False Coupling Interactions in Static Timing Analysis

Ravishankar Arunachalam
IBM Corporation
11400 Burnet Road
Austin TX 78758
ravaru@austin.ibm.com

Ronald D. Blanton
Carnegie Mellon University
5000 Forbes Avenue,
Pittsburgh PA 15213
blanton@ece.cmu.edu

Lawrence T. Pileggi
Carnegie Mellon University
5000 Forbes Avenue,
Pittsburgh PA 15213
pileggi@ece.cmu.edu

ABSTRACT

Neighboring line switching can contribute to a large portion of the delay of a line for today's deep submicron designs. In order to avoid excessive conservatism in static timing analysis, it is important to determine if aggressor lines can potentially switch simultaneously with the victim. In this paper, we present a comprehensive ATPG-based approach that uses functional information to identify valid interactions between coupled lines. Our algorithm accounts for glitches on aggressors that can be caused by static and dynamic hazards in the circuit. We present results on several benchmark circuits that show the value of considering functional information to reduce the conservatism associated with worst-case coupled line switching assumptions during static timing analysis.

1. INTRODUCTION AND MOTIVATION

It is evident in today's deep-submicron circuits technologies that intra-layer coupling capacitance is a dominant part of the total parasitic capacitance. Coupling capacitance affects performance via noise and delay. Noise is observed when a coupled line switches and causes the voltage on an otherwise static line to fluctuate. A potentially more important effect of coupling is the impact on signal delay. The delay degradation can affect the performance of a circuit even when the noise does not cause a failure. In the presence of coupling, the delay of a particular signal depends not only on the input signal, the strength of the gates and the RC interconnect load, but also on the activity of coupled lines. When a coupled line switches in the opposite direction, the signal delay increases, and when it switches in the same direction, the signal delay decreases. The affected line is usually referred to as the "victim", and the coupled line as the "aggressor".

Various approaches have been presented to compute the gate and interconnect delays in the presence of coupling. These models are usually incorporated in static timing analysis with worst-case assumptions for aggressor switching. This is because STA performs an input-pattern independent analysis and the exact switching waveforms at aggressor lines are not known. This results in excessive conservatism in timing analysis and valuable designer resources could be wasted in speeding up circuits, when the circuit actually meets timing specifications. In order to reduce this conservatism, it is important to determine whether the aggressors and victim can potentially switch (nearly) simultaneously.

There are two reasons why coupled lines may not switch together - temporal isolation and functional isolation. In STA, timing characteristics are usually specified by arrival time windows, which represent the earliest and latest times that a signal can undergo a transition (either rising or falling). If a victim and an aggressor have arrival time windows that do not overlap with one another, the two lines are "temporally isolated" from each other. Techniques to identify temporal isolation and reduce pessimism in delay computation have been presented in [1][6][11].

Functional isolation, on the other hand, arises due to the logical relationships between gates. If the logic dictates, for example, that the victim and aggressor can never switch in opposite directions in the same clock cycle, then we have a "false coupling interaction". It is important to determine the impact of functional isolation, even at the cost of extra CPU time, since it is unpredictable and largely design-dependent. A related problem encountered in static timing analysis is the well known false-path problem, which has been extensively researched. However, the problem of functional relationships of coupled signals is different in nature and has been addressed only recently for the noise problem[2][8][9].

In this paper, we present a comprehensive methodology to identify valid interactions between coupled lines, including a thorough analysis of glitches that can be caused by potential hazards. We use an ATPG approach with a six-valued algebra that includes static and dynamic hazards. Results are shown for benchmark circuits to consider the importance of modeling functional information to reduce the conservatism of most coupled delay models. The remainder of the paper is organized as follows. In Section 2, we formulate the problem and identify the conditions required for a valid coupling interaction. In Section 3, we define static and dynamic sensitization criteria for these interactions. Our ATPG-based approach is presented in Section 4, and results are presented in Section 5. We conclude in Section 6.

2. PROBLEM FORMULATION

We limit the problem of finding false interactions to combinational circuits only. Sequential circuits can be separated at flip-flops and the flip-flop outputs can be treated as primary inputs, as is usually done in STA. A victim can have, and usually has, multiple aggressors coupled to it. Coupling interactions between the victim and aggressors can be categorized into two types: delay-

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.

increase and delay-decrease. The delay-increase interaction occurs when aggressors switch in the opposite direction to that of the victim, and vice-versa. In this work, we will not consider the noise problem caused by coupling.

