

Analog Macromodeling of Capacitive Coupling Faults in Digital Circuit Interconnects*

Aditya D. Sathe
Intel Corporation[†]
Hillsboro, Oregon
aditya.sathe@intel.com

Michael L. Bushnell
ECE Dept., Rutgers University
Piscataway, NJ 08854
bushnell@caip.rutgers.edu

Vishwani D. Agrawal
Agere Systems
Murray Hill, NJ 07974
va@agere.com

Abstract

Today's high-performance chips are clocked at several GHz, resulting in designs whose performance specifications are violated by very small defects. High stuck-fault coverage is insufficient to detect these timing failures, so we propose a new analog coupling delay fault model and analog macromodeling technique to generate tests for these faults. To our knowledge, this is the first time that analog macromodeling, along with multiple-delay sequential digital fault simulation, using differing rise and fall times for digital logic gates, effectively detected coupling timing faults.

We propose a new *Crosstalk Candidate Reduction* (CCR) algorithm, which looks at the entire set of possible signal line couplings and eliminates impossible and uninteresting couplings from the final list. On various circuits, CCR reduced the coupling candidates by 98.1%, on average. The analog macromodels eliminate errors and uncertainty about whether signals actually couple, and also avoid complete analog simulation during fault simulation, as all analog macromodels are precomputed. The analog macromodel is independent of the circuit-under-test, because it models generalized interconnect. The method efficiently handles large circuits with more than 10,000 coupling faults, while obtaining coupling fault coverages in the range of 4 to 10% on sequential circuits, and up to 33% on combinational circuits. These are the first coupling fault results for sequential circuits. Nearly all remaining coupling faults are redundant. We accurately eliminate many candidate coupling signals using an efficient time windowing technique that is of $O(\# \text{ logic gates} \times \# \text{ fanouts})$ complexity.

1 Introduction

The current trends in circuit design – increasing clock rates, decreasing feature sizes, increased number of interconnect layers and gate density, increased current density,

and higher voltage drops along the power nets – are increasing the significance of crosstalk and ground-bounce noise effects. The increasing number of metal layers is favoring crosstalk between lines because the upper levels are farther from a stable voltage reference and are therefore more susceptible to perturbations. The increasing number of transistors on a chip leads to more devices switching simultaneously, resulting in increased power supply noise. Capacitive crosstalk noise and power supply noise can lead to *noise faults*. Analysis shows that excessive noise, most of the time, leads to delay faults [1]. Traditional fault models and tests are becoming inadequate for deep submicron technologies, which have new failure modes.

The typical design validation methodologies do not ensure correct circuit operation throughout the design envelope, in the presence of noise. Some specific types of noise, in chips with small feature sizes and fast rise times, can cause completely validated chips to malfunction, as follows:

1. Crosstalk between adjacent lines in a long bus, which distorts signal waveforms [1, 2, 5, 6, 7].
2. Ground-bounce (simultaneous switching noise), which perturbs power line voltages [4, 11, 12, 14].
3. Transmission line effects, which distort waveforms in long wires due to reflections and cause ringing.

These noise effects are already wide-spread in current designs, and will only worsen in future chips.

Options for the Designer. The designer can either choose to eliminate potential errors caused by crosstalk via redesign or ignore them until post-manufacturing testing. The latter option is often taken by designers due to time-to-market issues. Test generation for crosstalk also enables more aggressive design. An ad-hoc validation methodology is not adequate for aggressive designs – *it is inadequate to view these effects merely as design issues and they will have to be dealt with as faults during manufacturing testing.*

*United States Provisional Patent filed by Rutgers University.

[†]This research was performed when Sathe was at the ECE Dept., Rutgers University.

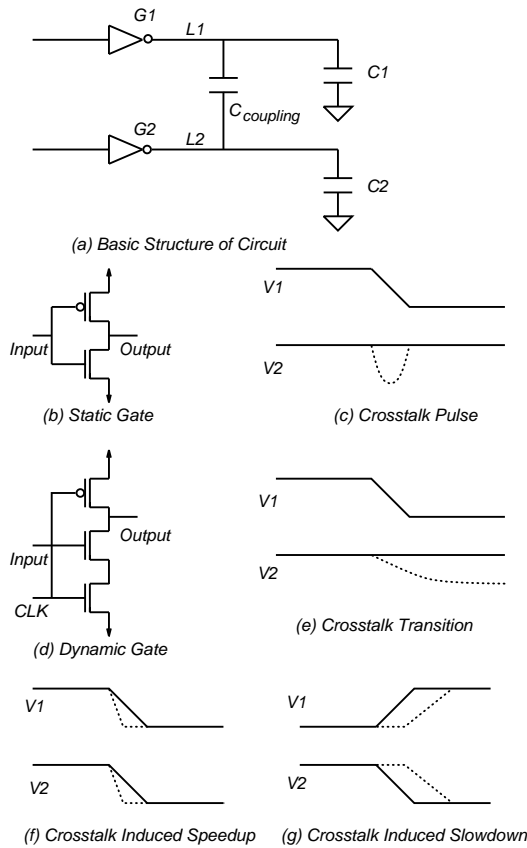


Figure 1: Crosstalk Effects.

