A Novel Pattern Generation Framework for Inducing Maximum Crosstalk Effects on Delay-Sensitive Paths *

Jeremy Lee and Mohammad Tehranipoor
ECE Department
University of Connecticut
{jslee,tehrani}@engr.uconn.edu

Abstract

The limitations of pattern generation tools are beginning to surface as parasitic coupling capacitance in high speed interconnects only worsens as the industry approaches sub-50nm technologies. This can create a gap between the delay experienced on critical and long paths during test and the delay of the same paths in the field. In this paper, we propose a novel structural test pattern generation procedure that magnifies parasitic crosstalk effects on delay-sensitive paths by inducing switching on nearby nets which have been identified using the parasitic information of the layout. This will ensure timing closure on the targeted path is still met while also minimizing escape ratio and improving in the field reliability. Results of the proposed layout-aware approach demonstrate the ability of the proposed framework to significantly increase crosstalk around the targeted delay-sensitive paths.

1 Introduction

As process technology continues to approach the very deep sub-micron regime, many second order effects that were negligible in the past now pose new obstacles during design and test [1]. Signal integrity, in particular, is playing a significant role when testing modern designs. The number of silicon failures and escapes caused by signal integrity problems is on the rise because existing design tools and test methodologies cannot fully address these new issues effectively. In nanoscale designs, delay has become dominated by interconnect-dependent RC delay, cross coupling, and via resistance. As a result, signal integrity issues must be resolved during design and considered during manufacturing test to ensure successful tape-out and reliability in the field.

*This work was supported in part by Semiconductor Research Corporation under contracts No. 1455 and 1587.

Structural patterns are often used to test DSPs, microprocessors and complex ASICs and SoCs. However, functional patterns may also be used to test and debug hard-to-detect, timing-related defects. The need to magnify the impact of these defects becomes increasingly important to increase the probability of detection. For instance, crosstalk effects can cause timing problems on targeted critical or long (*delay-sensitive*) paths either by slowing it down or speeding it up. If crosstalk effects are not carefully considered during manufacturing test, a chip may pass structural or functional tests, but a field failure may still occur.

Normal functional patterns may not be able to effectively maximize the crosstalk effects on a chip when targeting delay-sensitive paths. In addition to testing these paths, they must also model other functional use conditions in the rest of the design to effectively detect such hard-to-detect defects. However, when considering all sources of delay variation (crosstalk, process variation, temperature, etc.), generating such efficient functional patterns will be a challenging task and can be prohibitively expensive. As a result, new automatic test pattern generation (ATPG) techniques are required to maximize such effects while still ensuring high fault coverage and low pattern count. Such structural test patterns must also be used for validating the design before sign off. In addition to design validation and defect screening, these new patterns must also assist in diagnosis and failure analysis to identify the root cause of the failure.

Path delay fault tests are generally used to test delay-sensitive paths but are currently generated without taking any signal integrity issues into consideration. When the test pattern is generated for a path, the unspecified bits are filled in different ways, e.g. 0-fill, 1-fill, random-fill or adjacent-fill. These methods may not cause significant coupling effects on paths under test, which would allow the chip to pass during test but fail in the field since current ATPG methods are not aware of the potential switching that may occur around the path under test. New pattern generation

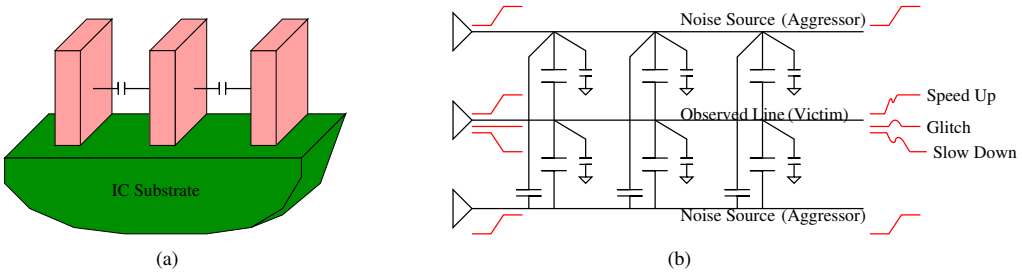

Figure 1. (a) Sidewall capacitance effects increase with shrinking feature sizes and (b) Interconnect C only model (for simplicity R is not shown).

procedures are required to consider transitions on all neighboring nets without extensive simulation. Also, new pattern generation methods are needed to maximize the coupling effects on critical paths to reduce escape and ensure reliability in the field.

1.1 Related Prior Work

Several techniques have been proposed to deal with crosstalk and signal integrity issues during verification and test. Crosstalk verification with interconnect process variation is discussed in [10]. The authors in [7] present fault modeling (called maximum aggressor (MA)) and simulation for crosstalk on SOC interconnects. The investigation has shown that the MA model may not always ensure highest crosstalk effects [14] [9] especially when mutual inductance effects are considered in addition to timing of transitions. Some researchers have proposed using on-chip sensors or glitch/delay detectors [17] [16] to detect noise and delay violations. The drawback of such techniques is that the sensors must be tuned and very accurate and adding one sensor per interconnect is prohibitively expensive. Due to their high sensitivity to voltage and timing, process variation can also negatively impact their operation. The authors in [14] [15] utilize the boundary scan cells to generate test patterns to detect noise and delay violations on a system chip and observe the responses which are then scanned out using boundary scan shift procedure. Most of the proposed techniques, mentioned above, target only buses or interconnects between cores in a SOC rather than internal paths.