We denote a rising transition on a line by $\uparrow$ and a falling transition by $\downarrow$. Since we are dealing with transitions on signal lines rather than just single values, there are two different input vectors for which the circuit needs to be analyzed. Note that this implies that we use the 2-vector transition mode[9] for analysis. This is different from the floating mode in [3] whereby the initial state of the circuit is assumed to be known. We denote the two vectors by v_1, v_2 and the clock period of the circuit by T. $s(t;v_1,v_2)$ represents the "state" of a particular signal, t time units after vector v_2 is applied. Though the state is defined after v_2 is applied, it depends on the value of the signal before v_2, and hence v_1. Possible values of $s(t;v_1,v_2)$ are 0, 1, $\uparrow$, $\downarrow$ for $0 < t < T$, and 0, 1 for $t=0$ and $t=T$. We assume the circuit reaches steady state after v_1 is applied and before v_2 is applied.

Definition 1: A system S is defined as a set of lines with one victim V and n aggressors A_1, A_2...A_n that are coupled to it.

$$S = \{V, A_1, A_2...A_n\}$$

Definition 2: The delay-increase interaction of a system $S=\{V,A_1,A_2,..A_n\}$ is *valid* only if:

$$\exists\ v_1, v_2, t',\ 0<t'<T,$$
$$V(t';v_1,v_2) =\uparrow,\ A_i(t';v_1,v_2) = \downarrow,\ i=1, 2..n \quad (1)$$

or

$$V(t';v_1,v_2) =\downarrow,\ A_i(t';v_1,v_2) =\uparrow,\ i=1, 2..n \quad (2)$$

Similarly, the delay-decrease interaction is *valid* only if:

$$\exists\ v_1, v_2, t',\ 0<t'<T,$$
$$V(t';v_1,v_2) =\uparrow,\ A_i(t';v_1,v_2) =\uparrow,\ i=1, 2..n \quad (3)$$

or

$$V(t';v_1,v_2) =\downarrow,\ A_i(t';v_1,v_2) = \downarrow,\ i=1, 2..n \quad (4)$$

For all aggressors to contribute to the delay of the victim, the corresponding coupling interaction must be valid. It must be noted that an interaction which is invalid for a system with one victim and n aggressors can be valid for a subsystem that includes only a subset of the aggressors.

It is not necessary to compute the value of each signal for each time point $t\in [0,T]$ in order to determine whether a coupling interaction is valid. The arrival time windows can be used to determine whether a t' can possibly exist. This process of temporal pruning identifies those systems where the aggressors and victim can "potentially" switch simultaneously. It should be noted that STA can never ensure that they switch together, since it performs an input pattern independent analysis. We can now remove the simultaneity constraint from the conditions given above, which means the conditions (1) and (2) can be relaxed to:

$$\exists\ v_1, v_2, t', t_i,\ 0<t'<T,\ 0<t_i<T,\ i=1, 2..n$$
$$V(t';v_1,v_2) =\uparrow,\ A_i(t_i;v_1,v_2) = \downarrow,\ i=1, 2..n \quad (5)$$

or

$$V(t';v_1,v_2) =\downarrow,\ A_i(t_i;v_1,v_2) = \uparrow,\ i=1, 2..n \quad (6)$$

The functional and temporal aspects of the problem are separated from each other in this manner. Even though we remove the simultaneity constraint, we only incur an error on the conservative side. If it so happens that interactions classified as valid under the above conditions cannot take place in reality, then the victim delays are overestimated. This is acceptable because we are only interested in bounds for the delays and arrival times in STA. Our goal in identifying false interactions is to still bound the actual delays while ensuring that the bounds are as tight as possible.