1.1 Crosstalk Noise Effects

Interconnect on a VLSI chip has certain associated parasitic capacitances – capacitance between the interconnect and the ground plane, and capacitance between two or more neighboring interconnects. Crosstalk effects show up when the capacitance with the adjacent interconnect is significantly greater than the capacitance with the ground plane. These effects are illustrated in Figure 1 [1, 2, 4, 5, 6, 7, 8].

A *crosstalk pulse* occurs due to the coupling between a circuit line having a signal transition, and a line that is driven by the output of a static gate and is holding a steady value (see Figures 1 a, b, and c). Here the voltages of lines $L1$ and $L2$ are $V1$ and $V2$, and $G1$ and $G2$ are static inverters of the type shown in Figure 1 b. A transition on $L1$ causes a pulse on $L2$.

A *crosstalk transition* occurs due to coupling between a circuit line with a transition and a line driven to a constant value by the output of a tri-stated dynamic gate. Suppose that $G2$ is a dynamic gate of the type shown in Figure 1 d. The charge lost by the steady line, through the coupling capacitance when the other line transitions, cannot be recovered, unlike in the static case (see Figure 1 e).

Crosstalk delay occurs when both lines have transitions in the same clock cycle. If both capacitively coupled lines

have identical transitions, each transition completes sooner, leading to crosstalk delay-speedup (see Figure 1 f). If the coupled lines transition in opposite directions then each line takes longer to complete the transition, leading to crosstalk delay-slowdown (see Figure 1 g). These effects can increase the rise and fall times of signals by a factor of two or three.

1.2 Our Approach

We develop a *coupling fault model* for random logic testing of combinational and sequential circuits. The fault specification requires enumeration of physical attributes of the two lines that are coupling. The manifestation of the fault effect in the circuit – as additional delay injected on both the coupled lines – depends on the fault attributes as well as the skew between the opposite transitions on *aggressor* and *victim* lines. We have abstracted analog effects into the digital domain, through an *analog macromodel*, so that circuit simulation can be done digitally. We use simulation-based delay-fault test generation for the entire coupling fault universe for the circuit. The universe for an n -gate circuit has $n(n-1)/2$ faults. Our *Crosstalk Candidate Reduction (CCR) Algorithm* removes the faults that can never be excited/detected, based on temporal constraints, combined with critical path information, both of which are obtained from a static timing analysis of the circuit. The method achieved a 4 to 10% capacitive coupling delay fault coverage on sequential circuits with fault lists of 5000 to 8000 coupling faults. It got up to 33% fault coverage on combinational circuits. Prior work by Breuer *et al.* [8] achieved fault coverages of 40 to 50% on combinational circuits. However, their fault lists were manually generated, whereas ours are automatically generated and reduced for only possible coupling faults, and this represents a much larger fault universe than theirs.

This paper is organized as follows. Section 2 covers prior work, and Section 3 describes the analog macromodel development and lookup. Section 4 details the fault list reduction and the CCR algorithm, and presents results from simulation-based test generation. We present a theorem describing the conditions necessary for detecting coupling faults. Section 5 concludes.

2 Prior Work

Chen *et al.* [5] characterized the electrical behavior of capacitively coupled aggressor and victim lines, by deriving the transfer function of the aggressor-victim pair and subjecting them to different types of inputs. Two forms of crosstalk noise – crosstalk pulse and crosstalk delay – were analyzed. Expressions relating the slowdown (speedup) to circuit parameters, the rise/fall times of the input transitions, and the skew between the transitions are derived.

Chen *et al.* [7, 8] used a PODEM-based combinational *automatic test pattern generation* (ATPG) approach for

testing for crosstalk delay due to near simultaneous transitions in adjacent interconnects. The mixed-signal test generator computed analog properties such as noise strength (delay) and signal timings. For a specific target crosstalk coupling a vector pair was generated that created a worst-case crosstalk effect at the target. It created either a logic error or the maximum noise effect (delayed transition) at a *primary output* (PO). This algorithm is applied only to line pairs selected by the designer, based on layout knowledge, because it cannot handle all potential line pairs. The algorithm generated tests for circuits of reasonable size, but was too slow for circuits larger than c7552. It handled only combinational circuits.

Jiang and Cheng [11] analyzed the performance degradation caused by noise in power supply lines, for deep sub-micron CMOS devices. ΔI noise is caused by the change in the instantaneous current in the wire (package or on-chip interconnect) inductance and is proportional to $L \frac{dI}{dt}$. The IR drop is the resistive voltage drop caused by the instantaneous current through power and ground lines. This noise reduces the actual voltage reaching a device, which increases cell and interconnect propagation delays. They used a statistical modeling technique to evaluate the effect of power supply noise on circuit performance, in terms of delays at POs. For $0.25\mu\text{m}$ - 2.5V technology, on average, the circuit delay due to both inductive ΔI and resistive IR voltage drop was 33% with an average power supply noise of 0.27V . For $0.25\mu\text{m}$ technology, delay increased by 58% due to IR drop.