Authors in [13] propose validation and test generation for crosstalk-induced delay and noise for SOC interconnects. An analytical model for crosstalk was developed in [4] and used as a basis for pattern generation to induce delay due to crosstalk in [5]. However, this approach only generates patterns for a single aggressor affecting a target path. The procedure proposed in [8] considers a genetic algorithm based approach when inducing crosstalk into delay test patterns. There has also been a proposed academic ATPG that considers crosstalk and transition arrival times during pattern generation [6] [12] but it lacks the immediate use in practice since they are computationally intensive.

1.2 Contribution and Paper Organization

Crosstalk has become a significant factor during on-chip signal integrity analysis in modern designs. This can be attributed to the fact that as technology scales, interconnect spacing and width are also being reduced, while, in an effort to reduce wire resistance, the thickness is not scaling at the same rate. This produces tall sidewalls between long parallel interconnects separated by very little space, which is ideal for creating a capacitor. Figure 1 shows the sidewall capacitance between nearby wires and the interconnect C model for the same wires. The cross-coupling capacitance is only expected to increase as technology continues to scale, magnifying potential crosstalk effects.

The coupling capacitance is proportional to both the rise and fall time of transitions on aggressor nets and the spacing of interconnects around the victim net. However, the amount of coupling experienced when crosstalk occurs is dependent on two additional factors: aggressor transition direction with respect to the victim and the arrival time of aggressor transition with respect to victim transition arrival time. While both components have a direct effect on coupling and propagation delay, in this work we focus on the former, i.e. transition direction, due to the difficulty in correctly predicting actual circuit timing without extensive simulation and potentially prohibitive process variation and statistical analysis. Although previous work has addressed transition timing [6] [12], we will intend to add this feature in the future.

In this paper, we propose a novel and effective structural delay test pattern generation framework to magnify crosstalk effects across delay-sensitive paths. By considering parasitic information, obtained using actual layout information, and transition behavior, a novel practice-oriented pattern generation procedure

is developed to accurately target such timing-related defects. The physical layout information will be transported into extraction and timing analysis tools and then neighboring nets affecting delay-sensitive paths (i.e. hindering performance) will be identified and used in the pattern generation process. Our proposed pattern generation procedure, called *Weighted-Xtalk-TDF ATPG*, combines two well established path delay fault (PDF) and transition delay fault (TDF) pattern generation and utilizes test points on neighboring nets. We will also develop a compaction scheme to generate a final pattern from the set of generated vectors for each delay sensitive path that will increase the switching activity on the nearby nets.

The paper is organized as follows. In Section 2, we introduce Weighted-Xtalk-TDF ATPG to induce maximum crosstalk effects on delay-sensitive paths. Section 3 discusses our implementation of the pattern generation procedure. We present experimental results and analysis in Section 4. Finally, we discuss our conclusions in Section 5.

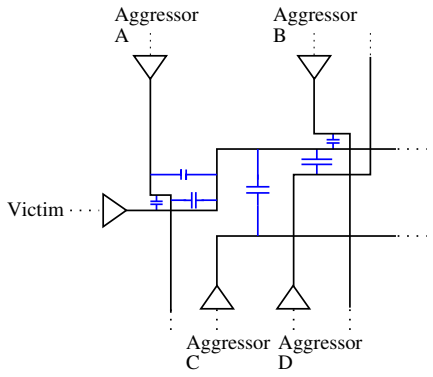

Figure 2. The coupling capacitance effects on a victim line from four aggressors. The amount of coupling is represented in the size of the capacitor, which are larger for longer distances of parallel interconnects. For simplicity, coupling between aggressors are not shown.

2 Inducing Maximum Coupling Effects on Delay-Sensitive Paths

Accurate characterization of coupling effects have previously required SPICE simulations or static crosstalk analysis. Although these methods may be relatively effective for design, SPICE simulations are extremely time consuming while static analysis does not always provide accurate worst-case scenario for path timing. However, we propose an efficient approach that takes layout information into account when generating a pattern that will maximize crosstalk effect on targeted paths. Our objective is to remove the need for aggressive guardbanding that is often used to deal with signal integrity issues in today's integrated circuits.

2.1 Identifying Nearby Nets

Identifying those nets that have a significant effect on the delay-sensitive paths require knowledge of the physical design. However, simply looking at all nets within a particular area of the targeted path will not necessarily identify nearby nets that will have a significant crosstalk effect. For example, two nets that are routed closely together but for a very short distance will not create enough crosstalk to impact the timing of the delay-sensitive path. But if we use the extracted coupling capacitance of each of the nets, we can identify those nets that will have a significant effect and also reduce the number of nearby nets that will be targeted when inducing noise on the delay-sensitive path.