3. STATIC VS. DYNAMIC SENSITIZATION

Definition 3: The delay-increase coupling interaction of a system is *statically sensitizable* iff:

$$\exists\ v_1, v_2:$$
$$V(0;v_1,v_2)=0,\ V(T;v_1,v_2)=1,$$
$$A_i(0;v_1,v_2)=1,\ A_i(T;v_1,v_2)=0,\ i=1,2..n \quad (7)$$

or

$$V(0;v_1,v_2)=1,\ V(T;v_1,v_2)=0,$$
$$A_i(0;v_1,v_2)=0,\ A_i(T;v_1,v_2)=1,\ i=1,2..n \quad (8)$$

A similar definition holds for the delay-decrease interaction.

Static sensitizability implies that there exist input vectors v_1 and v2 that cause transitions on each of the victim and aggressor lines. An interaction that is not statically sensitizable is "statically unsensitizable".

Theorem 1: If (7) holds for an input vector pair (v_1,v_2), then (8) holds for the vector pair (v_2,v_1).

This will reduce the number of cases that must be analyzed by half for each system.

Example 1: Consider the circuit shown in Figure 2. Consider the delay-increase interaction with $V=\uparrow$, and $A=\downarrow$. For this interaction to be statically sensitizable, there must be a (v_1,v_2) such that $V(T;v_1,v_2)=1$ and $A(T;v_1,v_2)=0$. Justifying these values requires $a(T;v_1,v_2)=0$, $e(T;v_1,v_2)=0$, $c(T;v_1,v_2)=1$, $f(T;v_1,v_2)=0$ and $A(T;v_1,v_2)=1$. But this leads to a contradiction on A. Hence the delay-increase interaction for this system is statically unsensitizable.

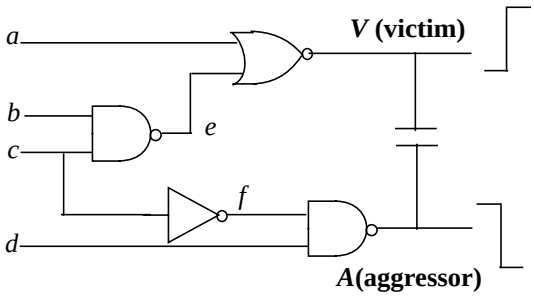

FIGURE 1: Static sensitization criterion for a coupling interaction

On the other hand, the delay-decrease interaction is sensitizable with the input vectors v_1={0,1,1,1} and v_2={0,1,0,1}.

Theorem 2: A statically sensitizable interaction is always valid.

Proof: Since the state of the victim and each aggressor signal changes after v_2 is applied, they must necessarily undergo transitions in between. Hence a statically sensitizable delay-increase interaction has to satisfy (5) or (6). A similar argument applies for the delay-decrease interaction.

Definition 4: The delay-increase interaction for a system S is said to be *dynamically sensitizable* iff:

$$\begin{gathered}\exists\, v_1, v_2: \\ V(0;v_1,v_2)=0,\ V(T;v_1,v_2)=1, \\ \exists\, D \subset \{1,2..n\},\ D \neq \phi,\ d \in D,\ t_d,\ 0<t_d<T: \\ A_d(0;v_1,v_2) = A_d(T;v_1,v_2) \neq A_d(t_d;v_1,v_2) \\ A_i(0;v_1,v2)=1,\ A_i(T;v_1,v_2)=0,\ i \notin D \\ or \\ \exists\, v_1, v_2: \\ V(T;v_1,v_2)=1,\ V(T;v_1,v_2)=0, \\ \exists\, D \subset \{1,2..n\},\ D \neq \phi,\ d \in D,\ t_d,\ 0<t_d<T: \\ A_d(0;v_1,v_2) = A_d(T;v_1,v_2) \neq A_d(t_d;v_1,v_2) \\ A_i(0;v_1,v_2)=0,\ A_i(T;v_1,v_2)=1,\ i \notin D\end{gathered} \quad (9), (10)$$

The set D contains the list of all indices d such that A_d is a "dynamic" aggressor. The signal value on a dynamic aggressor A_d is the same at t=0 and t=T. However, because of a glitch on the line, it toggles state in between. It does not matter whether the glitch is due to a static 0-hazard or a static 1-hazard. In either case, the aggressor transitions in both directions, and hence can cause a valid interaction with the victim. The condition for a dynamically sensitizable interaction is that at least one aggressor must satisfy the above property.

Theorem 3: A dynamically sensitizable interaction is a valid one.