Krstic *et al.* [14] proposed a new delay test generator that also produced the maximum power supply noise conditions along the path being tested. Power supply noise is a result of the switching activity in the circuit and is highly dependent on the input vector. Their *genetic algorithm* created patterns causing high power supply noise and, thus, these tests produced significantly longer path delays.

Margolese and Ferguson [15] developed an algorithm for eliminating possible aggressor-victim net pairs for crosstalk analysis. They derived the conditions necessary for a crosstalk induced pulse to propagate into a flip-flop and developed an algorithm to reduce the number of candidate pairs. They do not consider clock lines or asynchronous signals as potential aggressors or victims.

Keller *et al.* [13] reduced the coupling fault list for crosstalk induced delay faults significantly, by using circuit topological and timing information. They assumed that the delay introduced by the coupling fault is always one gate delay and that individual gate delays are unity. This may be unrealistic, because of the lack of any analog information or real values of gate delays.

3 Analog Macromodeling

We have used a new analog macromodeling approach, along with a multiple delay digital fault simulator, for test gen-

eration for coupling faults [16]. The macromodel enables us to use an essentially *digital* simulation environment, to simulate the effects of coupling faults, which are *analog* phenomena. The additional local delay appearing on an interconnect, resulting from a coupling fault between two interconnects, can be obtained from the precomputed analog macromodel. We limit our research to crosstalk *delays*.

3.1 Interconnect Modeling for Timing Evaluation

The propagation of a signal along a wire depends on many factors, including the distributed resistance and capacitance of the wire, the driving source impedance, and the load impedance. For deep submicron technology ($\lambda \leq 0.25\mu\text{m}$) the interconnect delay dominates gate delay.

The distributed RC model (Figure 2) is a fairly accurate way to model interconnects [9]. The interconnect delay indicated by the model depends on the number of ladder RC sections, and accuracy increases as the number of sections increases. When the number of basic cells is sufficient, the delay reaches a constant value.

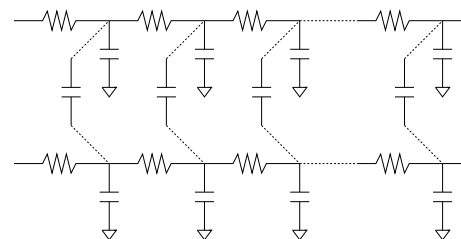


Figure 2: RC Model to Simulate Fault Effect.

3.2 Evaluation of Coupling Fault Effects

The evaluation of coupling fault effects is essentially an evaluation of the slowdown due to crosstalk delay. If two capacitively coupled lines have opposite transitions, as in Figure 3, then both lines take longer to complete the respective transitions.

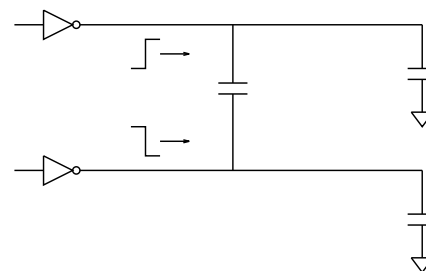


Figure 3: Coupled Interconnects.

The additional delay that is introduced in both these interconnects depends on the following factors:

1. Rise/fall time of the signals on both interconnects
2. Coupling capacitance per unit length between the interconnects
3. Length that the two interconnects run parallel/couple
4. Time skew between the opposite transitions
5. RC properties of the interconnects.

When these parameters are known, the interconnects can be modeled as described earlier as RC sections, with additional coupling capacitances. This model can be simulated to get the interconnect signal propagation time.

We used the resistance per unit length, and capacitance per unit length values for the $0.25\mu\text{m}$ *Taiwan Semiconductor Manufacturing Company* (TSMC) process, to create the RC model for developing the macromodel. For a given value of coupling capacitance between two interconnects, and a given length, we simulated the model by varying the time skew between the opposite transitions. All other parameters remaining the same, when skew is varied, we observe the maximum delay for simultaneous opposite transitions, and as the skew is increased, the delay reduces as shown by Figure 4. Similar curves can be plotted using some more typical values of capacitance per unit length, corresponding to the coupling capacitance values for parallel interconnects on each adjacent pair of layers. The entire set of data points is organized as a lookup table.

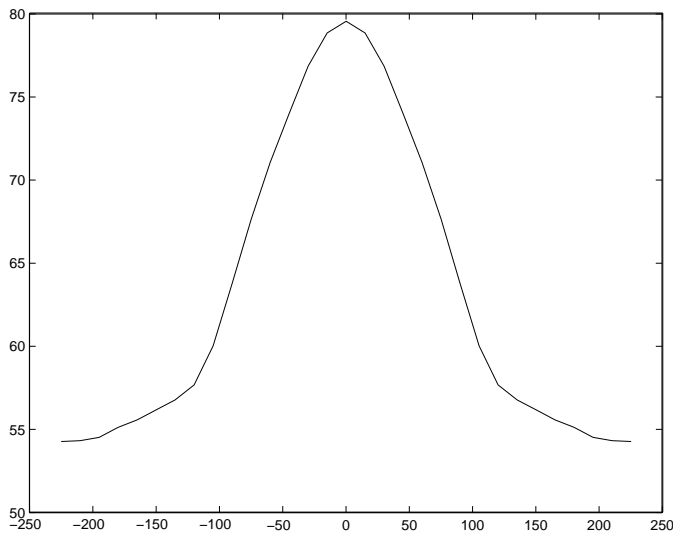


Figure 4: Delay vs. Skew Graph Obtained from the Macromodel. All times are in pico-seconds.