As shown in Figure 2, after routing, the victim path may have many aggressor paths that are minimally spaced but only parallel for short distances while having a few aggressors that are parallel to the net for a significant distance but several pitches away. The relative coupling of the aggressors with the victim is shown by the size of the capacitor. The coupling value between the Victim and Aggressor C will reflect the pitch and parallel distance. This will be larger than the coupling capacitance between the Victim and Aggressor A, but less than the coupling between the Victim and Aggressor D. So, even though A is closer to the Victim than C, it is parallel for a very short distance, but since the pitch between D and the Victim is smaller, the coupling will be larger even though the parallel distance is shorter. Even though Aggressor A is physically closer than either Aggressors C or D, it may not be considered a neighboring net if the coupling value does not reach a *minimum threshold*.

A minimum coupling threshold is used to reduce the complexity during analysis by filtering some of the neighboring nets that have almost no effect on the victim path. This will eliminate nets that may be near each other but are routed perpendicularly.

2.2 Xtalk-TDF ATPG

In the remainder of this paper, we will focus on inducing crosstalk effects to increase propagation delay (slow down) by maximizing opposite transitions on neighboring nets, but the proposed approach can also be utilized to *i*) reduce propagation delay (speed up) by forcing transitions in the same direction on neighboring nets and *ii*) increase IR-drop and ground bounce.

There are a variety of options available to drive an

opposite direction transition on the neighboring path. The first option, which is computationally expensive, is to deterministically generate a pattern that tests the delay-sensitive path while concurrently justifying the pattern for transitions on neighboring nets [5]. A second option would be to approach this problem with a genetic algorithm, but is not deterministic and requires iterative simulations [8]. However, our approach intelligently combines two established delay testing techniques, path delay fault (PDF) and transition delay fault (TDF) pattern generation, to maximize transitions on the neighboring aggressors.

In the proposed method, PDF patterns are used to ensure the targeted delay-sensitive paths are being tested while TDF patterns are used to generate switching on those neighboring nets that have already been identified as effective aggressors. To ensure that these TDF patterns are still compatible with the PDF patterns that test the delay-sensitive paths, the PDF patterns are used as ATPG constraints. The direction of the transition on a net of the delay-sensitive path determined by the PDF pattern will determine whether a slow-to-rise (*str*) or a slow-to-fall (*stf*) fault on the neighboring net is targeted during TDF ATPG. Hereafter, we will call the TDF-based method that targets the neighboring aggressors all at once *Xtalk-TDF ATPG*.

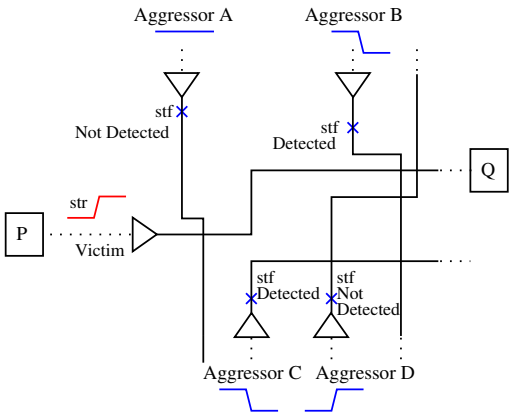

Figure 3. A path delay fault pattern that successfully detects a slow-to-rise fault on one of the nets on path PQ (victim path) is used as constraints when generating a single pattern that will also detect slow-to-fall transition faults on neighboring aggressors.

Using the same circuit as before, Figure 3 shows a rising transition created by a PDF pattern on the Victim path. For this example, *stf* faults on aggressors are added to the fault list and used during TDF pattern generation. The generated pattern causes Aggressor A to remain quiescent for the duration of the test, but causes transitions on Aggressors B and C. Since a *stf* on Aggressors B and C was in the fault list, an opposite transition from the victim net was successfully generated. However, for Aggressor D, due to a combination of constraints from the PDF and inducing transitions at other fault sites, a rising transition is generated leaving the *stf* fault undetectable. So, the total number of detectable transition faults relies on the constraints imposed by using the PDF pattern and prior faults that have already been detected.

While generating a single TDF pattern using Xtalk-TDF ATPG with constraints from the PDF pattern is effective at ensuring that some of the switching around the targeted path is being stimulated, there are still several limitations when trying to maximize crosstalk. First, during Xtalk-TDF ATPG, we are only aware of switching at those sites that are detected. At the same time, unaccounted transitions may occur at fault sites that have been classified as ATPG untestable since the transition could not be propagated to an observation point. Also, the ATPG algorithm does not prioritize those nets that have greater coupling, so it could potentially generate a pattern that creates switching on the least effective neighboring nets while ignoring the more effective neighbors. In other words, generating a large number of opposite transitions on aggressors will not ensure maximum crosstalk effect on the target path.

2.3 Virtual Test Point Insertion

During Xtalk-TDF ATPG, a transition fault will be considered detected when a transition can be activated at a fault site and also propagated to an observation point. However, the propagation phase of the ATPG algorithm can be a drawback when attempting to induce switching around delay-sensitive paths. Since only the actual switching is necessary, propagating the transition to an observation point is unnecessary and may be creating additional care-bits that could be better used to activate a transition at another fault site. Also, the transition may only be activated but cannot be propagated, so while the effect would be desired, the pattern is not kept by the ATPG.