Proof: Without loss of generality, we consider only the delay-increase interaction described by (9). If (9) is satisfied, then it implies that V has to be $\uparrow$ at some time point. Similarly, for all $i \notin D$, A_i=$\downarrow$ at some point. For $d \in D$, since a glitch occurs on the A_d, it must have both $\downarrow$ and $\uparrow$. Hence (1) is satisfied. A similar argument holds if (10) is true.

Example 2: Consider the same circuit presented in Figure 1, but with the delay assignments as shown in Figure 2. Consider the input vectors v_1=(0,1,1,1) and v_2=(0,1,0,0). If we simulate these vectors, we get:

$V(T;v_1,v_2)=0$, $V(T;v_1,v_2)=1$, $A(0;v_1,v_2)=A(T;v_1,v_2)=1$.

However, because of the delay assignments shown above, $A(2;v_1,v_2)=\downarrow$, as shown in Figure 2. The glitch on the aggressor can cause an increase in the victim's delay.

A valid coupling interaction can be either statically sensitizable or dynamically sensitizable or both. We present an ATPG-based approach to identify valid coupling interactions.

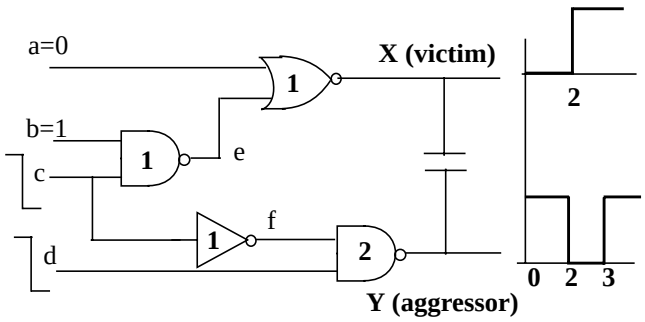

FIGURE 2: Dynamically sensitizable coupling interaction

4. ATPG APPROACH

Since valid interactions can be dynamically sensitizable, any algorithm to detect valid interactions has to include glitches caused by static and dynamic hazards. We choose ATPG techniques because they can easily incorporate such elements in an algebra. We present a six-valued algebra that is used in conjunction with our ATPG approach. The values of the algebra incorporate both input vectors v_1 and v_2.

4.1 Six-valued Algebra

The elements of our algebra are shown in Table 1.

TABLE 1: Six-valued algebra

Signal	Description
s0	Steady 0-signal
s1	Steady 1-signal
p0	Transition to 0
p1	Transition to 1
g0	Glitch on a 0 signal
g1	Glitch on a 1 signal

Some examples of signals and how they are propagated using this algebra are shown in Figure 3. A p0 (transition to 0) signal on a line

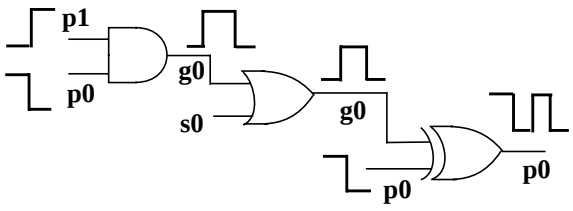

FIGURE 3: Signal propagation using the six-valued algebra

L implies that $L(0;v_1,v_2)$ = 1 and $L(T;v_1,v_2)$=0. Similarly, a g0 signal on a line L implies that $L(0;v_1,v_2)$=0, $L(T;v_1,v_2)$=0 *and* that there can potentially be a glitch on the line. It should be noted that the transitions p0 and p1 can include dynamic hazards, *i.e* signals that have one or more intermediate transitions as shown at the output of the XOR gate in Figure 3. Similarly, g0 and g1 can include multiple glitches on a line. A value of g0 on a line implies that the initial and final values are the same and that there is potential for at least one glitch.

Since the effects of both v_1 and v_2 are implicitly accounted for in the algebra, one input vector in this algebra is enough to analyze a system for valid interactions. The four possible input signals that can occur at a primary input line L are given in Table 2. We assume

TABLE 2: Possible values for primary inputs

$L(0;v_1,v_2)$	$L(T;v_1,v_2)$	Signal name
0	0	s0
1	1	s1
1	0	p0
0	1	p1

that glitches cannot occur at primary inputs. This assumption holds as long as the primary inputs are either the actual synchronous inputs to the circuit or outputs of edge-triggered flip-flops. In case they are outputs of transparent latches, they can also take the g0 and g1 values. We do not consider such cases in this work. An input vector that only includes signals from the six-valued algebra is denoted by v_s.

