Abstract—In this paper, we propose a new test-generation method for delay faults considering crosstalk-induced delay effects, based on a conventional delay automatic-test-pattern-generation (ATPG) technique in order to reduce the complexity of previous ATPG algorithms and to consider multiple-aggressor crosstalk faults to maximize the noise of the victim line. Since the proposed ATPG for crosstalk-induced delay faults uses physical and timing information, it can reduce the search space of the backward implication of the aggressor’s constraints, and it is helpful for reducing the ATPG time cost compared to previous works. In addition, since the proposed technique targets the critical path for the original delay test as the victim lines, it can improve test effectiveness of delay testing. Experimental results demonstrate the effectiveness of the proposed method.

Index Terms—Automatic test-pattern generation (ATPG), crosstalk delay faults, fault pruning, path-delay faults, timing analysis.

I. INTRODUCTION

As feature sizes decrease in deep-submicrometer circuit designs, coupling capacitance dominates total capacitance, and crosstalk faults in the logic circuit become significant and responsible for major timing violation and signal integrity issues [1]. In local interconnections of the logic circuit, capacitive crosstalk is the dominant source of signal-integrity-related failures. Crosstalk faults result from parasitic coupling between adjacent signal nets since this coupling injects extra ac current into a coupled net.

Crosstalk faults can be classified into two types: crosstalk-induced glitches and crosstalk-induced delays. Crosstalk-induced glitch [2] occurs when a victim line is intended to be at a stable state and results in an unwanted noise pulse on the net. A crosstalk-induced delay [3] occurs when noise is injected on a net when a signal transitions and results in a change in the delay of the net. In most circuits, crosstalk-induced delay, particularly slowdown delay, leads to the chip failure more so than the crosstalk-induced glitch [1], [3].

In current trends in integrated-circuit design, it is impossible to eliminate errors caused by crosstalk because of stringent area and performance requirements in recent designs. These noises due to crosstalk could be eliminated by resizing drivers, shielding interconnect techniques, rerouting signals, and repeater insertion techniques. However, these redesign techniques using the analysis of crosstalk-induced noise may be very expensive in terms of time and design efforts. In addition, since low process variation cannot be expected for all cases, these redesign techniques cannot guarantee a high noise immunity of the circuits. Therefore, automatic test-pattern generation (ATPG) for crosstalk faults is an important method to verify and test that a chip can meet certain performance requirements.

Among previous ATPG methods for crosstalk faults, some consider noise effects from the logic errors caused by crosstalk-induced glitch faults [4]–[7]. However, testing for crosstalk-induced effects using delays has recently received more attention [8]–[16]. Since the pattern generation for crosstalk-induced delay faults requires timing information, reducing the high complexity of the ATPG process is a major issue of previous test-generation methods. For this purpose, many researchers have developed the deterministic ATPG technique [9]–[13] for single crosstalk-induced delay faults based on the path-oriented decision-making (PODEM) algorithm [17]. However, such techniques still suffer from the computational complexity of their algorithm. In addition, since their algorithms considered only a single pair of a victim line and an aggressor line, they cannot propagate a maximal crosstalk-induced effect along a critical victim path, on which the effect is more likely to cause delay test failure.

To alleviate these problems, a genetic-algorithm-based pattern-generation technique has been proposed in [15]. To consider maximal crosstalk-induced delays, the technique in [15] used a critical victim path and a set of aggressor lines interacting with the line. However, the technique was time-consuming because it was based on the genetic algorithm and did not deal with the timing information efficiently. In [16], a novel ATPG technique to generate patterns that will excite the worst-case delay at the victim line by switching the maximal set of aggressors was proposed. This technique could not resolve the time-complexity problem and the usage of timing information.

Therefore, in this paper, we propose a new test-generation method for delay faults considering crosstalk-induced delay effects, based on a conventional delay ATPG technique in order to reduce the complexity of the previous ATPG algorithm and to consider crosstalk faults induced by multiple aggressors so as to maximize the noise of the victim line. Since the proposed ATPG uses the test patterns obtained from the conventional path-delay ATPG, the proposed ATPG can reduce the search space of the backward implication of the aggressor’s constraints, and it is...
helpful for reducing the ATPG time cost compared to previous works. In addition, since the proposed technique targets the critical path for the original delay test as the victim lines, it can improve the effectiveness of delay testing.

For a sensitizable victim path, a new algorithm is proposed, which appropriately activates its associated aggressor–victim pairs for the maximization of crosstalk. The proposed algorithm uses parasitic information, such as coupling capacitance between a node of the victim and an aggressor, and timing information, such as static timing window and the crosstalk-induced noise delay model. Using the parasitic and timing information, the proposed ATPG can reduce many false aggressors and handle multiple possible aggressors coupled to a victim lying along a path. Moreover, the proposed ATPG activates the aggressors in the best possible way to induce maximum crosstalk slowdown along a path. To efficiently use the timing information, we propose a new timing refinement scheme with the delay test pattern of the given victim path. It helps to more precisely identify the aggressors coupled to the victim path.

Generally, deterministic test patterns for path-delay faults have typically only between 1% and 5% of the bits specified, and don’t-care values are filled with 0s and 1s. Therefore, using these don’t-care values in the test cubes for path-delay faults, the conventional delay test patterns can be updated to efficiently detect crosstalk-induced delay defects. From this fact, the goal of the proposed test generation is to optimize the conventional delay test patterns in order to maximize noises of victim paths.

