

Temporofunctional Crosstalk Noise Analysis

Donald Chai
Department of EECS
University of California
Berkeley, CA 94720
donald@eecs.berkeley.edu

Alex Kondratyev
Cadence Berkeley Lab
2001 Addison
Berkeley, CA 94704
kalex@cadence.com

Yajun Ran
Department of ECE
University of California
Santa Barbara, CA 93106
ranyj@ece.ucsb.edu

Kenneth H. Tseng
Cadence Design Systems
555 River Oaks Parkway
San Jose, CA 95134
kentseng@cadence.com

Yosinore Watanabe
Cadence Berkeley Lab
2001 Addison
Berkeley, CA 94704
watanabe@cadence.com

Malgorzata Marek-Sadowska
University of California
Santa Barbara, CA 93106
mms@ece.ucsb.edu

ABSTRACT

Noise affects circuit operation by increasing gate delays and causing latches to capture incorrect values. This paper proposes a method of characterizing correlation of signal transitions in multiple nets by considering both timing and functionality of the signals, and uses it in an analysis procedure to eliminate noise faults that cannot actually happen when such correlations are considered. It uses four-variable Boolean logic to characterize signal transitions in a time interval, and formulates Boolean satisfiability between aggressors and a victim under the min-max delay model for gates. The technique has been successfully applied to commercial ASIC designs and has eliminated up to 35% of delay noise faults reported by a state-of-the-art noise analysis tool.

Categories and Subject Descriptors

B.7.2 [Integrated Circuits]: Design aids

General Terms

Algorithms, Reliability, Verification

Keywords

crosstalk noise, SAT formula, timed Boolean logic

1. INTRODUCTION

With the advance of process technology noise immunity is turning into a metric of the same importance as power, area and timing because it directly influences all of them.

When neighboring nets switch, the coupling capacitors cause transfer of charge between them. Depending on the relative times and directions of switching the amount of crosstalk noise varies. The net that is affected by the noise is known as a *victim* net, while the neighboring nets which affect it are known as *aggressors*. Noise faults may manifest itself as *functional noise faults* in which victim nets take incorrect values, or *delay noise faults* in which correct transitions on victim nets are delayed or speed up. In most industrial tools, noise analysis is typically done after layout information is available, so that the set of potential ag-

gressors for a net is limited to its neighbors and coupling capacitances are known [1, 2, 8].

Timing correlations between transitions in aggressor and victim nets are usually captured by switching windows obtained by Static Timing Analysis (STA). A switching window is a timing interval in which a net can possibly make transitions. If switching windows of two aggressor nets do not overlap then this combination of aggressors cannot simultaneously contribute to a noise fault. Pruning by switching windows is extremely efficient [3, 9, 1, 2, 8] because STA is of linear complexity of the size of a circuit.

The simplicity of pruning heuristics comes at a cost. STA does not take into account the functional correlations between nets. Therefore the results of noise analysis based on the layout information and switching windows might be overly pessimistic. The main goal of this paper is to improve the accuracy of noise analysis by taking into account both temporal and functional correlations in a uniform way.

Prior work on using circuit functionality to reduce the pessimism in noise analysis has taken on different flavors. [10] observes that some signals may always transition in opposite directions, and uses such functional correlation to estimate noise delay faults in critical paths. [6] exploits compatible output don't care sets to prune transitions at aggressors to only those that attack victims at their care state. [5] uses logic implications between nets to specify feasible switching behaviors, and forms a constraint graph from which noise is estimated by finding a maximum weighted independent set of aggressors. The closest to ours is the work in [4], which to the best of our knowledge, is the first to introduce a Boolean satisfiability (SAT) formulation to the noise analysis problem. In [4] the analysis is reduced to the search for two input patterns which could be successively applied to a circuit to justify the feasibility of simultaneous transitions at a chosen subset of aggressors.