To avoid both of these issues, we alter the netlist to include additional test points at the neighboring nets. The effect of the alteration is "only" for TDF pattern generation and *has no impact* on the final design of the chip and actual location of neighboring nets. These virtual test points provide new observation points to reduce the amount of effort the ATPG needs to propagate the transition. This reduces the number of care-bits necessary in the pattern since fewer gates need to be controlled by the ATPG and allows us to conclude that any ATPG untestable faults are due to uncontrol-

lable transitions at the fault site since a transition at the fault site should be immediately observable.

2.4 Weighted-Xtalk-TDF ATPG

As mentioned above, the main disadvantage of generating a single TDF pattern is the lack of control over which faults are detected by the ATPG engine first. Neighboring nets with higher coupling capacitance should be given priority over nets that will have little effect. Otherwise, if the lower effect neighbors (neighbors with smaller coupling effects) are detected by the ATPG first, bits in the pattern used for detection of this neighbor may conflict with those neighbors with higher coupling effects, preventing the ATPG from detecting those fault sites in a single pattern.

Weighted-Xtalk-TDF ATPG is our proposed method for circumventing the ATPG limitation and generates a single vector per neighboring net/segment. It will create a vector set that can then be compacted using an algorithm that is aware of the coupling effects and prioritize those vectors that will create activity on the highly coupled neighbors.

Since there are potentially many neighboring nets to a single delay-sensitive path, the granularity of the Weighted-Xtalk-TDF ATPG can be changed to meet performance requirements. Here, we consider three cases for pattern generation. In all three cases, Xtalk-TDF ATPG is used *with* virtual test points inserted into the netlist (called the virtual netlist).

1. *Full*: In this case, one pattern is generated using Xtalk-TDF ATPG considering the full neighboring fault list. A single pattern is generated. Note that no compaction is required in this case since it is very similar to Xtalk-TDF ATPG described in Section 2.2 except that it uses virtual test points to provide higher observability. Using these virtual test points will yield a better result.

2. *Segmented*: Generates one vector using Xtalk-TDF ATPG for each segment of the targeted path in the temporary netlist. A segment is defined as all nets near any net on the target path. In this case, n vectors will be generated, where n is the number of segments which is equal to the number of nets on the target path. These vectors are then compacted to generate a single pattern considering the size of coupling capacitances.

3. *Per-neighbor*: Generate one vector per neighbor net using Xtalk-TDF ATPG. In this case, there could be as many vectors as there are neighbors, m. Similar to *Segmented* case, the compaction procedure must be used for these vectors to generate the final pattern.

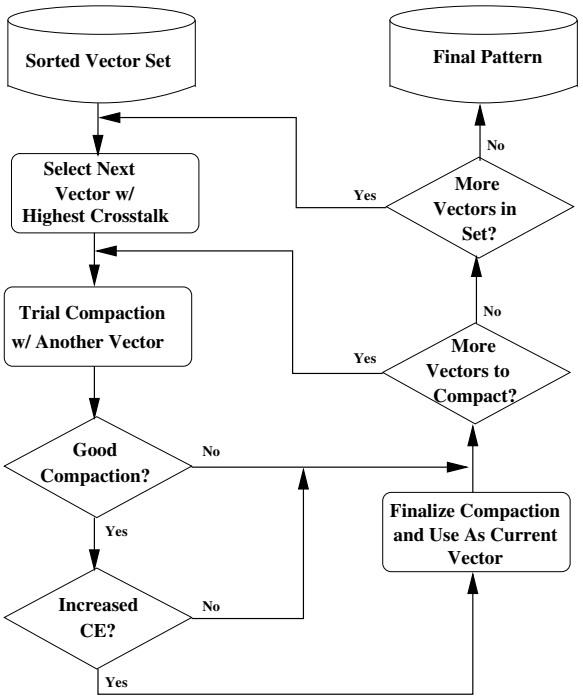

Figure 4. Flow diagram for crosstalk-aware compaction algorithm.

Compacting either the segmented or per-neighbor TDF vector sets can be approached in a similar manner. A greedy algorithm approach to compaction is used after sorting the vector set from greatest crosstalk effects to least. Figure 4 shows a flow diagram of the compaction algorithm. First the algorithm will check if two vectors can be compacted (compatible vectors). If possible, a second check is performed to ensure that compacting the two vectors will result in a greater number of crosstalk effects than the original vector. If so, the compaction is retained and the new vector is used until it can be compacted with another vector or the end of the vector set is reached. This process will be performed starting with each vectors in the set once. Although this may impact run-time since the algorithm has a complexity of $O(n^2)$, it will create a larger number of vector combinations to select the best pattern from.

Determining whether compaction would result in increased crosstalk is based on the crosstalk metric in Equation 1.

$$CE = \sum_{i=0}^{m} dC_i, \text{ where}$$
$$d = \begin{cases} 1, & \text{Opposite transition} \\ 0, & \text{No transition} \\ -1, & \text{Same transition} \end{cases} \quad (1)$$

The equation considers for all m neighboring nets, the direction of the transition with respect to associated net of the targeted delay-sensitive path, d, and the amount of coupling between the two nets, C_i. A positive value of CE indicates the targeted path will experience slow-down due to induced crosstalk effects from the switching neighbors. Similarly, a negative CE indicates the targeted path will experience speed-up. The proposed compaction procedure is simple and quite fast. However, it can be improved by taking the timing of transitions in to account which is part of our future work.