The six valued algebra that we use is essentially the same as the one presented in [7]. In [7], an eight-valued algebra is also presented to include dynamic hazards, and differentiate them from clean transitions. In our work, there is no necessity to make the distinction and the values p0 and p1 also include dynamic hazards. The distinction is required only if justification of the values on the victim or aggressor forced a primary input to a dynamic transition. In such a case, because of the constraint that primary inputs can only have clean transitions, the particular interaction would be classified as invalid, whereas without the distinction it would be considered valid. We claim that this possibility can never occur. A forced dynamic transition will be required at an input of a gate only if the output is forced to be a dynamic transition. However, the conditions necessary for static or dynamic sensitization only specify that a transition has to occur on a victim or an aggressor. They do not specify whether the transition has to be clean or give a limit on the number of transitions.

It is important to show that the algebra we use is closed with respect to boolean operations. In addition to the six values in the algebra, it becomes necessary to use the X value during ATPG, where X represents an unknown value for a line. For proving closure, it suffices to show that the algebra is closed with respect to the AND, OR and NOT operators. Closure for the NOT operator is trivial. The table for the AND operator is given in Table 3, and the OR

TABLE 3: AND operation on signal values

AND	s0	s1	p0	p1	g0	g1	X
s0	s0	s0	s0	s0	s0	s0	s0
s1	s0	s1	p0	p1	g0	g1	X
p0	s0	p0	p0	g0	g0	p0	X
p1	s0	p1	g0	p1	g0	p1	X
g0	s0	g0	g0	g0	g0	g0	X
g1	s0	g1	p0	p1	g0	g1	X
X	s0	X	X	X	X	X	X

operator is dealt with in a similar manner.

We can now restate the conditions for static and dynamic sensitization in terms of the algebra of Table 1.

Theorem 4: A delay-increase interaction for a system is statically sensitizable iff

$$\exists\, v_s\text{: } V\text{=p0}, A_i\text{=p1}, i\text{=1,2..}n \qquad (11)$$

It can be seen that the condition given above is the same as (7) in the definition of a statically sensitizable interaction (definition 3). Since we proved via theorem 1 that (7) and (8) are equivalent, we need not find a vector to satisfy (8) separately.

Theorem 5: A delay-increase interaction for a system is dynamically sensitizable iff

$$\exists\, v_s\text{: } V\text{=p0}, A_i \in \{\text{p1,g0,g1}\}, i\text{=1,2..}n \qquad (12)$$

or

$$\exists\, v_s\text{: } V\text{=p1}, A_i \in \{\text{p0,g0,g1}\}, i\text{=1,2..}n \qquad (13)$$

Again, under the definition of our six-valued algebra, (12) is equivalent to (9), and (13) is equivalent to (10).

Equations (11), (12) and (13) together give the necessary and sufficient conditions for a coupling interaction to be valid. We can further combine (12) and (13) into one equation since they too are equivalent. Specifically, if a vector v_s from the six-valued algebra satisfies (12), then we will show that the vector v_s' satisfies (13), where v_s' is defined as follows: (i) For a primary input with value s0 or s1 in v_s, the value remains the same in v_s'. (ii) a primary input with p0 in v_s is replaced by p1 in v_s' and a primary input with p1 in v_s is replaced by p0 in v_s'.

Lemma 1: For any line L, $L(v_s')$ = p0(p1) iff $L(v_s)$=p1(p0).

Since the initial and final values in v_s and v_s' are interchanged for each primary input, the steady-state values for each line will also get interchanged under v_s and v_s'.

Theorem 6: For any line L, $L(v_s')$=s0(s1) iff $L(v_s)$=s0(s1).