From the simulations we observed a timing window such that, if opposite transitions occurred in this time frame, there was considerable delay due to the coupling capacitance. This window is used later in the multiple delay fault

simulator. If opposite transitions do not occur on the coupled pair or both transitions occur further apart than the time window width, the fault is not excited, and this coupling has no effect on overall delay. If opposite transitions do occur on the coupled pair within this time window, then the macromodel lookup table provides the pre-computed additional delay, for the given fault parameters.

4 Fault Simulation

4.1 Fault Model

We have used a new type of *coupling fault model* in this work. Every pair of interconnects that are likely to couple in the chip layout is incorporated into the coupling fault list, as a coupling fault. This is quite different from the other fault models traditionally used for test generation – the stuck fault model and the path delay fault model.

The coupling fault model does not have an implicit fault list, because it is completely layout dependent. Multiple layout implementations of the same netlist could have different coupling fault lists generated, depending on the proximity of specific interconnect pairs. The circuit extractor would be used for parasitic extraction. The coupling fault list is generated after considering the values of parasitic capacitances between interconnect pairs. The fault list is stored with the following parameters for each fault:

1. First coupling gate (stem or fanout)
2. Second coupling gate (stem or fanout)
3. Type of coupling (resistive/capacitive/inductive). Only capacitive coupling has been considered here
4. Capacitance per unit length
5. Length of parallel run
6. Layers of first and second interconnects.

We assume only one aggressor and one victim for each coupling fault. This analog macromodeling approach can be generalized to multiple aggressors and victims, even with multiple types of couplings (R, L, or C). However, there will be an increased computation because the number of potential couplings for the *Coupling Candidate Reduction* (CCR) algorithm (described later) to process will significantly increase. The method can also be generalized to allow different coupling faults to be active simultaneously along the same victim path. Again, the main penalty will be the increased number of potential couplings for the CCR algorithm to process.

4.2 Fault List Generation

Given a circuit netlist, all pairs of interconnects are potential crosstalk candidates. The pairs that would actually couple capacitively depend on the layout geometry, and other physical design aspects. In the absence of layout information, we generate tests for coupling faults between all possible interconnect pairs. The tests for potential coupling faults in any specific layout implementation of a design can then be drawn from this test vector super-set.

Accordingly, in a n -gate circuit, the number of coupling faults to consider would be $n(n - 1)/2$. However, only a fraction of these coupling faults needs to be considered as:

1. Some of the coupling faults cannot be excited, because:
 - (a) Opposite transitions cannot be generated on the interconnect pair in the desired timing interval.
 - (b) Boolean conditions do not permit opposite transitions on the coupling interconnects.
2. The fault is excited but the fault effect does not reach the output, because:
 - (a) Some of the excited faults may lie on a path with a large slack, so the slack absorbs the additional injected delay. The delayed signal arrives well ahead of the clock, and the fault is not detected.
 - (b) Boolean conditions that excite the fault also block the delay effect propagation to POs and flip-flops.

We can reduce the fault list considerably, by checking for the occurrence of conditions in 1a and 2a. We characterize each netlist node with the following parameters, through a two-pass technique:

1. The timing window in which the output of each gate (node) can change: maximum and minimum values.
2. The length of the longest path from each node to any PO or flip-flop.

This involves static timing analysis of the circuit. The first pass is similar to Hitchcock's approach [17]. We traverse the netlist starting from the *primary inputs* (PIs) and flip-flop outputs, until a PO or flip-flop input is reached. Since the netlist is levelized, we are sure of having visited the predecessors of a given node before visiting the node itself. Each node is tagged with the minimum and maximum times at which a transition can occur at the node output. From the maximum transition time values of the POs and the flip-flop inputs, we get the length of the circuit critical path. There is no path *enumeration* required.

The second pass through the netlist is a backward pass. We start from the POs and flip-flop inputs, until we reach the circuit PIs or flip-flop outputs. We tag each node with the length of the longest path from that node input to any PO or flip-flop input. For each node, we select the fanout

branch that gives the longest path to a PO. To this delay to a PO from that node, we add the maximum transition time for that gate. After these two passes are done, we can reduce the fault list size.

For every pair of possible coupling nodes, we compare the switching timing windows. If there is an overlap between the two windows for a pair of nodes, these gates may have opposite transitions separated by a time less than a maximum allowed value. This is the first condition that a coupling fault must satisfy, to be included in the fault list.

We need to ensure that a coupling node lies on a critical or near-critical path, if the delay effect at the output is to miss the clock. For any node in the netlist, we know the maximum time taken from the input to the node and also the maximum time taken from this node to an output. We add these two maximum values to get the length of the longest path that this gate lies on. If this is larger than a *critical threshold* then this coupling node pair satisfies the second criterion and is put in the fault list.