3 Weighted-Xtalk-TDF ATPG Framework

Our pattern generation framework involves three major steps as shown in the flow diagram in Figure 5. The three steps consist of parasitic extraction of the physical design; identification of the delay-sensitive path and neighboring nets; and Weighted-Xtalk-TDF ATPG. Each of these steps are briefly described in the following.

• **Step 1: Parasitic Extraction:** For our implementation, we extracted the physical parasitic effects of the layout using Synopsys Star-RCXT [3]. This provides a 3-D extraction of the layout and will acquire wire resistances, capacitance to ground, and coupled capacitance. To maintain reasonable run-times, only parasitic effects above a user-defined minimum coupling threshold are extracted, which can be modified to include greater or fewer capacitances. This initial filtering criteria limits the number of nets (m) that will be considered as neighboring nets to further reduce complexity.

• **Step 2: Delay-Sensitive Path and Neighboring Net Identification:** We identify the delay-sensitive path using PrimeTime SI [3]. PrimeTime SI uses the parasitic information from the extraction tool to determine the critical path and other long paths that are sensitive to delay variation. For this work, we used the critical path identified by the tool.

Since the tool stores the parasitic coupling information, custom scripts have been developed to report this for each net of the critical path to assist in identifying the effective neighboring nets. If a net is coupled to a neighboring net in several locations, PrimeTime SI will sum the coupling capacitances into a single value. Only those segments with a coupling capacitance above a user-defined threshold will be identified as a neighboring net.

The identified neighboring nets are then used as TDF fault sites as well as test points inserted into a temporary netlist used only for pattern generation. This will give the ATPG tool more observation points during the pattern generation phase of the framework.

• **Step 3: Weighted-Xtalk-TDF ATPG:** The test pattern generation itself can be divided into three (3) components. First, PDF pattern generation to provide constraints during TDF generation. Second, TDF pattern generation, which can be separated into three levels of granularity. Finally, crosstalk-aware compaction algorithm is used for generating the final pattern.

The PDF pattern generation is rather straightforward with little modification to default settings. We generated a robust PDF pattern while leaving all don't-care values unfilled. To ensure the pattern is compatible with TDF generation, the pattern is applied using a launch-on-capture (LOC) [11] clocking scheme. From the PDF pattern, any states filled with a care-bit are then extracted and utilized as a constraint during TDF pattern generation. Note that since LOC is being used, there is a high probability the pattern being applied is functionally valid since the resulting pattern is a functional response of the original test pattern.

As addressed in Section 2.2, the fault sites for TDF testing are based on the neighboring nets of the critical path. The direction of the transition fault is determined by the desired effect on the critical path (opposite direction for slow-down, same direction for speed-up). The direction of the transition on the critical path segment has already been determined during critical path identification. The final stage then generates TDF patterns for the fault lists. Depending on the level of desired granularity during pattern generation (full, segmented, per-neighbor), the number of vectors will vary.

The compaction tool implements the algorithm specified in Section 2.4. In order to determine the switching activity created by each of the patterns from Segmented and Per-neighbor TDF ATPG, the patterns are simulated with Synopsys VCS and monitored using our well-developed Verilog PLI routines. The final result of the compaction tool is a single pattern that maximizes crosstalk effects across the critical path.

4 Experimental Results and Analysis

We implemented Weighted-Xtalk-TDF ATPG on a Linux x86 architecture with a 3.0GHz processor and 1 GB of RAM. Several ISCAS'89 benchmarks were run through physical synthesis using the Faraday 130nm Standard Cell Library [2]. Astro [3] was used to perform the placement and routing of the standard cells.

One critical path was identified by PrimeTime SI for each benchmark. Also, one PDF pattern was generated for each ceritical path using TetraMax ATPG. A cus-