The rest of this paper is organized as follows. In Section II, we explain the background of the delay-testing methodology and the crosstalk-induced noise. Section III explains how to prune and select crosstalk-induced delay faults, while the test-generation method for the proposed crosstalk-induced delay faults is presented in Section IV. To validate the efficiency of the proposed ATPG, experimental results are demonstrated in Section V. Section VI summarizes the contributions of this paper.

II. BACKGROUNDS

A. General Delay Test

A delay test in a logic circuit with full-scan chains can be detected only by applying a sequence of two test patterns. The first pattern, known as an initialization pattern, sets up the initial condition in a circuit so that the fault slow-to-rise or slow-to-fall signal at the gate input or output can affect the primary or scan output of the circuit. The second pattern, known as the transition or activation pattern, propagates the effect of the activated transition to the primary or scan output of the circuit.

Many researchers have proposed a variety of methods and fault models for detecting delay defects. The fault models are classified into mainly three groups: 1) the transition fault [18]; 2) the path-delay fault [19]; and 3) the segment delay fault [20]. The path-delay-fault model considers the cumulative effect of the delays along a specific combinational path in the circuit. If the cumulative delay in a faulty circuit exceeds the clock period for the path, then the test pattern that can exercise this path will fail in the chip. Since the targeted victim lines of the proposed pattern generation are critical or long paths that are vulnerable to failure on the chip due to delay defects, the path-delay-fault model is used in the proposed method. In addition, since the path-delay model can detect small distributed delays, it can provide a mechanism for monitoring crosstalk-induced delay defects that have subtle but deleterious effects on paths that are near the timing limit for the circuit.

Many test-pattern-generation methods for path-delay faults have been presented in a standard scan environment [21]–[24]. As mentioned earlier, since the initial vector of every two vector pair delay tests can be scanned into the circuit’s internal scan flip-flops, all that was required to generate the tests was to trace the test requirements through two clock cycles. Therefore, the functional justification [18] is required to efficiently generate test patterns for path-delay faults. In the proposed method, the path-delay test-pattern generator for full-scan design using functional justification presented in [25] is used to efficiently generate test patterns for path-delay faults.

B. Crosstalk-Induced Delay Effects

In a circuit, the majority of on-chip wiring is composed of locally short lines that have a maximum length of 1–3 mm. Such short lines are driven by small devices with a large effective impedance $Z_{\text{driver}}$ compared to the line impedance $Z_0$. Therefore, the interconnect lines appear as lumped capacitive loads. Because of the small wiring cross sections in on-chip local lines, the local lines have a very high resistance $R_{\text{line}}$. The short local lines exhibit an $RC$-like behavior and are represented by $RC$ circuits. Therefore, a lumped $RC$ circuit model has been used for the analysis and estimation of the crosstalk-induced delay noise in local lines. In the proposed method, the $RC$ interconnection model is used to reduce false-aggressor lines and to effectively generate test patterns for crosstalk-induced delay faults.

The traditional delay formulation for the $RC$ interconnection model of short lines is shown in the following [26]:

\[
\text{Delay} = Z_{\text{driver}} \cdot (C_L + C_{\text{line}}) + \frac{(R_{\text{line}} \cdot C_{\text{line}})}{2} \quad (1)
\]

where $R_{\text{line}}$ is the resistance of the line and $C_L$ and $C_{\text{line}}$ represent the load and line capacitances, respectively. The delay equation (1) is the sum of the gate delay and interconnection delay.

Since short and dense lines have a very large capacitive coupling between adjacent lines, the crosstalk-induced delay is data pattern dependent. Depending on the direction of the voltage on the neighboring lines, the effective capacitance of the line, $C_{\text{line}}$, will be changed. For an even-propagation-mode pattern, the effective capacitance $C_{\text{line}}$ will be $C_{22} - C_{12}$. If there is no adjacent switching pattern, the effective capacitance will be $C_{22}$. In addition, for an odd-propagation-mode pattern, the effective capacitance will be $C_{\text{line}} = C_{22} + 2C_{12}$. The larger the capacitive coupling, the larger the delay variation on interconnection lines. An example of the pattern-dependent delay is shown for lines that have a resistance $R_{\text{line}} = 494$ $\Omega$/cm, self-capacitance $C_{22} = 1.73$ pF/cm, and coupling capacitance $C_{12} = 0.484$ pF/cm. As shown in this example, since the crosstalk-induced delay is maximized for
the odd-mode propagation, the constraint between the aggressor and the victim is that the switching direction between them must be opposite in order to generate test patterns for maximal slowdown delay faults.

If a transition that propagates along a critical path is affected by the crosstalk-induced slowdown delay, then the incorrect value may be captured at a primary output or a scan flip-flop by a capture clock in Fig. 1. Therefore, victim lines of the target crosstalk faults in the proposed method are the same critical paths targeted in conventional delay tests.

If the nominal delay of a victim node is defined as the delay calculated when there is no switching transition on an aggressor node, the crosstalk-induced delay is calculated using

\[ t_{\text{crosstalk}} = t_{\text{victim}} - t_{\text{norm}} \]  

where \( t_{\text{crosstalk}} \), \( t_{\text{victim}} \), and \( t_{\text{norm}} \) are the time delay caused by crosstalk-induced noise, the total delay of the victim line, and the nominal delay, respectively.