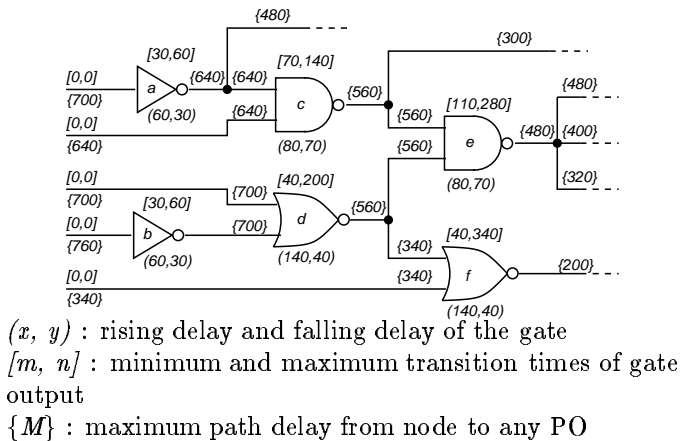


Figure 5: Timing Data at Each Node after Both Passes.

Figure 5 shows the information collected from the two passes of the netlist. Thus, we add to the coupling fault list those node pairs that have their signal transition windows overlapping, and such that at least one of the nodes from the pair lies on a critical path.

Critical path lengths can be identified from the POs as well as the PIs. The maximum value of the *maximum gate transition time* at the POs corresponds to the critical path length. The maximum value of the *maximum time to PO* at the PIs also gives an identical value of critical path length. In the case of the circuit in Figure 5 the critical path length is 760, as seen from the PI nets. We can now determine the maximum length of the path on which a gate lies, by simply adding the two parameters associated with each gate: (maximum time from PIs to this gate + maximum time to POs).

Crosstalk Candidate Reduction (CCR) Algorithm

1. Assign gate delays from simulation values
2. Forward pass over levelized netlist
 - (a) For each level
For each gate, if not visited and with all predecessors visited:
 - i. min-transition-time of gate =
 $\min(\min\text{-transition-times of fanin gates}) + \min(\text{rise/fall gate delays})$
 - ii. max-transition-time of gate =
 $\max(\max\text{-transition-times of fanin gates}) + \max(\text{rise/fall gate delays})$
 - iii. Mark the gate as visited
3. Reset the visited tags of all gates
4. Backward pass over levelized netlist
 - (a) For each level
For each gate, if not visited and with all successors visited:
 - i. maximum-delay-to-PO of gate =
 $\max(\max\text{-delays-to-PO of all gate fanouts}) + \max(\text{gate rise/fall delays})$
 - ii. Mark this gate as visited
5. Take the maximum of the maximum transition times of all gates and assign that as the critical path length
6. Calculate critical threshold from the critical path length
7. For (gate 1 = 1 to total-gates - 1) and (gate 2 = gate 1 + 1 to total-gates)
 - (a) Find if gate 1 is on a critical path (i.e., length greater than the critical threshold). Add the parameters associated with the gate to get the maximum length of the path on which gate 1 lies: (maximum-time-to-PO - greater of the rise/fall delays) + max transition time of gate 1
 - (b) Increment gate 1 index until the longest path on which gate 1 lies is greater than the critical threshold. A gate on the critical path has been identified (victim). Now check which gates (gate 2) can couple with this gate.
 - (c) Check for timing window overlap of gates 1 & 2
 - (d) If overlap, add the gate 1 - gate 2 pair to the crosstalk candidate fault list and increment gate 2 and repeat, else increment gate 2 and repeat.

Table 1: Fault List Reduction after the Case 1 Experiment.

Ckt.	Potential Coupling Faults	# Coupling Faults Case 1	Crit. Path Delay	Critical Threshold Value (ps)
c17	78	72	26	26
c432	20,503	12,760	154	154
c880	109,746	21,474	246	246
c3540	1,514,670	76,279	557	557
s27	153	86	87	87
s298	10,011	1206	121	121
s344	18,915	3057	233	233
s349	19,110	3077	233	233
s400	19,110	3593	134	134
s641	104,196	27,376	633	633

Table 2: Fault List Reduction after the Case 2 Experiment.

Ckt.	Potential Coupling Faults	# Coupling Faults Case 2	Crit. Path Delay	Critical Threshold Value (ps)
c17	78	72	26	21
c432	20,503	12,760	154	149
c880	109,746	26,576	246	241
c3540	1,514,670	198,484	557	552
s27	153	105	87	82
s298	10,011	1206	121	116
s344	18,915	3057	233	228
s349	19,110	3077	233	228
s400	19,110	3593	134	129
s641	104,196	27,376	633	628

Clock Line Coupling. We do not consider interconnects coupling with clock lines, or clock lines coupling with other clock lines, for test generation because the clock is always active. Hence, the delay effects due to clock coupling would show up in any test set that tests for other faults.

Tables 1 and 2 show the fault list reduction that is obtained by applying the above crosstalk candidate reduction algorithm. *Critical Path Delay* gives the circuit critical path delay time in ps. The victim line has to be on a path of delay greater than the *Critical Threshold Value* to be a potential victim of coupling to any other path. The two cases impose different constraints on the length on the path on which the crosstalk candidate gate lies. In Case 1 we take victim gates lying on the critical path. The *critical threshold* here equals the critical path length. In Case 2, the critical threshold is (critical path length - maximum injected delay). This allows victims that lie on paths marginally shorter than the critical path to become possible crosstalk candidates.