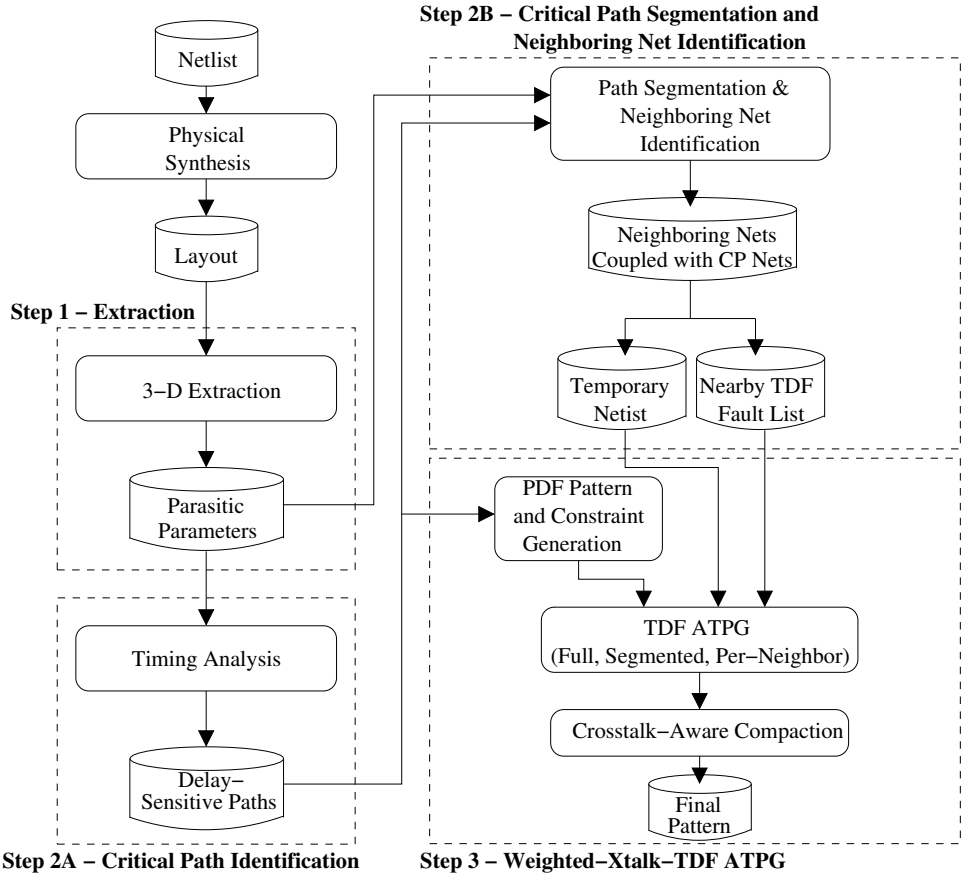

Figure 5. Flow diagram of pattern generation to maximize crosstalk in an identified delay-sensitive paths.

tom Perl script was used to extract the care bits from the PDF patterns to use as constraints during TDF pattern generation. DFT Compiler was used to generate a new temporary netlist with the virtual test points inserted. Table 1 shows the number of flip-flops in each benchmark in Column 2, number of constraints generated by the PDF pattern for the target critical path in Column 3, and number of virtual test points inserted on the aggressors near the critical path iis each benchmark in Column 4. The number of virtual test points will greatly depend on the routing congestion in each circuit and length of the target path. In fact, the number of test points is equal to the number of nearby nets with greater than a coupling threshold. So, the size of circuit does not directly correlate to the number of test points that will be inserted into the temporary netlist. An example of this can be seen when comparing benchmarks s15850 and s38584. Although s38584 contains approximately three times as many gates in the design than s15850, the number of virtual test points inserted is an order of ten fewer.

We generated patterns using Full, Segmented, and Per-neighbor approaches. Our crosstalk-aware compaction tool was used to reduce the TDF vector sets generated using Segmented and Per-neighbor granularity to a single final pattern. To verify the patterns generated by the framework, we have compared the pattern to a the PDF pattern generated by TetraMax with random filling.

Table 1. PDF Constraints and Virtual Test Points.

Benchmark	# of FFs	# of Constraints	# of Virtual Test Points
s9234	211	31	68
s13207	638	24	456
s15850	534	59	622
s35932	1,728	13	151
s38417	1,636	65	270
s38584	1,426	28	69

Table 2 shows the switching activity created on the neighboring nets induced by the final pattern generated by the framework. Columns 2 and 3 show the average number of transitions on neighboring nets in the opposite and same direction for a conventional PDF pat-

Table 2. PDF Pattern vs. Weighted-Xtalk-TDF Framework Switching Comparison.

Benchmark	PDF Switching		Weighted-Xtalk-TDF Switching					
			Full		Segmented		Per-Neighbor	
	Opposite	Same	Opposite	Same	Opposite	Same	Opposite	Same
s9234	22	11	31	10	25	7	21	6
s13207	111	85	128	93	101	70	87	59
s15850	136	106	197	156	102	69	169	109
s35932	31	31	73	16	49	13	65	13
s38417	70	67	76	84	74	73	77	76
s38584	8	10	20	8	21	5	22	5

Table 3. PDF vs. Weighted-Xtalk-TDF Framework Propagation Delay.

Benchmark	PDF Delay	% Incr. Weighted-Xtalk-TDF Delay		
		Full	Segmented	Per-Neighbor
s9234	2.943	4.7	5.6	5.6
s13207	2.264	-2.1	5.5	5.2
s15850	3.530	9.6	12.5	11.6
s35932	1.322	9.3	9.6	9.5
s38417	3.094	4.11	4.12	4.11
s38584	3.234	3.3	5.3	8.4

tern. Columns 4-9 show the number of transitions induced by the Weighted-Xtalk-TDF ATPG in the opposite and same directions for Full, Segemented, and Per-neighbor approaches. In all cases except for s13207, the amount of switching activity in the opposite direction was increased when compared to conventional PDF pattern generation. However, when comparing the amount of switching activity between the three approaches, the Full granularity consistently generated the most switching in the opposite direction. Although this may seem like an advantage in favor of the Full granularity, it also generated the greatest amount of switching in the same direction too. This demonstrates the crosstalk-aware compaction algorithm's ability to balance and prioritize the switching activity based on the magnitude of the coupling effects for both the Segmented and Per-neighbor approaches. Also, since the Per-neighbor granularity has the ability to activate each neighbor individually, it can more effectively control the switching activity along the delay-sensitive path compared to the the Segmented granularity.