While good progress was achieved in understanding how to incorporate functional correlations into noise analysis, a general and practically viable solution to this problem is still lacking. This is because most of the prior work impose oversimplified assumptions either about the delay models (like fixed delays in [4]) or about the captured functional correlations [10, 5, 6]. Our approach uses the more general delay

model that uses intervals ($[min, max]$) and distinguishes rising/falling and pin-to-output delays. Also, our SAT formulation is more general than [4] so that rise and fall transitions are considered separately in a given timing interval.

The technique has been integrated with a commercial noise analysis tool (CeltIC) and applied to industrial ASICs as well as MCNC benchmarks. The experiments show that the procedure can be executed efficiently and eliminates up to 35% of delay noise faults in industrial examples.

2. NOISE ANALYSIS FLOW

The suggested noise analysis flow is shown in Figure 1. Given an extracted netlist, *CeltIC* first finds an aggressor set for every net. Although a net might be impacted by many of its neighbors it often has a small set of strong aggressors (usually less than 5), which are defined by the ability to inject noise exceeding a predefined threshold. Other (weak) aggressors are modeled by a single “virtual” aggressor that is computed using weighted lumped approximations.

In delay noise analysis the amount of noise injected from every aggressor and their alignment offset to the victim is estimated using reduced order modeling methods. Once the worst aggressor subset is determined, the total delay change is calculated using a single fast circuit simulation considering the non-linear combined effects of the selected aggressors and the victim transition.

We derive the noise contribution from each aggressor i ($noise_i$) by fitting the accurate total noise ($noise_{total}$) determined by circuit simulation to a linear function of capacitance. Then $noise_i = c_i \times n$, where c_i is the fraction of total coupling capacitance on the victim from aggressor i , and n is derived so that $\sum_i noise_i = noise_{total}$.

Crosstalk delay analysis and timing analysis are mutually dependent. On one hand, delay faults depend on the arrival times of signals and on their slew rates. On the other hand, switching windows must take into account possible slow downs or speed ups due to noise. This dependency is resolved by iteration between noise and timing engines. For every net with a set of aligned aggressors, the noise engine in *CeltIC* finds the worst-case noise (functional or delay) by a fast circuit simulation. The delay noise is then back annotated to a timing engine and switching windows of nets are updated. *

The place of the presented work in the overall noise analysis flow is shown by the shadowed block in Figure 1. Our tool is used as a post-processor that makes an additional pruning of the faults reported by *CeltIC* through taking into account temporal and functional correlations between victims and aggressors. Based on converged switching windows for every net the tool builds a timed SAT formula to check whether a worst case scenario for speedup or slowdown (in case of delay faults) or for glitch (in case of functional faults) is feasible for a given net. If the formula is unsatisfiable the reported noise fault is infeasible. In that case the tool reports the worst case among the remaining feasible scenarios and the corrected values for noise faults.

3. SAT FORMULATION

3.1 Preliminaries

To combine functional and temporal correlations, we first present how we integrate the specification of timing with functionality.

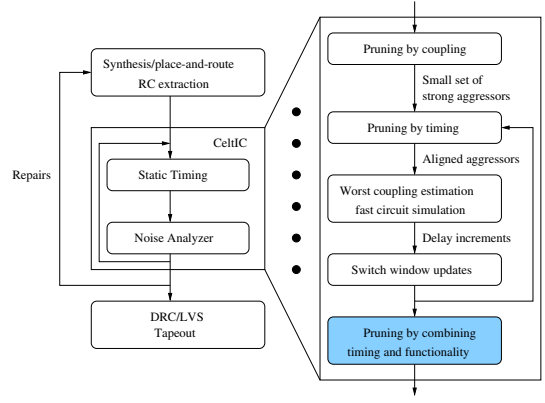


Figure 1: Noise analysis flow

We introduce a system of timed Boolean variables to capture switching activities passing through gates with varying delay. For a wire y in time interval $T = [t_1, t_2]$, we introduce four binary variables $y_T^0, y_T^1, y_T^r, y_T^f$ that indicate whether the wire is low, high, rising, or falling, respectively at some time in $[t_1, t_2]$. By using a padding of switching windows by a half-slew delay it is possible to abstract away transitions at the ends of the interval.