In a general logic circuit, we can suppose that the driving resistance of the aggressor is linear. Therefore, the crosstalk-induced delay \( t_{\text{crosstalk}} \) can be calculated using linear superposition where the noise induced by each aggressor is calculated. For this reason, to generate test patterns for maximal crosstalk-induced delay, all possible aggressors are activated in the generated delay test patterns. The variation of the crosstalk-induced delays according to the number of aggressors is shown in Fig. 2.

III. OVERALL PROCESS OF ATPG-XP

The proposed ATPG for crosstalk-induced delay faults, called ATPG-XP, consists of three main procedures: 1) conventional delay test-pattern generation; 2) crosstalk-induced fault pruning; and 3) test-pattern generation for crosstalk-induced faults. Fig. 3 shows the overall procedure of the proposed ATPG-XP.

As mentioned in Section II, once the crosstalk-induced delay can be calculated using linear superposition where the noise induced by each aggressor is calculated, all possible aggressors should be activated in the generated delay test patterns in order to generate test patterns for maximal crosstalk-induced delay.

To achieve this goal, we propose various methods to remove false crosstalk faults from all the suspects of possible aggressor lines. In this case, all lines, except the lines on the victim path, can be possible suspects for aggressor lines. However, a lot of lines are false aggressors because they cannot excite or test crosstalk-induced faults on the target victim path. The proposed pruning methods focused on removing those false-aggressor lines from initial possible suspects for a given victim path. It is obvious that the proposed method is not an optimal solution because the proposed method is based on the results of a conventional delay ATPG. However, for a generated test pattern for a delay fault, since the proposed ATPG can modify the test pattern to activate real aggressors as many as possible, the modified test patterns make crosstalk-induced delay errors on the victim path maximal.

First, the proposed ATPG generates a delay test pattern for a given victim path through the conventional delay ATPG engine [25]. This delay test pattern helps one to increase the ability of false-crosstalk-fault pruning and to reduce the complexity of the pattern generation for the proposed crosstalk faults. Note that false crosstalk faults can be defined if it is unexcitable or if it cannot be tested. Next, initial aggressor candidates are
Fig. 3. Overall procedure of the proposed ATPG.

generated for a given victim path, and then, false crosstalk faults are eliminated in the initial aggressor list using physical and timing information. In this paper, the aggressor candidates can be defined as the lines that have the opposite transition value against the victim lines because the slowdown delay will be occurring when a switching direction between a victim line and an aggressor line is opposite. In particular, for temporal pruning, we propose a new timing refinement scheme with a delay test pattern of the given victim path in order to more precisely identify the aggressor candidates coupled to the victim path. The final procedure of the proposed method is the test-pattern generation based on the original delay test pattern. Since this procedure uses unspecified values in the original delay test pattern and results of logic simulation with the delay test pattern of a given victim path, the test-generation algorithm is much simpler than previous methods for crosstalk-induced delay faults.

The details of each procedure will be explained in the subsequent sections.

IV. PRUNING FALSE AGGRESSORS

Since the proposed method uses delay test patterns obtained from the conventional delay ATPG, the victim paths in the target fault list of ATPG-XP are identical to the delay test. Even though there is one victim path in a fault list, the number of possible target faults is impractical to determine because all the nodes, except the victim nodes, can be aggressors, and the combination of possible aggressors increases exponentially according to the number of nodes in a given circuit. Therefore, to reduce the complexity of test-pattern generation for crosstalk-induced delay faults, methods to prune the false aggressors and to determine appropriate target faults are required. In ATPG-XP, two aggressor-pruning methods, namely, spatial pruning and temporal pruning, are used for selecting target faults.

Because of computational complexity, in the proposed method, any functional pruning is not performed separately. Instead, false-aggressor candidates that have no functional relationship with the victim path, which means that the aggressor line cannot satisfy the logical condition for causing crosstalk-induced slowdown, are eliminated during the test-pattern-generation procedure for crosstalk-induced slowdown faults.

A. Spatial Pruning

Intuitively, two adjacent lines in the layout are potential candidates for capacitive coupling. From this fact, spatial pruning can be performed. Only adjacent lines may have crosstalk coupling effects. Using commercial parasitic-extraction tools, coupling capacitances between any adjacent lines can be extracted. Any capacitive coupling effect across more than one metal line is insignificant in the modern process and is typically ignored.

However, it is very difficult to identify aggressor lines that have capacitive effects across one metal line through analyzing the physical layout of the circuit. Therefore, to efficiently select aggressor lines of a given victim line, the following criterion is used with coupling-capacitance values obtained from parasitic extraction:

\[ C_{\text{candidate}_i} \geq C_{\text{threshold}} \]  \( (3) \)

where \( C_{\text{candidate}_i} \) and \( C_{\text{threshold}} \) are the coupling capacitance between the candidate aggressor line \( i \) and the given victim line and the threshold coupling capacitance, respectively. Note that the threshold coupling capacitance is obtained by analyzing the coupling-capacitance values whose effects cause crosstalk-induced noise or is defined by users.