However, some of the coupling faults cannot be excited because: (a) Opposite transitions cannot be generated on the interconnect pair involved in the desired timing inter-

val, or (b) Boolean conditions do not permit opposite transitions on the interconnect pair involved. Also, an excited fault may not propagate to the output, as the Boolean conditions that excite the fault and propagate it are in conflict.

In comparison with the work of Keller *et al.* [13], our crosstalk candidate reduction algorithm, which is based on actual analog delay simulations of gates in the TSMC 0.25 μ process, uses the values of timing windows derived through actual analog simulations. It is more realistic.

4.3 Fault Simulator

We used a *multiple-delay* fault simulator using 3 logic values. Signals experience two types of delays. The time interval between an input change and the output change of a gate is called the *inertial* or *switching delay*. The time interval between the generation of a signal transition at a gate output and its arrival at the input of a fanout gate is known as the *propagation delay* or *transport delay*.

Switching Delay. Figure 6 shows the response of a 2 input NAND gate to a given set of inputs. The third waveform is what an analog simulator would output for the given gate inputs. After input *a* switches to logic 1, both the *n*-transistors of the NAND gate turn ON and the load capacitance of the gate gets a discharge path to ground. After the discharge has started input *b* switches to logic 0, switching off the path to ground and switching ON a path to V_{DD} . The output then goes back to V_{DD} .

Digital simulators produce different responses in the transient region, depending upon the delay modeling [3]. We used the *multiple-delay* model. Each gate is assigned a rise delay (d_r) and fall delay (d_f). We simulated each type of gate using the Cadence Spectre simulator to obtain the rise and fall delays. The multiple-delay model approximates the delay of an analog waveform. A zero-delay or unit-delay simulator would have produced a glitch in the output, for the inputs as in Figure 6. Our simulator filters out glitches which are narrower than the gate delay. This is crucial to correct simulation of coupling delay faults.

Propagation Delay. We consider fanout nets as circuit elements. A fanout net simply transmits the signal from the stem to each branch with some delay. Thus, a net with *n* branches will have *n* delays, one for each fanout.

Our static compactor uses *reverse order restoration* (ROR) as proposed by Guo *et al.* [10] to compact test vectors detecting capacitive coupling delay faults. Restored subsequences are used to create a new test sequence, and are placed in the new test sequence in reverse order of their inclusion in the applied test set. ROR saves simulation time, without sacrificing fault coverage.

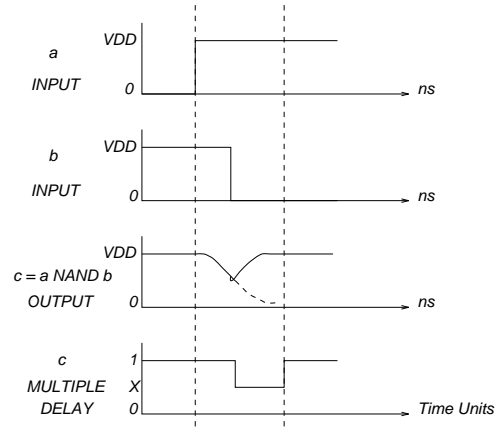


Figure 6: Timing Model of a NAND Gate.

4.3.1 Simulation-Based Test Generation Results

Table 3 shows the coupling fault coverage obtained for a few benchmark circuits. The Critical Path Delay and Critical Threshold values for the sequential circuits in these experiments were the same as for Tables 1 and 2. However, for the two combinational circuits, a different condition was applied, namely, that both aggressor and victim lines had to lie on different critical paths. 10,000 random vectors were applied in each case. *Faults detected* are those that cause a logic error at strobe time at the POs. This may be either because the signal on one of the logic paths to the POs got delayed, resulting in a wrong value getting strobed, or a wrong value being latched in a flip-flop due to delay in a signal feeding a flip-flop in a previous clock cycle. *Faults Excited* are those for which opposite transitions occurred on the coupling interconnect pair, within the desired time interval. *Fault Effect at PO* gives the number of faults for which the delay effect due to the coupling fault appeared at a PO. A detected fault does not set this flag. If we have a detected fault at one PO and a delay effect at another PO, then both fault-detected and delay-effect-at-PO flags are set. This may happen for a few faults. The fault coverages are low, even after the CCR algorithm is applied, because we use a very liberal timing window for the paths, allowing many signals to couple with a victim by the time the victim's signal propagates to a PO. Also, a basis of this method is static timing analysis, which does not consider whether paths are *false*. Many paths that are candidates for coupling can never couple, because Boolean conditions prevent coupling fault sensitization or propagation.

This method also compacts the coupling fault test patterns, using a multiple delay fault simulator and *reverse order restoration* [10] of patterns after they have been dropped from the test sequence. 2,000 vectors were simulated at a time, and 5 passes of random vector generation and compaction were run, leading to 10,000 vectors origi-

nally, and far fewer than that in the compacted sequence. Since this method is based on random pattern generation, and compaction after fault simulation, then other vector sets, computed for other fault models, can also be added to the vector stream that is considered for coupling fault test generation.