Proof: We will start with the assumption that $L(v_s)$=s0(s1) and show that $L(v_s')$=s0(s1). A similar reasoning can be used to prove the converse. First, we observe that if $L(v_s)$=s0, then $L(v_s') \in \{\text{s0,g0}\}$. This is because the initial and final values of each line are just interchanged between v_s and v_s'. If L is a primary input, then it cannot take the value g0, and hence the theorem hods. For the case that L is not a primary input, we will

prove by contradiction. Suppose L is the output of gate G, and $L(v_s') = L(v_s)$. For the output of G to have a stable value (s0 or s1), at least one of it's inputs must be stable. Also, since $L(v_s') \neq L(v_s)$, at least one of those stable inputs of G must change values between v_s and v_s'. This can be rigorously proven by considering all possible inputs for the three basic gates: NOT, AND and OR. Thus there exists an input I of G such that $I(v_s) \neq I(v_s')$ *and* $I(v_s)$ is a stable value. This process of back tracing can continue till we reach a primary input, for which we know that a stable value in v_s remains unchanged in v_s'. Hence if $L(v_s)$=s0(s1), $L(v_s')$=s0(s1).

Theorem 7: If $A_i(v_s) \in \{g0,g1\}$, $A_i(v_s') \in \{g0,g1\}$.

If $A_i(v_s') \in \{s0,s1,p0,p1\}$, we would get a contradiction for $A_i(v_s)$ based on lemma 1 and theorem 6. But we know that the six-valued algebra is closed, hence all signals have to belong to {s0,s1,p0,p1,g0,g1}. Hence, $A_i(v_s') \in \{g0,g1\}$.

Theorem 8: If there exists a v_s that satisfies (12), then there exists a v_s' that satisfies (13).

From lemma 1, theorem 6 and theorem 7, it follows that v_s' satisfies (13). Theorem 8 is quite powerful because it reduces by half the number of cases to be analyzed for dynamic sensitization too.

4.2 Choice of ATPG Algorithm

The factors that guide our choice of an algorithm are different from those that determine the best approach for usual ATPG. The goal of ATPG is to find a test vector to detect a particular fault, or in other words find an input vector that justifies certain values on certain signal lines. For the problem of identifying false interactions, it is more useful to show that no vector exists that can justify the required values at the victim and the aggressor. The test vector itself is not of any practical use. Since we use a multi-valued algebra, an output value usually can be obtained from a variety of input combinations. Hence, exploring the search space by starting from the victim and aggressor lines, as in the D-algorithm[10], turns out to be inefficient. We used a modified version of the PODEM[5] algorithm for our approach, since the search space explored in PODEM is the primary inputs. Another reason for choosing PODEM was that the primary inputs have a constraint that they can assume only a subset of values from the six-valued algebra. By its very nature, PODEM avoids conflicts at primary inputs and hence is a suitable candidate for analyzing system interactions.

There is, however, a serious limitation of PODEM in the presence of multiple objectives. If one objective causes a local conflict with another objective, then PODEM would take a long time to detect it, since it will have to exhaust the search space at the primary inputs. We implemented an imply-and-check routine to detect such obvious conflicts since our primary goal is to detect invalid coupling interactions as quickly as possible. Given a set of objectives, our approach identifies an input values using a modified version of PODEM for multiple objectives. Our algorithm is summarized in Figure 4. The reorder function is used to determine the order of selecting primary input values based on the current objective as well as the number of inversions encountered while backtracing.

```
doesTestPatternExist(){
    set All lines to X
    If (implyAndCheck == FAILURE) return NO
    reset lines to X
    if (Podem()==FAILURE) return NO
    else return YES
}

Podem(){
    if all objectives satisfied, return SUCCESS;
    if conflict on any objective, return FAILURE
    if backtrack limit reached, return FAILURE
    currentObj = first objective with X value
    primary Input pi = backtrace(current Obj)
    piValueList = [s0 s1 p0 p1]
    reorder(piValueList, currentObj)
    for each value in piValueList{
        imply(pi, value);
        if (Podem()==SUCCESS) return SUCCESS;
    }
    imply(pi, X)
    return FAILURE;
}
```

FIGURE 4: Modified PODEM algorithm to identify valid coupling interactions

5. RESULTS

Since we did not have real circuits with coupling as well as functional information, we used the ISCAS benchmarks and selected random coupled systems from each of them. Up to five aggressors were allowed, since the delay impact of each aggressor is very small if the number of aggressors is large. The difference between the levels of any two lines in a coupled system was limited, to simulate real-circuit behavior by localizing the coupling. For each circuit, we generated 100 coupled systems and studied whether the delay-increase and delay-decrease interaction for each system was sensitizable. Since both types of interaction have to be studied independently, we had 200 sets of objectives. The number of statically sensitizable and dynamically sensitizable interactions were calculated separately. The total number of valid and invalid interactions ones are also reported. The results are tabulated in Table 4. The run times reported are on a Pentium III 500 MHz PC running Linux.