First, all the possible candidates related to a given victim path are identified, which are inserted into the aggressor candidate list of the given victim path in decreasing order of coupling-capacitance value \( C_{\text{candidate}_i} \). If the coupling-capacitance value in the candidate list does not satisfy criterion (3), then the aggressor candidates can be eliminated from the aggressor candidate list. Therefore, by using criterion (3), the number of potential aggressors can be reduced significantly.
B. Temporal Pruning

Static timing analysis (STA) computes the minimum and maximum values of arrival and transition times for rising and falling transitions at each circuit line. These calculated values are called timing windows of circuit lines. The crosstalk-induced slowdown delay is strongly related to the timing window of each line. If the timing window of the victim line overlaps the timing window of the aggressor line, then the victim line may be affected by the aggressor line, and crosstalk-induced noise may occur on the victim line. Otherwise, the victim line will have no crosstalk-induced noise. In this case, the functional information is ignored, but the timing information is included. Therefore, detecting the overlap of timing windows between coupled lines is required to reduce the false crosstalk faults in the aggressor candidate list obtained from spatial pruning. Previous test-pattern-generation methods for crosstalk-induced faults [11]–[14] use the STA-based temporal-pruning method to find overlaps of timing windows between coupling lines. Since the timing window obtained from the STA-based pruning method has a very pessimistic range, their pruning results are very pessimistic.

Initially, without considering the effects of crosstalk noises, timing windows are calculated from the STA method. Since STA computes the timing windows with unspecific input values, the computed timing windows represent the minimum and maximum values over the universe of all possible vectors. Therefore, it is a very pessimistic pruning technique to find overlaps of timing windows between coupling lines using the original timing window. In addition, since the timing window obtained from STA does not consider the effects of crosstalk-induced delay noises, some false-aggressor lines can be included in the aggressor candidate list.

The second step of the proposed temporal pruning is updating timing windows of a circuit with the generated test pattern of the victim path. Since previous methods did not have any specific information to propagate the crosstalk-induced errors and they only consider the minimum and maximum values of arrival and transition times for rising and falling transitions at each circuit line like traditional STA calculations, timing windows generated by previous methods [11]–[14] are very pessimistic and have very wide ranges. It leads to miss the real aggressor lines for the target victim path. Since the proposed ATPG uses the delay test patterns generated from the conventional delay ATPG, the ranges of initial timing windows are narrower when the input vectors are partially specified. For a given test pattern, the recalculated timing windows have narrower and more accurate ranges than those of previous works [11]–[14]. Note that, in general, delay test patterns are partially specified and include many don’t-care values. The recalculated timing windows with the delay test patterns lead to the identification of more false-aggressor lines. For example, a two-input AND gate is used to explain that the timing window becomes narrower when specified values are used. Fig. 4(a) shows the switching characteristics of a two-input AND gate. As shown in Fig. 4(a), the switching delay depends on the path of the switching signal. Fig. 4(b) shows the timing analysis with specified values. In Fig. 4(b), while the timing window of the output with unspecified values is \((\text{min. delay}, \text{max. delay}) = (0.085, 0.105)\), the timing window of the output can be changed to \((0.097, 0.097)\) when the inputs are specified. Therefore, the timing windows of a circuit are updated by using a delay test pattern of a given victim path.

The final step of the proposed temporal pruning is finding overlaps between the aggressor candidates and the victim path using the updated timing windows. Fig. 5 shows how to identify the overlaps of the updated timing windows. Note that the logic values use the values of two time frames and that timing window ranges are shown as \((\text{minimal arrival time}, \text{maximal arrival time})\) for a rising or falling transition. In addition, the propagation delay of AND gates in Fig. 5 uses the characteristics shown in Fig. 4(a).

As shown in Fig. 5(a), assuming that the victim path is D–I–K–N–Q and using the initial timing window calculated by a conventional STA, the number of timing windows that overlapped with the timing window of the victim path is 12. However, since the updated timing windows with delay test patterns are narrower and more accurate than the initial timing window with all unspecified values, many false-aggressor lines can be eliminated in the aggressor candidate list of the victim path. In an example shown in Fig. 5(b), after updating the timing windows with the delay test patterns of the given victim path (D–I–K–N–Q) and analyzing the results of the recalculated timing window, the number of potential aggressor lines becomes three (nodes A, B, and M). In this example, the number of false-aggressor lines eliminated through the proposed temporal pruning is nine, and the reduction ratio of search spaces for the proposed ATPG is 75%. In Fig. 5(b), after temporal pruning, the candidate lists of the target crosstalk faults are \{aggressor, victim\} = \{A, D\}, \{B, D\}, \{M, K\}, and \{M, N\}. As shown in this example, the proposed temporal pruning using the updated timing windows can significantly reduce the number of false-aggressor lines and increase the search time for a test-pattern-generation algorithm.

Applying the proposed spatial and temporal pruning might be insufficient to reduce false-aggressor candidates and search
spaces for the proposed ATPG. Just as the proposed temporal pruning can be used in timing analysis to eliminate false aggressors that are never responsible for the crosstalk-induced delay, functional pruning can be used in logic simulation to eliminate those signals that can never be responsible for noise problems because of their functional relationship.