Table 3 reports the percentage increase in delay generated by Weighted-Xtalk-TDF for each granularity compared to the conventional PDF pattern. The delay values were acquired by simulating a SPICE-level model of each benchmark using Nanosim/VCS mixed simulation [3]. Column 2 shows the propagation delay of the conventional PDF pattern to create a transition at the input of the flip-flop at the end of the identified path. Columns 3-5 show the percentage increase in delay for Full, Segmented, and Per-neighbor approaches, respectively. The presented framework increases the path delay due to crosstalk in all cases except for the Full granularity of s13207. When combining these results with those in Table 2, the negative change in delay is likely due to the increase of switching activity in the same direction. So, although the Full granularity is able to increase switching in the opposite direction, the amount of coupling with the same direction transitions outweighs any delay that may have been induced by the increased opposite direction transitions. Note that often the percentage increase in delay for each of the Weighted-Xtalk-TDF approach are very close. This is likely due to the simulation resolution used by the SPICE simulator. Since simulating the ISCAS89 benchmarks at the transistor level can be a time-intensive process, the time resolution was increased to reduce simulation time. As a result, some resolution in the delay measurement is also lost.

4.1 Framework Run-Time

Since the presented framework is heavily integrated into existing steps of design verification and DFT flows, there is little run-time overhead. Critical and long path identification is typically done during timing closure and for PDF pattern generation. While reporting these paths, including a report of all the coupled neighboring nets and converting this list into a fault list typically runs in the order of minutes if many paths are being reported. The same will also hold true for pattern generation. Table 4 breaks down the run-time overhead for each component of each framework step when running for a single path. Column 2 lists the granularity of the approach used. Column 3 lists the overhead of each tool used to segment and identify all neighbors that are coupled to the delay-sensitive path and gen-

Table 4. Weighted-Xtalk-TDF ATPG Run-Time Breakdown.

Benchmark	Granularity	Segmentation, Neighbor ID,& Virtual TPI(sec.)	Xtalk-TDF Generation (sec.)	Xtalk-Aware Compaction (sec.)	Total Time (sec.)
s9234	Full	6	1	–	7
	Segmented	6	16	1	23
	Per-Neighbor	6	29	1	36
s13207	Full	9	1	–	10
	Segmented	9	12	3	24
	Per-Neighbor	9	219	83	311
s15850	Full	14	1	–	15
	Segmented	14	21	12	47
	Per-Neighbor	14	315	483	812
s35932	Full	8	2	–	10
	Segmented	8	6	2	16
	Per-Neighbor	8	86	32	126
s38417	Full	18	2	–	20
	Segmented	18	19	5	42
	Per-Neighbor	18	119	8	145
s38584	Full	16	2	–	18
	Segmented	16	17	2	35
	Per-Neighbor	16	43	5	64

erate the temporary netlist with virtual test points. Column 4 shows the time to generate the vector sets for each approach with constraints in place. Column 5 shows the amount of time to compact the vectors to generate the final pattern. Finally, Column 6 sums the time of the previous stages for the total overhead of the framework for each approach.

Since the same neighbors are targeted regardless of the approach and a single temporary netlist is generated, the time of segmentation, neighbor identification, and virtual TPI will be the same for each approach. TDF Pattern generation will vary depending on the number of patterns that will need to be generated. Since the Full approach will only generate a single pattern, pattern generation will be very short. However, for the other two approaches, pattern generation time will depend on either the number of segments in the path for Segmented pattern generation or the number of neighbors when using the Per-neighbor approach. Xtalk-Aware compaction does not create any overhead for Full granularity since a single pattern was generated during the TDF pattern generation stage. The compaction time for the other two approaches will depend on the number of vectors generated due to the iterative nature of the compaction algorithm. As can be seen, the inefficiency of the crosstalk-aware compaction algorithms slightly impacts run-time for s15850 due to the vector set generated by the Per-neighbor approach.

4.2 Targeting Multiple Delay-Sensitive Paths

While the results above have been targeting a single critical path in each of the benchmarks, the framework can target multiple paths just as easily. Rather than generating a separate temporary netlist for each path, a single netlist can be generated that inserts all of the virtual test points at one time. It may even be possible to use the same PDF pattern to test multiple paths, however this will increase the number of constraints during Xtalk-TDF pattern generation.

Table 5 compares the SPICE simulation of the delay using a conventional PDF pattern and the Weighted-Xtalk-TDF ATPG (Per-neighbor) approach for ten testable critical paths in s38584. For the ten identified paths, 231 virtual test points were inserted for all of the neighbors as opposed to the 69 virtual test points inserted for a single path as shown in Table 1. Column 1 lists the path number, Column 2 reports the average path delay when the conventional random-filled PDF pattern is applied, and Column 3 shows the percentage increase of the Per-neighbor approach. For most cases, the path experiences an increase in path delay when the pattern from our Weighted-Xtalk-TDF ATPG are applied. However, there are two instances where the delay is actually decreased. These cases are likely due to the fact that we do not yet consider transition arrival time on the neighbors w.r.t. the transition arrival time on the targeted path. The total run-time of this approach for all ten patterns is 425s, which is dominated mostly by Xtalk-TDF pattern generation for each neighbor.