We define interval arithmetic in the conventional manner. Assuming $T_1 = [t_x, t_y]$ and $T_2 = [t_a, t_b]$:

$$\begin{aligned} T_1 + T_2 &= [t_x + t_a, t_y + t_b] \\ T_1 - T_2 &= [t_x - t_b, t_y - t_a] \\ T_1 \cup T_2 &= [\min(t_x, t_a), \max(t_y, t_b)] \\ T_1 \cap T_2 &= [\max(t_x, t_a), \min(t_y, t_b)] \end{aligned}$$

Furthermore, it can be shown that $-$ distributes over \cup .

Our approach refines noise estimates based on forward STA by incorporating functionality to prove that a set of nodes may never switch during some time interval. We do so by propagating timing information backward via required timing analysis and constructing a formula of timed Boolean variables which may be checked using a conventional SAT solver.

3.1.1 Gate Sensitization

Consider the case of a single AND gate $y = ab$. From library characterization, the pin to output fall (rise) delay for input a to output y is given as an interval D_a^f (D_a^r), and corresponding delay for input b is given as D_b^f (D_b^r).

Given these, by setting b to a non-controlling value '1', $a_T^r \iff y_{T+D_a^r}^r$.

In its own turn the boundary conditions for $b = 1$ could be derived by considering two different scenarios:

1) rising transition of b happens *not later* than the interval of interest (for $T = [t_1, t_2]$ and $D_b^r = [t_{min}^r, t_{max}^r]$) it means that b^r may happen not later than $(t_2 - t_{min}^r)$,

2) falling transition of b happens *not earlier* than the interval of interest (for $T = [t_1, t_2]$ and $D_b^f = [t_{min}^f, t_{max}^f]$) it means that b^f may happen not earlier than $(t_1 - t_{max}^f)$.

Bearing in mind that the interval $T' = [t_1 - t_{max}^f, t_2 - t_{min}^r]$ could be approximated as $T - (D_b^r \cup D_b^f)$ we can derive:

$$y_T^r = a_{T-D_a^r}^r b_{T-(D_b^r \cup D_b^f)}^1 + b_{T-D_b^r}^r a_{T-(D_a^r \cup D_a^f)}^1$$

The conditions for each of the variables on the right hand side are specified in a similar fashion, continuing through

the transitive fan-in until we reach the inputs. The full set of constraints for an AND gate is shown below.

$$\begin{aligned} y_T^1 &= a_{T-(D_a^r \cup D_a^f)}^1 b_{T-(D_b^r \cup D_b^f)}^1 \\ y_T^0 &= a_{T-(D_a^r \cup D_a^f)}^0 + b_{T-(D_b^r \cup D_b^f)}^0 \\ y_T^r &= a_{T-D_a^r}^r b_{T-(D_b^r \cup D_b^f)}^1 + b_{T-D_b^r}^r a_{T-(D_a^r \cup D_a^f)}^1 \\ y_T^f &= a_{T-D_a^f}^f b_{T-(D_b^r \cup D_b^f)}^1 + b_{T-D_b^f}^f a_{T-(D_a^r \cup D_a^f)}^1 \end{aligned}$$

In general, y_T^1 and y_T^0 can be generated by substituting timed variables in the original formula and its complement. y_T^r and y_T^f can be generated using the Boolean difference to check sensitization conditions as follows (timing subscripts omitted for clarity):

$$\begin{aligned} y^r &= \sum_i (x_i^r y_{x_i^0}^0 y_{x_i^1}^1 + x_i^f y_{x_i^1}^0 y_{x_i^0}^1) \\ y^f &= \sum_i (x_i^r y_{x_i^1}^0 y_{x_i^0}^1 + x_i^f y_{x_i^0}^0 y_{x_i^1}^1) \end{aligned}$$

3.2 Input Conditions

In pure functional analysis, a solution is found in terms of two input vectors, one applied at $t = -\infty$ and another applied at $t = 0$. The single switch assumption is reasonable for inputs into combinational logic in a synchronous design. When we incorporate timing, we assume that an initial vector is applied at $t = -\infty$ and allow the input transition at net x to occur at any time during some $sw_x = [e_x, l_x]$.