The most straightforward approach to finding functionally false aggressor lines is to exhaustively simulate all input-vector pairs. This is rarely computationally feasible. Therefore, a variety of works have been performed using a binary decision diagram (BDD) or satisfiability (SAT) solver. However, these methods are computationally expensive. In addition, the techniques based on BDD or SAT are necessarily limited because many circuits do not have compact BDD representations or compact conjunctive normal form formulations.

To alleviate these problems, in the proposed method, any functional pruning is not performed separately. Instead, false-aggressor candidates that have no functional relationship with the victim path, which means that the aggressor line cannot satisfy the logical condition for causing crosstalk-induced slowdown, are eliminated during the test-pattern-generation procedure for crosstalk-induced slowdown faults. Since logic simulation is required for functional pruning, the functional-pruning method performed separately is a redundant process. In addition, during the selection of aggressor constraints and the backtracing of the proposed ATPG, functionally false aggressors can be eliminated automatically.

### V. TEST-PATTERN GENERATION FOR CROSSTALK-INDUCED DELAY FAULTS

After pruning the false-aggressor lines in the aggressor candidate list, the crosstalk fault models are generated as targets of the proposed ATPG. In order to generate test patterns to maximize crosstalk-induced delays for the given paths, a crosstalk path-delay fault (XPDF) is defined as a critical path and a set of aggressor candidates coupled to the critical path. Critical paths refer to paths whose delay is longer than a given percentage of the longest propagation delays in the circuit, and are the same as general path-delay faults.

For example, consider the victim path $p$ (D→K→M→Q→X→Y) with a rising transition at the input of the victim path shown in Fig. 6. The transition signals of other lines are also shown. Note that the switching characteristics of all gates in this example are the same as that of the two-input AND gate shown in Fig. 4. Since the timing windows are updated with the given delay test pattern for the victim path, many false-aggressor lines are already eliminated through temporal pruning. Therefore, in this example, the number of remaining aggressor candidates is eight (nodes F, G, I, J, N, O, W, and Z). Through the analysis of the timing windows of the aggressor candidates, we can determine that line N can be coupled with line K on victim path $p$. In a similar manner, other pairs of coupled lines that can cause a crosstalk-induced slowdown delay can be determined. Thus, the final XPDF fault is defined as \{the victim path, a set of aggressor candidates coupled to the victim line on

<table>
<thead>
<tr>
<th><img src="image1.png" alt="Diagram" /></th>
<th><img src="image2.png" alt="Diagram" /></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(b)</td>
</tr>
</tbody>
</table>

Fig. 5. Example of the proposed temporal pruning. (a) Use of conventional STA. (b) Use of STA with delay test patterns.
Fig. 6. Example of an XPDF fault model.

\[ p \ (\text{victim, aggressor, coupling capacitance between them}) = \{p, (D, F, C_{DF}), (D, G, C_{DG}), (D, I, C_{DI}), (D, J, C_{DJ}), (K, N, C_{KN}), (K, O, C_{KO}), (Q, W, C_{QW}), (V, Z, C_{VZ})\} \]

The goal of the proposed ATPG for XPDF faults is to find a maximal test pattern for path \( p \) with a conventional delay test pattern. To find a worst-case test for a given victim path \( p \), the don’t-care values can be used to apply some extra constraint transitions, which maximizes crosstalk-induced noise. In [15], a genetic algorithm is used to find a worst-case test for the victim path after a conventional path-delay-fault ATPG process without a justification process. However, since it does not care whether the target fault is activated, the generated test pattern in [15] is actually not a real test for the target fault. This fact leads to difficulties in diagnosis. In addition, the ATPG based on the genetic algorithm tends to be very time consuming. Therefore, unlike previous works, the proposed ATPG requires a process to find possible worst-case test patterns for XPDF faults using don’t-care values in delay test patterns generated by a conventional path-delay-fault ATPG.

The overall algorithm of the proposed test generation for XPDF faults is shown in Fig. 7. The proposed test-generation algorithm consists of five steps: 1) logic simulation with the generated delay test pattern of a victim path; 2) sorting the aggressor candidates by decreasing order of coupling capacitance; 3) applying a constraint for each aggressor candidate; 4) backward implication from the aggressor’s constraints; and 5) forward implication to verify the generated test pattern. In order to test a XPDF fault, two parts of the circuit should be sensitized: 1) the critical victim path \( p \) and 2) a subpath from a primary or pseudopriority input to an aggressor line coupled to a victim line on the victim path \( p \). In these two conditions, since an original delay test pattern is generated for the target victim path and the proposed ATPG for XPDF faults is used for the original delay test pattern, the first condition can be easily satisfied. Therefore, the steps of the proposed ATPG focus on sensitizing the subpath in condition 2) and maximizing the noise effects of potential aggressor lines.