5 Conclusion

In this paper, we have presented a novel pattern generation framework to maximize crosstalk effects across delay-sensitive paths. The proposed ATPG, called Weighted-Xtalk-TDF, is practice-oriented and

Table 5. s38584 Ten Testable Critical Paths Delay.

Path No.	PDF Delay	Per-Neighbor % Incr.
1	3.327	7.5
2	3.234	8.4
3	3.286	-0.1
4	3.082	8.1
5	3.233	4.0
6	3.283	0.3
7	3.095	12.4
8	3.392	0.5
9	3.215	-4.0
10	3.092	4.0

combines PDF and TDF pattern generation to test a path and stimulate transitions on neighboring nets. The ATPG considers the size of coupling capacitances and direction of transitions in order to determine a pattern that can maximize crosstalk effects on the targeted path. Three levels of granularity have been presented to allow flexibility between control over individual transition direction throughout the circuit and run-time. The experimental results for several benchmarks show that the Weighted-Xtalk-TDF ATPG is able to increase the coupling effects across critical paths. Generating Weighted-Xtalk-TDF ATPG for multiple delay-sensitive paths has also been explored and the framework scales for additional paths.

6 Future Work

Transition arrival time on neighboring paths play a large role in the effectiveness of the coupled noise experienced on the targeted path. Although transitions must be properly oriented with respect to each other, if skewed by as little as 200ps, the neighboring net will appear quiescent as the transition propagates through the targeted path. We intend to expand this work to also include transition arrival time by stimulating transitions on the neighboring nets through multiple paths to increase the likelihood one of these paths will generate a transition that will arrive close to the transition on the targeted path.

References

[1] International technology roadmap for semiconductors (itrs), 2005.

[2] Umc 0.13μm logic sp (fsg) process high density core cell library v2.0. Faraday, Inc., 2006.

[3] User manual for synopsys toolset version 2006.09. Synopsys, Inc., 2006.

[4] W. Chen, S. Gupta, and M. Breuer. Analytic models for crosstalk delay and pulse analysis under non-ideal inputs. In *Proc. of Intl. Test Conf. (ITC'97)*, pages 809–818, 1997.

[5] W. Chen, S. Gupta, and M. Breuer. Test generation for crosstalk-induced delay in integrated circuits. In *Proc. of Intl. Test Conf. (ITC'99)*, pages 191–200, 1999.

[6] W. Chen, S. Gupta, and M. Breuer. Test generation for crosstalk-induced faults: Framework and computational results. In *Proc. of Asian Test Conf. (ATS'00)*, pages 305–310, 2000.

[7] M. Cuviello, S. Dey, X. Bai, and Y. Zhao. Fault modeling and simulation for crosstalk in system-on-chip interconnects. In *Proc. of Intl. Conf. on Computer-Aided Design (ICCD'99)*, pages 297–303, 1999.

[8] A. Krstic, J. Liou, Y. Jiang, and K. T. Cheng. Delay testing considering crosstalk induced effects. In *Proc. of Intl. Test Conf. (ITC'01)*, pages 558–567, 2001.

[9] S. Naffziger. Design methodologies for interconnects in ghz+ ics. In *Proc. of Intl. Solid-State Conf.*, 1999.

[10] N. S. Nagaraj, P. Balsara, and C. Cantrell. Crosstalk noise verification in digital designs with interconnect process variations. In *Proc. of Intl. Conf. on VLSI Design*, pages 365–370, 2001.

[11] J. Savir and S. Patil. On broad-side delay test. In *Proc. of VLSI Test Symposium*, pages 284–290, 1994.

[12] A. Sinha, S. Gupta, and M. Breuer. An enhanced test generator for capacitance induced crosstalk delay faults. In *Proc. of Asian Test Conf. (ATS'03)*, pages 174–177, 2003.

[13] A. Sinha, S. K. Gupta, and M. A. Breuer. Validation and test issues related to noise induced by parasitic inductances of vlsi interconnects. *IEEE Trans. of Adv. Packaging*, pages 329–339, 2002.

[14] M. Tehranipour, N. Ahmed, and M. Nourani. Multiple transition model and enhanced boundaryscan architecture to test interconnects for signal integrity. In *Proc. of Intl. Conf. on Computer Design (ICCD'03)*, 2003.

[15] M. Tehranipour, N. Ahmed, and M. Nourani. Testing soc interconnects for signal integrity using boundary scan. In *Proc. of IEEE VLSI Test Symposium (VTS'03)*, pages 158–163, 2003.

[16] M. Tehranipour, N. Ahmed, and M. Nourani. Testing soc interconnects for signal integrity using extended jtag architecture. *IEEE Trans. on Computer-Aided Design*, 23(5):800–811, May 2004.

[17] S. Yang, C. Papachristou, and M. Tabib-Azar. Improving bus test via iddt and boundary scan. In *Proc. of Design Automation Conf. (DAC'01)*, pages 307–312, 2001.