When required time windows are propagated backwards from the analyzed nets, multiple timing windows might be obtained at a primary input, introducing many timed Boolean variables (one for each window). For primary inputs, the following two conditions must hold for such variables:

- persistence – variables at adjacent points in time of the same net must have the same value unless there is a transition event between the two, and
- monotonicity – there is at most one transition event

These conditions may be enforced by examining all points (in time) of interest. Let P consist of all endpoints of back-propagated windows plus the two points $\{e_x, l_x\}$. P induces a partition of sw_x . Introducing variables x_{t_i} , where $t_i \in P$, $i = 1, k$ ($k = |P|$), we have the following relation:

$$t_1 = e_x, t_k = l_x, \quad x_{t_i} = switch_{i-1} \oplus x_{t_{i-1}}$$

where $switch_i$ denotes a transition between t_i and t_{i+1} . Monotonicity is enforced by the following:

$$\forall_{i \neq j} \overline{switch_i} + \overline{switch_j}$$

The behavior of a variable over an interval can be described by its endpoints. Suppose $T = [t_1, t_2]$, then $x_T^1 = x_{t_1} + x_{t_2}$, $x_T^0 = \overline{x_{t_1}} + \overline{x_{t_2}}$, $x_T^r = \overline{x_{t_1}} x_{t_2}$, $x_T^f = x_{t_1} \overline{x_{t_2}}$.

3.3 Temporal Pruning

In these analyses, we may derive required times outside the bounds of the arrival times. In the above example with the AND gate, it is possible that a never rises during time $T - D_a^r$. Then we may exclude a from consideration. In general, assuming required time interval $T = [t_1, t_2]$ and arrival time interval $sw = [e, l]$ and denoting by $x^{-\infty}$ and $x^{+\infty}$ the initial and final (settled) values of the net respectively the following pruning is possible:

	$t_2 < e$	$t_1 < e$	$t_2 > l$	$t_1 > l$
x_T^1	$x^{-\infty}$	$x_{T \cap sw}^1 + x^{-\infty}$	$x_{T \cap sw}^1 + x^{+\infty}$	$x^{+\infty}$
x_T^0	$x^{-\infty}$	$x_{T \cap sw}^0 + x^{-\infty}$	$x_{T \cap sw}^0 + x^{+\infty}$	$x^{+\infty}$
x_T^r	0	$x_{T \cap sw}^r$	$x_{T \cap sw}^r$	0
x_T^f	0	$x_{T \cap sw}^f$	$x_{T \cap sw}^f$	0

3.4 Path Merging

In arrival time analysis, adjacent or overlapping switching windows may be merged without loss of accuracy. However, in required time analysis, dependency information must be preserved to check functional correlation.

Whenever there are re-convergent paths, there may be an exponential number of paths between two nets, which in turn leads to an exponential number of switching windows and timed Boolean variables propagated. To deal with this, we approximate our analysis by selectively merging a pair of switching windows T_a and T_b to create a new window $T_{new} = T_a \cup T_b$. While by our interval semantics, $x_{T_{new}}^r$ equals $x_{T_a}^r + x_{T_b}^r$, we make the approximation that $x_{T_{new}}^r \implies x_{T_a}^r x_{T_b}^r$ so that only $x_{T_{new}}^r$ needs to be manipulated later. Similar simplifications are made for x^f , x^0 , and x^1 .