The first step to generate test patterns for XPDF faults is to perform a logic simulation with a delay test pattern of a victim path \( p \) (XFL: the fault list of the XPDFs XPDFs, list of aggressors AGG\(_k\): the total number of target aggressors AGG\(_k\)\_i: \( i \)-th aggressor line in the XPDFs VIC\(_h\): \( h \)-th victim line coupled to AGG\(_h\)\_i C\(_{C_{k\_i}}\): the coupling capacitance value of \( k \)-th candidates in XPDFs, C\(_{C_{r\_i}}\): the total coupling capacitance which affects to the victim path \( p \) Tx\(_{k\_i}\): the generated test pattern for the XPDF\(_i\)).

\[
\text{Test-generation for XPDF}() \{ \\
\text{while (XFL is not empty) {}} \\
\text{XPDF\(_i\) = select_XPDF(XFL);} \\
\text{sort_aggressor_candidates_in_XPDF(XPDF\(_i\));} \\
\text{AGG\(_T\) = 0;} \\
\text{for (k=0; k<the number of candidates; k++)} \\
\text{flag = apply_constraints(AGG\(_k\));} \\
\text{if (flag == SUCCESS) {}} \\
\text{flag = backward_implication(AGG\(_k\));} \\
\text{Tx\(_k\) = forward_implication(XPDF\(_i\));} \\
\text{if (flag == SUCCESS) {}} \\
\text{C\(_{C_{r\_i}}\) = C\(_{C_{k\_i}}\) + C\(_{C_{r\_i}}\);} \\
\text{AGG\(_T\) = AGG\(_T\) + 1;} \\
\} \\
\text{else {}} \\
\text{XPDF\(_i\) = eliminate_candidate(AGG\(_k\), VIC\(_h\));} \\
\text{continue;}} \\
\text{if (AGG\(_T\) > 0)} \\
\text{Tx\(_k\) = forward_implication(XPDF\(_i\));} \\
\}
\]

Fig. 7. Test-generation algorithm of the proposed ATPG.
model, larger coupling-capacitance values will generate larger crosstalk-induced noise. In addition, because the number of possible aggressor combinations is enormously large in the selected XPDF fault, searching all combinations is not practical. Therefore, as a heuristic method, we use the sorted aggressor list in the selected XPDF fault to maximize the crosstalk-induced delay caused by potential aggressor lines. For example, consider the XPDF fault in Fig. 6. The aggressor lines in the XPDF fault can be sorted to the following:

\[
\{(Q, W, C_{QW}), (D, G, C_{DG}), (K, N, C_{KN}), (V, Z, C_{VZ}), (D, F, C_{DF}), (D, I, C_{DI}), (D, J, C_{DJI}), (K, O, C_{KO})\},
\]

where \(C_{QW} > C_{DG} > C_{KN} > C_{VZ} > (C_{DF} = C_{DI} = C_{DJI} = C_{KO})\).

In the third step, an aggressor candidate line is selected from the sorted aggressor candidate list of the XPDF fault, and then, the constraint transition signal on the selected aggressor line is applied. Assuming that the W line is selected as a first aggressor candidate, Fig. 8 shows how to apply the constraint signal on the selected aggressor line. As shown in Fig. 8, since the victim line coupled to the W line is the Q line and the transition of the W line is falling, the constraint signal applied on the Q line must be a rising transition.

After applying the constraint on the selected aggressor line, backward implication from the selected aggressor line is performed to sensitize subpaths from the aggressor to primary or pseudoprimary inputs. This step is similar to the backtracing step in a conventional PODEM-based delay ATPG. In this step, nonrobust path-delay constraints are encouraged to be used in the ATPG of crosstalk-induced faults because it leads to increase the possibility to generate affecting transitions. In addition, since the proposed ATPG uses logic-simulation results with the original delay test patterns, the conflicts caused by the constraints on the aggressor line can be identified quickly. Fig. 9 shows the backward implication for the pair of coupling candidates (Q, W) shown in Fig. 8.

During the backward-implication step, the lines passed backward are marked, such as lines I, J, and O in Fig. 9. Then, the proposed ATPG checks if these marked lines have any compatibility with other aggressor candidates in the selected
XPDF fault. If the marked line is compatible with other aggressor candidates, the compatible line can be checked with the aggressor candidate list, and the backward implication from it may not be required. Otherwise, the marked line can be eliminated from the aggressor candidate list. This step is a part of the functional-pruning procedure in the proposed ATPG. For example, during the backward implication shown in Fig. 9, line O assigns the rising transition, and this transition cannot satisfy the condition for crosstalk slowdown delay due to the aggressor candidate (K, O, C_{DKO}) in the selected XPDF fault. Therefore, this aggressor candidate (K, O, C_{DKO}) can be eliminated. In addition, as shown in Fig. 9, since lines I and J assign the falling transitions, and these lines are compatible with the aggressor candidates (D, I, C_{CDI}) and (D, J, C_{CDJ}), respectively, these aggressor candidates can be marked in the aggressor candidate list.

After executing the backward implication and checking the compatibility of other aggressor candidates, the final step, i.e., forward implication, is performed with the updated test pattern. In the forward implication of the proposed ATPG, since the activated transition on the target aggressor line is not required to propagate to primary or pseudoprimary outputs, this step only determines if there are any conflicts on other lines and if there are any compatible aggressor lines. Fig. 10 shows the result of the forward implication with the updated test pattern shown in Fig. 9.