Table 3: Results of Fault Simulation on Benchmarks.

Ckt	# Coupling Faults	Faults Det	Faults Exc	Flt Eff at PO	Fault Cov. %
c17	52	34	52	51	65.3
c432	8736	2949	8690	8534	33.7
s27	132	40	94	66	30.3
s298	1206	128	705	112	10.6
s344	3057	124	2371	199	4.1
s349	3077	124	2376	198	4.0

¹10,000 vectors simulated in each case

Examples of Success and Failure of Test Generation

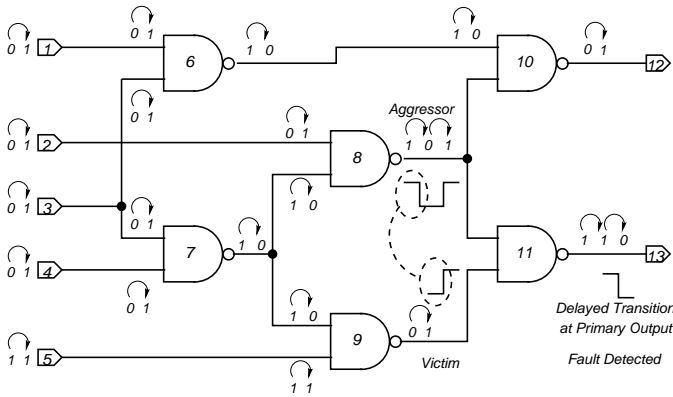


Figure 7: Successful Detection of Coupling Fault.

Figure 7 shows the successful detection of a coupling fault between the output stem of gate 8 and the output stem of gate 9. Here a *glitch* causes the coupling delay fault. We want a transition along a critical path to be delayed due to coupling, for successful fault detection. At the same time, we want opposite transitions on the coupled lines, in a specific timing window. The given vector pair does just that. A glitch generated at the output of gate 8 couples with a rising transition at the output of gate 9 (coming much later, but still within the timing window). This delays the rising transition at the output of gate 9 and results in a delayed falling transition at the output of gate 11.

Figure 8 shows an undetectable or *redundant* coupling fault. The coupling fault is between the stem of *primary inputs* (PIs) 3 and 4. The non-detection of this coupling fault by our test generator is not a failure of test generation. There does not exist a vector pair that can excite this fault and also propagate the delayed edge to a PO. The same

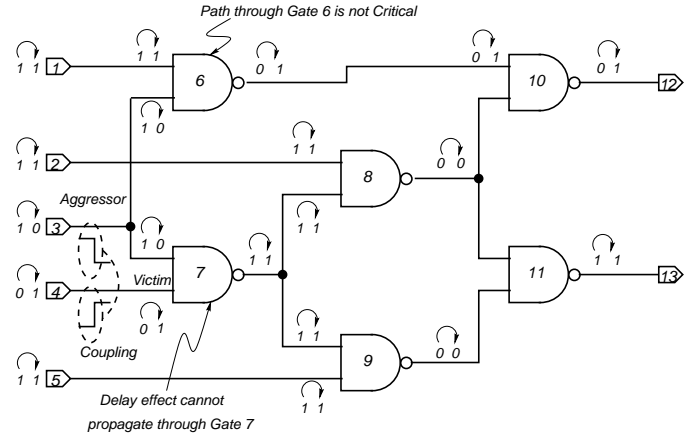


Figure 8: Undetectable Coupling Fault.

argument holds true even if the aggressor and victim roles were interchanged and the transition direction were to be reversed. Hence, this coupling fault is *redundant*.

4.3.2 Existence Proof for Coupling Fault Tests

The Crosstalk Candidate Reduction algorithm uses the following theorem to reduce the list of candidate coupling faults by 91.8% for the Case 1 experiments and 84.8% for the Case 2 experiments.

Theorem 1 Conditions necessary for detecting a capacitive coupling fault are:

1. The aggressor and victim lines should have transitions in opposite directions.
2. Transition time windows for the aggressor and victim lines should overlap, for Condition 1 to happen.
3. The victim line is on one of the critical paths, or, on a line that is shorter than the critical path, by an amount equal to the maximum delay injected at the fault site by the fault mechanism.
4. The fault-detecting-vector must be able to sensitize the critical-path segment from the victim to the PO, i.e., the stuck-at-0 (rising transition at the victim) or stuck-at-1 (falling transition at the victim) fault at the victim should be testable.

The first three conditions above are applied by the CCR algorithm. There is no way to apply the fourth condition, since we are using a static timing analyzer. The fourth condition can only be enforced by a complete ATPG program. Therefore, our fault coverages are pessimistic, as the real fault efficiency is much higher, because many of the couplings listed as testable actually are impossible to sensitize or propagate.

The above conditions are *necessary* but not *sufficient* to guarantee coupling fault detection, because, in spite of all

four conditions being satisfied, the additional delay injected might be insufficient to cause the delay effect at the PO to exceed the clock period.