3.5 Single Net Noise Analysis

In single net noise analysis, we determine whether a group of aggressor nets may cause noise faults on a particular victim net v . It takes as input

- a set of feasible switching windows for the nets derived from arrival time analysis using noisy gate delays,
- a set of gate delays specified in terms of pin-to-output delay intervals, considering previously determined noise delay faults,
- a victim v and a set of aggressors Agg ,
- a slew of the victim v ,
- maximal slowdown $\delta^+(v)$ and speedup $\delta^-(v)$ due to noise when all aggressors are considered active, and
- a switching window of $v [e_v, l_v]$ when all aggressors are silent and a step Δ to discretize slowdown or speedup on v .

For functional noise, the task is to find if there is any feasible combination of signal transitions between the victim and the aggressors. We take the maximum overlapping of aggressors' switching windows to test the noise effect on v . For delay noise, we analyze both the maximal slowdown and the maximal speed up that are feasible. In the former, we test the noise effect on v in a switching window given by $[l_v - slew/2 + \delta, l_v + \delta]$. Here, δ is a possible slowdown and we iterate the analysis from $\delta = \delta^+(v)$ to $\delta = 0$ while decrementing it by the discretization step Δ at each iteration, until the slowdown δ is identified as feasible. Similarly, for the speed up analysis, we use $[e_v - slew/2 - \delta, e_v - \delta]$, where δ is decremented from $\delta^-(v)$ to 0. In either case, following the required time propagation, we define timed Boolean variables for corresponding transitions in each timing window given for each net, and build a SAT formula by encoding their relationship across gates using the gate functionalities as described in Section 3.1.1.

When we analyze delay noise, if δ is less than the maximum slowdown (or speedup), then it is necessary to examine which subsets of the aggressors could contribute to causing that amount of slowdown (or speedup) at v . This is because $\delta^+(v)$ is defined as the maximum slowdown observed when all the aggressors are active, and thus in order to result in a smaller slowdown, only a subset of the aggressors must contribute. Every subset of aggressors is characterised by the

amount of coupling noise it injects and hence by the amount of slowdown (or speedup). As the number of strong aggressors is small all possible subsets could be checked explicitly by solving the corresponding SAT formula. To speed up this check we use a compression technique that groups subsets of aggressors by the amount of coupling noise and check the feasibility of the group in whole via SAT solving.

4. EXPERIMENTAL RESULTS

The proposed technique was tested on several industry circuits. Their characteristics (size and number of clock domains) are shown in Table 1.

Circuit	Ind1	Ind2	Ind3	Ind4
#Inst	15k	35k	115k	234k
#Nets	17k	41k	140k	273k
#clocks	2	5	7	8

Table 1: Test circuit profile

We chose the first 400 nets that CeltIC had reported as the worst delay noise (slowdown), and applied the single net noise analysis for each net, as given in Section 3.5, to see how much pruning could be made. For each net, only the strong aggressors were considered, since virtual aggressors’ timing windows are set to infinite and their effect is irrelevant during the SAT solving. For these nets, about 60-80% of the aggressors are among the same clock domain, although we also considered the aggressors in different clock domains. We used zChaff [7] as the SAT solver, and set the threshold of interval merging in the backward timing propagation to 5. For functional noise, the pruning was marginal (from 5% to 10% for the considered industrial circuits), while the more significant pruning was observed for delay noise.

Table 2 shows the amount of coupling noise pruning which was identified by SAT solver. Columns $\delta^+(v)$, $0.5\delta^+(v)$ and “no noise” shows the percentage of nets that were identified to have maximal feasible slowdown not more than $\delta^+(v)$, $0.5\delta^+(v)$ and 0 respectively. According to column $\delta^+(v)$ from 25% to 35% of the nets cannot have the worst slowdown $\delta^+(v)$ (as reported by CeltIC) when timing and functional correlation are considered. Column $0.5\delta^+(v)$ shows the ratio of nets that cannot experience $0.5\delta^+(v)$ slowdown, while column “no noise” shows the ratio of nets that cannot experience noise at all even though their adjacency and timing windows analysis does predict a potential slowdown $\delta^+(v)$. Column “CPU” shows the average CPU runtime required for a single iteration (one δ value) in the slowdown analysis per net, when running on 750 MHz Sparc v9 machine. From these numbers follows that analysis of 400 most noisy nets in each example is in the order of minutes and scales well with the size of example.