As shown in Fig. 10, there is no conflict between line Z and other lines. In addition, there is no compatible aggressor line during the forward implication in Fig. 10. Thus, we can identify a set of effective aggressor lines \(\{Q, W, C_{CW}, (D, I, C_{DI}), (D, J, C_{DJI})\}\) and update the test pattern from \(\{A, B, C, D, E, F, G, H, I, J\} = \{X1, X1, 11, 01, 11, XX, XX, 00, XX, XX\}\) to \(\{X1, X1, 11, 01, 11, XX, XX, 00, 10, 10\}\). In addition, the remaining aggressor candidates are \(D, G, C_{CDG}, (K, N, C_{CKN}), (V, Z, C_{CVZ}),\) and \(D, F, C_{CDF}\) in this example.

For the remaining aggressor candidates, by repeating steps 3)–5), the final test pattern can be generated, which maximizes the effects of the crosstalk-induced noises between possible aggressor lines and the victim path. In the example shown in Fig. 10, after repeating steps 3)–5) for the remaining aggressor candidates, the final test pattern is generated, i.e., \(\{X1, X1, 11, 01, 11, 10, 00, 10, 10\}\), as shown in Fig. 11. In addition, the final aggressor lines coupled to the victim path \(p\) are \(\{Q, W, C_{CW}, (D, G, C_{CDG}), (K, N, C_{CKN}), (D, F, C_{CDF}), (D, I, C_{CDI}),\) and \(D, J, C_{DJI}\).\)

As shown in the test-generation algorithm of Fig. 7, after generating the final test pattern for the selected XPDF fault, the forward implication with the final test pattern should be performed to check whether there are any conflicts between the sensitized victim path and the final aggressor lines.

By repeating the aforementioned steps for all XPDF faults, the proposed ATPG can generate test patterns to activate possible crosstalk effects on the victim paths. Since the generated test patterns for XPDF faults can guarantee to sensitize both critical victim paths and subpaths from primary or pseudoprimary inputs to aggressor lines coupled with the victim paths, we can claim that the proposed ATPG is more effective than previous methods for crosstalk-induced slowdown faults.

VI. EXPERIMENTAL RESULTS

The proposed ATPG was implemented on C and was run on ISCAS 85, ISCAS 89, and ITC 99 benchmark circuits. All results were obtained with 0.13-\(\mu\)m technology. All experiments were performed on a Sun Blade 2000 system with Solaris.

In the first experiment, to compare the efficiency of the proposed ATPG with previous works, the fan-out weighted delay model is used to obtain timing information for experimental evaluation. In addition, due to lack of layout information, coupling capacitances are not considered in the fault selection. In this experiment, assuming that a single-aggressor line is coupled to the target victim path, a single-aggressor line is randomly selected for the proposed ATPG and the previous works in [12] and [13]. The maximum number of target paths for each benchmark circuit is 1000, while some small benchmark circuits have less paths. Table I shows a comparison of the proposed ATPG and previous works for single-aggressor lines.
In Table I, columns 1 and 2 show the names of the circuits and the number of target faults that consist of the critical victim path and the randomly selected aggressor lines. Columns 3, 5, and 7 show the number of detected faults for [12], [13], and the proposed ATPG-XP, respectively. In addition, the fault coverage of each method is illustrated in columns 4, 6, and 8.

As shown in Table I, the fault coverage of the proposed ATPG-XP for crosstalk-induced delay faults is the highest among the three techniques. Because the test-generation method in [12] is based on a single precise crosstalk-induced path-delay fault and the path sensitization criterion of robust test generation, it is obvious that the fault coverage decreases, while it can lead to high diagnosis resolution for crosstalk-induced delay faults. However, since the proposed ATPG uses delay test patterns for target critical paths, it guarantees that the effects of crosstalk-induced delay faults are propagated through the target critical paths. Therefore, test patterns generated by ATPG-XP can be useful for the diagnosis for delay faults. For crosstalk faults coupled to a single-aggressor line, we can claim that ATPG-XP is one of the most attractive ATPG methods for crosstalk-induced delay faults.

Table II shows the CPU time for test generation and the number of aborted faults for each technique. In Table II, columns 1 and 2 show the names of the circuits and the number of target faults that consist of the critical victim path and the randomly selected aggressor lines. Columns 3, 5, and 7 show the number of aborted faults for [12], [13], and ATPG-XP, respectively. In addition, the CPU time of each method is illustrated in columns 4, 6, and 8. The CPU time in the eighth column includes both the time required for test-pattern generation of path-delay faults and the extra time for the proposed method. These results show the efficiency and ability of test generation for the given faults.

As shown in Table II, the proposed ATPG-XP can generate test patterns for the same crosstalk faults faster than previous works. In addition, the number of aborted faults for the proposed method is less than those of previous works. In particular, the proposed method significantly reduces the test-generation time. Hence, ATPG-XP is more practical for large industrial circuits than previous works. In addition, since previous works, i.e., [12] and [13], originally target a pair of a victim node and an aggressor line, the original test-generation methods are more time consuming than those used in this experiment.