It can be observed from the table that the number of statically sensitizable interactions is typically less than the number of valid interactions. This implies that a portion of valid interactions are only dynamically sensitizable. Static sensitization alone, without incorporating glitches, will lead to optimistic results which is unacceptable in timing analysis. Our method of incorporating hazards and detecting valid interactions helps in identifying false interactions more accurately.

Since we account for all possible dynamic switchings, including both static and dynamic hazards, our approach is guaranteed to be conservative. This means that in any interaction that we claim as invalid, the victim and aggressors can never all switch together. The largest number of invalid interactions, 12 out of the total of 200, is observed for C1908. These false interactions could then be input to the static timing analyzer which could then ignore their contributions to coupling delay. The number of invalid interactions is observed to be relatively small for the other circuits. The results show that this number is largely design dependent and hence functional analysis is critical for eliminating conservatism.

6. CONCLUSIONS

We have presented a comprehensive analysis of false coupling interactions. We have classified them as statically and dynamically sensitizable depending on the transitions on aggressors and the victim. We have provided an ATPG approach with a multi-valued algebra to identify invalid coupling interactions. The results show the value of considering functional information to identify and eliminate pessimism in static timing analysis.

References

[1] R.Arunachalam, K.Rajagopal and L.T.Pileggi, "TACO: Timing Analysis With COupling," *Proc. Design Automation Conference*, 2000, pp. 266-269.

[2] P.Chen and K.Keutzer, "True Crosstalk noise analysis," *Proc. Intl. Conf. Computer-Aided Design*, 1999, pp. 111-116.

[3] H.C.Chen and D.H.C.Du, "Path sensitization in critical path problem," *IEEE Trans. Computer-Aided Design*, vol. 12, Feb. 1993, pp. 196-207.

[4] S.Devadas, K.Keutzer, S.Malik and A.Wang, "Certified timing verification and the transition delay of a logic circuit," *Proc. Design Automation Conference*, 1992, pp. 549-555.

[5] P.Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on Computers*, March 1991, pp. 211-222.

[6] S.Hassoun, "Critical path analysis using a dynamically bounded delay model," *Proc. Design Automation Conference*, 2000, pp. 260-265

[7] J.P.Hayes, "Digital Simulation with Multiple Logic Values," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, April 1986, pp. 274-282.

[8] D.A.Kirkpatrick and A.L.Sangiovanni-Vincentelli, "Digital Sensitivity: Predicting Signal Interaction using Functional Analysis," Proc. Intl. Conference on Computer-Aided Design**,**1996, pp. 536-541.

[9] R.Kundu and R.D.Blanton, "Identification of Crosstalk Switch Failures in Domino CMOS Circuits," Proc. Intl. Test Conference, 2000, pp.502-509.

[10] J.P.Roth, "Diagnosis of Automata failures: A calculus and a method," IBM Journal of Research and Development, Vol.10, No.4, pp.278-291, 1966.

[11] Y.Sasaki and G.De Micheli, "Crosstalk Delay Analysis using Relative Window Method," *Proc. 12th Annual Intl. ASIC Conference*, 1999, pp. 9-13.

TABLE 4: Distribution of coupling interactions in ISCAS85 benchmark circuits

Circuit	Statically sensitizable	Dynamically sensitizable	Reached backtrack limit	Valid interactions	Invalid interactions	CPU time (seconds)
C432	161	127	17	183	0	89.3
C499	152	152	48	152	0	16.4
C880	178	168	4	188	4	40.3
C1355	194	196	0	198	2	5.1
C1908	188	50	0	188	12	2.9
C2670	184	152	2	194	4	31.9
C3540	146	166	0	194	6	186
C5315	198	146	0	200	0	15.8
C6288	178	176	0	194	6	5.8
C7552	182	160	0	192	8	17.9