5 Conclusion

We developed a new capacitive coupling fault model, and a test generation mechanism, to exhaustively test any physical implementation of a given circuit netlist. We consider the entire fault universe of capacitive coupling faults as potential candidates for test generation. We also developed a method to filter out redundant coupling faults, which can never affect the circuit operation, through the *Crosstalk Candidate Reduction* (CCR) algorithm. We developed an analog macromodel, which was used to incorporate analog effects of interconnect-coupling, into our multiple-delay sequential digital fault simulator. The simulation-based test generator works for both combinational and sequential circuits. We also stated a theorem specifying the conditions necessary for a capacitive coupling fault to be detected. The method achieved a 4 to 10.6% capacitive coupling delay fault coverage on three sequential circuits with fault lists of 5000 to 8000 coupling faults, which were generated using our CCR algorithm. These are the first coupling fault results for sequential circuits. The method obtained up to 33.7% fault coverage on combinational circuits. Breuer *et al.* [8] reported fault coverages of 40 to 50% on combinational circuits. However, their fault lists were manually generated, whereas ours are automatically generated and reduced from all possible coupling faults, a much larger fault universe than their's. Analog macromodels were validated using the TSMC 0.25 μm process models.

It is generally experienced that the delay test coverage of paths drops as the path length increases, especially for sequential circuits. The objective in these cases is to test those few critical paths that are testable. In the case of coupling faults, a similar objective may prove to be useful.

The test generation times can be improved by a concurrent or parallel implementation of the capacitive coupling fault simulator, which is currently implemented as a serial fault simulator. The model can be extended to resistive bridges, self inductance, mutual inductance, and multiple combinations of these types of couplings.

References

- [1] M.A. Breuer, C. Gleason, and S.K. Gupta. New Validation and Test Problems for High Performance Deep Submicron VLSI Circuits. In *Tutorial Notes, VTS*, Apr. 1997.
- [2] M.A. Breuer and S.K. Gupta. Process Aggravated Noise (PAN): New Validation and Test Problems. In *Proc. of the ITC*, pages 914–923, Nov. 1996.
- [3] M.L. Bushnell and V.D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, Boston, 2000.

- [4] Y.S. Chang, S.K. Gupta, and M.A. Breuer. Test Generation for Maximizing Ground Bounce for Internal Circuitry with Reconvergent Fan-outs. In *Proc. of the VTS*, pages 191–200, Nov. 2001.
- [5] W. Chen, S.K. Gupta, and M.A. Breuer. Analytic Models for Crosstalk Delay and Pulse Analysis Under Non-Ideal Inputs. In *Proc. of the ITC*, pages 809–818, Nov. 1997.
- [6] W. Chen, S.K. Gupta, and M.A. Breuer. Test Generation in VLSI Circuits for Crosstalk Noise. In *Proc. of the ITC*, pages 641–650, Nov. 1998.
- [7] W. Chen, S.K. Gupta, and M.A. Breuer. Test Generation for Crosstalk-Induced Delay in Integrated Circuits. In *Proc. of the ITC*, pages 191–200, Nov. 1999.
- [8] W. Chen, S.K. Gupta, and M.A. Breuer. Test Generation for Crosstalk-Induced Faults: Framework and Computational Results. *JETTA*, 18:17–28, Feb. 2002.
- [9] D. Deschacht and E. Vanier. Accurate Modeling of Interconnects for Timing Simulation of Sub-micronic Circuits. In *Proc. of the IEEE Workshop on Signal Propagation on Interconnects*, pages 13–21, May 1997.
- [10] R. Guo, I. Pomeranz, and S.M. Reddy. On Speeding-up Vector Restoration Based Static Compaction of Test Sequences for Sequential Circuits. In *Proc. of the Asian Test Symp.*, pages 467–471, Dec. 1998.
- [11] Y.M. Jiang and K.T. Cheng. Analysis of Performance Impact Caused by Power Supply Noise in Deep Submicron Devices. In *Proc. of the DAC*, pages 760–765, June 1999.
- [12] Y.M. Jiang, K.T. Cheng, and A.C. Deng. Estimation of Maximum Power Supply Noise for Deep Submicron Designs. In *Proc. of the ISLPED*, pages 233–238, Aug. 1998.
- [13] K.J. Keller, K.K. Saluja, H. Takahashi, and Y. Takamatsu. On Reducing the Target Fault List of Crosstalk-Induced Delay Faults in Synchronous Sequential Circuits. In *Proc. of the ITC*, pages 568–577, Nov. 2001.
- [14] A. Krstic, Y.M. Jiang, and K.T. Cheng. Delay Testing Considering Power Supply Noise Effects. In *Proc. of the ITC*, pages 181–190, Nov. 1999.
- [15] M.A. Margolese and F.J. Ferguson. Using Temporal Constraints for Eliminating Crosstalk Candidates for Design and Test. In *Proc. of the VTS*, pages 80–85, May 1999.
- [16] A. Sathe, M.L. Bushnell, and V.D. Agrawal. Analog Macromodeling of Capacitive Coupling Faults in Digital Circuit Interconnects. In *Proc. of the North Atlantic Test Workshop*, pages 56–66, May 2002.
- [17] R.B. Hitchcock Sr. Timing Verification and the Timing Analysis Program. In *Proc. of the DAC*, pages 594–604, 1982.