In [12] and [13], a single crosstalk-induced path-delay-fault model, namely, the single precise crosstalk-induced path delay fault (S-PCPDF) model, is used. A target S-PCPDF fault consists of a victim path, an aggressor line, and a subpath that is to be sensitized to generate the necessary aggressor transition...
TABLE II
COMPARISONS OF THE EFFICIENCY OF TEST GENERATION FOR A SINGLE AGGRESSOR


However, since the proposed method considers constraints for sensitizing multiple-aggressor lines with the generated delay test patterns, the computational complexity of the proposed ATPG is much lower than those of [12] and [13]. Therefore, the number of aborted faults of the proposed method is much smaller than those of [12] and [13]. Moreover, since [12] and [13] consider the constraints for both path-delay faults and the aggressor line coupled with the victim line simultaneously, some paths are actually testable can be determined to untestable crosstalk-induced faults by using the methods of [12] and [13]. However, in the proposed method, since we can identify the testable victim paths and generate test patterns for delay faults of victim paths through the conventional delay ATPG, if an aggressor of a target XPDF fault cannot be sensitized during the proposed ATPG, this XPDF fault can be determined to the redundant fault, not the untestable fault. For these reasons, the proposed ATPG can achieve higher fault coverage than [12] and [13].

In the second experiment, only the delay-testable longest paths were randomly selected from Table I, and we randomly selected 100 aggressor lines for each victim path of large benchmark circuits. Note that the ASTRO tool [27] was used to place and route the bench circuits, and the STAR-RCXT tool [28] was used to extract the parasitic information with 0.13-μm library. While exact parasitic information for benchmark circuits is extracted in this experiment, we cannot use the parasitic information extracted from STAR-RCXT [28] because the size of the pad used for the routing process is too large for the benchmark circuits. Therefore, in this experiment, the threshold capacitance is defined arbitrarily, and 100 aggressor candidates for each victim path are randomly selected. Table III shows the experimental results of the proposed ATPG on large circuits.

In Table III, columns 1 and 2 show the names of the circuits and the number of delay-testable victim paths. Column 3 shows the number of initial aggressor lines selected randomly for each victim path. The number of finally detected faults and the test efficiency of the proposed ATPG are presented in the last two columns. The results show that the proposed method can achieve higher fault coverage than [12] and [13].
ATPG are shown in columns 5 and 6, respectively. In addition, the CPU time required for each benchmark circuit is illustrated in column 7. The CPU time in the seventh column does not include the time required for the test-pattern generation of path-delay faults. It only reflects the time required for pruning false-aggressor candidates and generating the test patterns for the target XPDF faults.

The average reduction ratio of false-aggressor candidates in this experiment is 91.15%. In addition, the average test efficiency of this experiment is 99.32%. As shown in Table III, the average CPU time for pruning false-aggressor candidates and generating the test patterns for XPDF faults is usually a few seconds, which is acceptable for generating crosstalk-induced fault tests for large industrial circuits. Therefore, since test patterns generated by the proposed ATPG can activate many potential aggressor lines and propagate the effects of crosstalk-induced slowdown delay through the target victim path, the proposed ATPG is more efficient for test generation of crosstalk-induced faults than other ATPG methodologies. In addition, the proposed ATPG can generate test patterns much faster than previous works.

VII. CONCLUSION

In this paper, we propose a new test-generation method for delay faults considering crosstalk-induced delay effects, which is based on a conventional delay ATPG technique in order to reduce the complexity of previous ATPG algorithms and to consider multiple-aggressor crosstalk faults to maximize the noise of the victim line.

For a sensitizable victim path, a new algorithm is proposed which appropriately activates its associated aggressor-victim pairs for maximization of crosstalk. To reduce the number of crosstalk-induced delay faults, the proposed algorithm uses parasitic information such as the coupling capacitance between a node of the victim and an aggressor. In addition, we propose a new timing refinement method to more accurately identify timing false crosstalk faults.

Unlike previous works, ATPG-XP uses a process to find the worst-possible-case test patterns for the generated crosstalk faults using don’t-care values in delay test patterns generated by a conventional path-delay-fault ATPG. As shown in experimental results, since test patterns generated by the proposed method can activate many potential aggressor lines and propagate the effects of crosstalk-induced slowdown delay through the target victim path, the proposed ATPG-XP is more efficient for test generation of crosstalk-induced faults than other ATPG methodologies.

REFERENCES


[27] ASTRO, Synopsys Corp. v2006.06.

[28] STAR-RCXT, Synopsys Corp. v2006.06.

Sunghoon Chun received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2002, 2005, and 2008, respectively. He was a Research Engineer with the ASIC Research Center, Yonsei University. He researched for test methodologies for systems-on-a-chip (SoCs). He is currently a Researcher with the Department of Computer Science and Engineering, University of California, La Jolla. His areas of interest include SoC testing, delay testing, fault diagnosis, functional testing for processor-based systems, and test methodologies for signal integrity faults. Dr. Chun was a recipient of the Samsung Golden Prize in the 2007 International SoC Design Conference.

Taejin Kim (S’07–M’09) received the B.S. and M.S. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea. He is currently with the Avionics R&D Laboratory, LIG Nex1 Company, Ltd., Daejeon, Korea. His interests include digital testing, and he is currently working on improving the effectiveness of system Joint Test Action Group and board testing.

Sungho Kang (S’87–M’93) received the B.S. degree from Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas (UT), Austin. He was a Postdoctoral Fellow with UT, a Research Scientist with the Schlumberger Laboratory for Computer Science, Schlumberger, Inc., and a Senior Staff Engineer with Semiconductor Systems Design Technology, Motorola, Inc. Since 1994, he has been a Professor with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul. His current research interests include system-on-a-chip (SoC)/VLSI design, VLSI computer-aided design, and SoC testing and design for testability.