_2. There are two choices of cones. If there is no way to find all sub-aggressors for the on-path at first, we should consider all possibilities for the worst crosstalk-induced delay or noise. For example, even if C_1 and C_2 paths the same levels and same types of gates to the primary output, they have different on-path gates, it means that those paths will have different sub-aggressors for maximizing crosstalk-induced delay or noise.

3.3 The aggressor maximization algorithm

```
Generate cone;
If (cone is generated)
  {
    Select one of $A_n$;
    Generate candidate on-path sets
    Select candidate on-path set
    Save PI vectors as Vsta
    Search sub-aggressors
    Start from first sub-aggressors
  While (conflict for activating another sub-aggressor)
   {
    Restore Vsta with activated PI vectors
    Start activating process from another sub-aggressor
    if (conflict when find signal values)
      {
        Restore Vsta with activated PI vectors
        Start activating process from another sub-aggressor
      }
    While (end of sub-aggressor is not reached)
      {
       Do same process
      }
     if (end of sub-aggressor for a on-path)
      {
       Find the worst case input vector PI
      }
     While (end of on-path for a cone is not reached)
       {
         Repeat
       }
     If (end of on-path for a cone is reached)
        {
          Done with activating process with a cone
          Select another cone
        }
     }
  Find the worst case input vector PI, for all on-path
```

Figure 4. The proposed algotirhm

In this section we discuss the aggressor maximization algorithm. We assumed that selected on-path can sensitize and propagate to any primary outputs. For propagating process, we considered as many different paths as possible due to the uncertainty of candidate sub-aggressors. Each candidate propagated path catalogs by order of primary outputs. When we generated cones for on-path, the proposed approach started to find candidate sub-aggressors. If we know coupling capacitors' value, we can easily decide priority order for activation. Without knowing coupling capacitors' value in the circuit, in this paper, we allocated sub-aggressors in order of location near the primary input of on-path. During allocating sub-aggressor value, it makes conflict for other sub-aggressors or not. To find truly worst noise or delay, we can save each PI values which activate a sub-aggressor. Because we wanted to test vectors which activate maximal noise or delay to on-path, if specified signal value has no conflict with other activation process, we use these values. When conflict occurs, we record conflict situation, and save primary inputs, then continue the activating process for the next sub-aggressor. After searching process for all sub-aggressor, we can find the appropriate sub-aggressors set for on-path. The proposed algorithm is shown in the following pseudo code:

4 Experimental Results

To evaluate the proposed ATPG method, we implemented the ATPG algorithm in C code. In the experiment, we used on robust test pattern generation method to allocate sub-aggressors in the circuit. When off-path input (sub-victim) has a static value, the other line (sub-aggressor) can has a transition value without conflicting with on-path propagation. In these experiments, we assumed there was a positive glitch in the sub-victim line. Due to a lack of layout information, we simulate all possible paths can be connected with a coupling capacitor. In the experiment, miltiple aggressors baesd on robust test generation results on ITC'99 benchmark circuits are shown in Table 1.

Circuits Names	# CVS	Testagble Faults			Fault coverage	CPU Time (sec)
		# TFs	#UTFs	#Afs		
b01	720	590	130	0	81.94%	0.15
b02	144	108	36	0	75%	0.02
b03	77312	37774	14662	24906	40.24%	102.98
b04	179620	54424	107052	18144	30.30%	119.33
b06	2640	2224	432	24	84.24%	0.49
b08	179620	54424	107052	18144	30.30%	119.33
b09	45048	9330	25273	104452	20.71%	150.99
b10	57204	25878	25292	6034	45.24%	43.79
b13	435836	353959	65736	16141	81.21%	1021.65

Table 1. EXpermental result on ITC'99

Column 1 shows the names of the circuits. Column 2 shows the information about candidate victim paths in the circuits (#CVs). The results about test generation are shown in coumn 3, where the numbers of testable faults (#TFs), the numbers of untestable faults (#UTFs), and the numbers of aborted faults(#AFs). The fault coverage (FC) of these circuits is shown in column 4. The last clumn shows CPU tim in seconds. The CPU time incluses time of selecting candidate paths and multiple aggressors based on robust test patterns generation for the target faults.

It is shown that the numbers of candidate target increase to several millions for larger benchmark circuits. The proposed approach perfectly matches with larger circuits. Although the CPU times also increase by the increasement of benchmark circuits' size, the detectable faults also increase. It can be seen that out ATPG program is efficeient for crosstalk-induced delay faluts.

5 Conclusion

We proposed the algorithm for maximization of the crosstalk-induced noise or delay. Using path delay model, we have to have off-path inputs to the on-path. Since the effects of off-path input are ignored, we can not allocate the maximum crosstalk-induced noise or delay. Even this paper, we can not consider all conditions of the off-path due to complexity of ATPG. For the activation victim line, we have to propagate the value of the victim-line to any primary output. During this process, we use the robust test pattern generation for crosstalk ATPG. It automatically selects the sub-aggressors for maximizing crosstalk induced delay or noise. Experiment results on ITC'99 circuits showed that the proposed technique can be applied to circuits of reasonable sizes within acceptable time.

Future work intends to extend applications to more off-path conditions when off-path does not have static values to increase test fault coverage. When we have access the layout information and timing information of the circuit, we will modify the proposed maximization algorithm can cooperating with layout information and timing information of the circuit, to increase test pattern accuracy.

References

[1] Xiaoliang Bai, Sujit Dey, "HyAC: A Hybrid Structural SAT Based ATPG for crosstalk," *Proc. International Test Conference*, pp. 112-121, 2003.

[2] M. Favalli, ""Victim gate" crosstalk fault model," *Proc. Of VLSI Defect and Fault Tolerance*, 2004.

[3] A. Rubio, N. Itazaki, X. Xu and K. Kinoshita, " An Approach to the Analysis and Detection of Crosstalk Faults in Digital VLSI Circuits" *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems*, Vol.13, No.3, pp. 387-394, March 1994.

[4] W. Y. Cehn, S. K. Gupta and M.A. Breuer, "Test generation in VLSI circuits for crosstalk noise," *Proc. International Test Conference*, PP. 379~385, 2001.

[5] R. Jundu and R.D. Blanton, "Timed Test Generation for Crosstalk Switch Failures in Domino CMOS Circuits," *Proc. Of VLSI Test Symposium*, pp. 379~385,2001.

[6] Aniket and R. Arunachalam, "A Novel Algorithm for Testing Crosstalk Induced Delay Faults in VLSI Circuits," *Proc. Of International Conference on Embedded Systems Design*, pp. 479-484, 2005.

[7] K. T. Lee, C. Nordquist and J. A. Abraham, "Automatic test pattern generation for crosstalk glitches in digital circuits," *Proc. Of VLSI Test Symposium*, pp.34-29, 1998.