Due to the limited access to industry circuits, we also experimented with MCNC benchmark circuits. We synthesized those circuits using a $.18\mu\text{m}$ process library using commercially used delay models and timing analysis. Instead of applying detailed physical layout, we randomly chose aggressor nets for each victim. The timing windows for the primary inputs were set to $[0, 50\text{ps}]$. Table 3 shows the

Circuit	Pruning (%)			CPU (s)
	$\delta^+(v)$	$0.5\delta^+(v)$	No noise	
Ind1	35.0	17.1	8.4	0.52
Ind2	29.0	12.2	7.0	2.03
Ind3	26.0	13.9	9.3	3.85
Ind4	25.0	17.4	12.7	1.68

Table 2: Noise pruning for industry circuits

Circuit	#nets	$\delta^+(v)$	$0.5\delta^+(v)$	no noise
C880	87	72.4	87.3	89.6
C1355	106	60.4	73.6	83.0
C1908	134	83.6	92.6	94.0
C2670	93	83.9	88.2	90.4
C3540	394	80.5	88.2	94.3
C5315	323	85.1	93.5	94.4
C7552	459	92.6	96.9	98.2
alu4	199	68.8	88.4	95.9
apex6	125	72.0	81.6	88.8
des	733	90.6	95.0	96.0
dalu	197	81.2	87.8	91.8
frg2	116	78.4	83.5	87.0
k2	315	76.5	92.4	96.5
t481	176	77.8	93.7	96.6
vda	166	80.1	92.2	95.8
pair	321	76.9	83.7	87.7
i10	493	83.4	90.3	92.5
AVG	261	79.1	88.8	92.5

Table 3: Slowdown analysis for MCNC benchmarks

results for the slowdown analysis. We observe a similar tendency with the industrial examples.

5. CONCLUSION

This paper presents a refined method for crosstalk noise analysis that takes into account both temporal and functional correlations between signals. It relies on SAT-based formulation for checking the feasibility of particular aggressor-victim switching patterns. The method is used in a commercial noise analysis tool to filter infeasible noise faults. The experiments show that up to 35% of delay noise faults can be eliminated with this technique.

6. ACKNOWLEDGMENTS

We would like to thank Luciano Lavagno (Cadence Berkeley Labs) and Vinod Kariat (Cadence Design Systems) on many stimulating discussions and useful insights.

7. REFERENCES

- [1] User guide. In *Celtic User Manual, Cadence Design Systems, Inc*, 2002.
- [2] User guide. In *Prime Time User Manual, Synopsys, Inc*, 2002.
- [3] R. Arunachalam, K. Rajagopal, and L. Pileggi. Taco: timing analysis with coupling. In *DAC*, 2000.
- [4] P. Chen and K. Keutzer. Towards true crosstalk noise analysis. In *ICCAD*, 1999.
- [5] A. Glebov, S. Garrilov, D. Blaauw, S. Sirichotiyakul, C. Oh, and V. Zolotov. False-noise analysis using logic implications. In *ICCAD*, 2001.
- [6] D. Kirkpatrick and A. Sangiovanni-Vincentelli. Digital sensitivity: predicting signal interaction using functional analysis. In *ICCAD*, 1996.
- [7] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *DAC*. ACM Press, 2001.
- [8] R. Levy, D. Blaauw, G. Braca, A. Dasgupta, A. Grinshpon, C. Oh, B. Orshav, S. Sirichotiyakul, V. Zolotov, and T. Xiao. Clarinet: A noise analysis tool for deep submicron design. In *DAC*, 2000.
- [9] T. Xiao and M. Marek-Sadowska. Worst delay estimation in crosstalk aware static timing analysis. In *ICCD*, 2000.
- [10] T. Xiao and M. Marek-Sadowska. Functional correlation analysis in crosstalk induced critical paths identification. In *DAC*, 2001.