Eliminating False Positives in Crosstalk Noise Analysis

Yajun Ran ranyj@ece.ucsb.edu
Alex Kondratyev† Yosinori Watanabe† {kalex,watanabe}@cadence.com
Malgorzata Marek-Sadowska mms@ece.ucsb.edu

Department of ECE, University of California, Santa Barbara, CA 93106
†Cadence Berkeley Labs, 2001 Addison Street, Berkeley, CA 94704

Abstract

Noise affects circuit operation by increasing gate delays and causing latches to capture incorrect values. Noise analysis techniques can detect some of such noise faults, but accurate analysis requires a careful examination of timing and functional properties of the circuit. This paper proposes a method to check the "true" noise impact on path delay. 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 proposed technique is scalable as it keeps the size of Boolean formulation linear to the size of the modeled circuit. By applying it to a set of large circuits, it has eliminated up to 50% of noise delay faults reported by conventional noise analysis method.

1. Introduction

As VLSI process technology advances, the noise problem has become one of major issues for chip designers. A crosstalk noise may cause functional failure if its induced glitch noise is incorrectly latched, or delay faults if its induced delay noise incurs timing violations.

One net introduces a crosstalk noise to another when the two nets are physically adjacent and transition at the same time with corresponding directions (same direction for speedup and opposite for slowdown). We call these conditions physical, temporal and functional correlation, respectively. This paper does not consider improvements in the accuracy of noise analysis stemming from better physical modeling. Physical modeling of crosstalk by itself is a topic of numerous research and industry works, see e.g. [1, 2, 9]. Our approach provides better accuracy of noise analysis by considering timing and functional correlations in a unified framework. In that way it is orthogonal to the works dealing with accurate physical modeling and could benefit from any progress reached there.

The temporal correlation could be considered to a certain extent using the static timing analysis. The analysis aims to compute the earliest and latest points in time for signal transitions to take place, and an aggressor whose switching window does not overlap with that of the victim may be excluded from consideration. Due to the mutual dependency of noise and timing window, iterations between switching window and noise analysis are performed [1, 2, 4, 9, 10]. But without considering the functional correlations, i.e. which directions of transition will take place at given nets, the iterative approaches tend to yield either inaccurate or exceedingly conservative results for practical designs [5].

In order to improve the accuracy of the noise analysis, both the temporal and functional analysis have to be accounted for. Approaches have been proposed to address this problem based on simplified assumptions on delay models (e.g. constant delays) or functional correlation (e.g. only in terms of settled values) [6, 7]. In [5], we have introduced an approach that is the most general, to the best of our knowledge, both in modeling power (delay of nets are characterized by $[min, max]$ intervals that distinguish between rise and fall transitions) and in the types of functional correlations to capture. The approach is based on a satisfiability (SAT) formulation using a four-valued timed Boolean logic, and [5] has demonstrated that for single nets up to one third of the aggressors that were originally reported by a state-of-the-art commercial tool did not introduce noise to the victim net.

This observation made us ask ourselves how much improvement in accuracy could be obtained when our analysis is carried out through an entire path or a fanin cone rather than single nets since we more care about the critical path timing. This paper addresses this problem, and presents a formulation based on the timed Boolean logic to compute bounds on the maximum slowdown and speedup in delay for a given path or a fanin cone due to crosstalk noise. Our approach is in contrast to the state-of-the-art procedures, where all noises on each net are simply propagated to the path end. Because of the temporal and functional correlation among the fanin cone of a path or an endpoint, the

1530-1591/04 $20.00 (c) 2004 IEEE

probability that worst-case noises can be all summed up is very small in general. By constructing a timed-SAT formula on the fanin cone of a path/endpoint based on the required transition analysis, we take all the correlations between the stages among a fanin cone into account, which results in less conservative analysis.

The approach has been implemented and tested on a set of industrial examples. The experiments showed that the approach is 1) efficient (discarding on average about 50% of delay faults reported by commercial tools as false ones) and 2) scalable (handling designs with the number of instances being hundreds of thousands).

Note, that the impact of crosstalk noise to critical timing is much stronger than in the case of false paths. We have observed that the delay of critical paths might increase up to 18% due to crosstalk. Our approach has been capable to recover about 50% of this delay increase when temporal and functional correlation were considered.

2. Functionality and timing

In a conventional flow, noise analysis first finds an aggressor set for every net and then iterates between noise calculation and timing analysis to achieve converged noisy switching windows for each net [4].

Computation of noisy switching windows provides a worst-case estimation of a noise impact. This estimation is conservative because (1) it assumes the worst possible alignment between victim and aggressors, (2) it ignores any functional correlations between transitions in a circuit and (3) it merely adds up the worst noise from all preceding stages of a net.

Our tool is used as a post-processor that makes an additional pruning of the faults reported by conventional analysis through taking into account issues (1-3). The reasoning about functional and temporal correlations is done by projecting the behavior of physical signals into the domain of Boolean timed variables.

2.1. Logic of intervals

Modeling a behavior of a circuit node at particular moment of time through multi-value logic is a well-known technique, see e.g. [3].

However the fixed delay model is overly optimistic for deep sub-micron designs. Delays may significantly vary due to process variation, IR drop and crosstalk noise. Because of delay variability it is not possible to tell exactly which behavior pattern (keeping steady values or switching) a signal might have at a certain moment of time. This naturally suggests to specify signal behavior by intervals rather than at discrete moments of time. In the sequel, we use the term interval and window interchangeably to refer to a continuous time region bounded by two numbers t_1 and t_2 with $t_1 \leq t_2$.

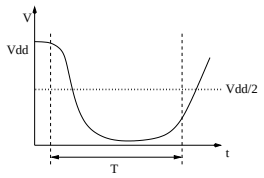


Figure 1. Behavior of a signal at interval T

By extending traditional 4-value logic to timing interval, we introduce four binary variables $y^0_T, y^1_T, y^r_T, y^f_T$ for a wire y in time interval $T = [t_1, t_2]$ to indicate whether the wire is low, high, rising, or falling, respectively at **some** time in $[t_1, t_2]$. The boundary between logical values '0' and '1' is defined by $V_{dd}/2$ voltage level. y^r_T (y^f_T) is true if there exists a pair $t_1, t_2 \in T : \ t_1 < t_2$ such that $y^0_{t_1}$ and $y^1_{t_2}$ ($y^1_{t_1}$ and $y^0_{t_2}$) are true. In Figure 1, the behavior of a signal y is described by the following set of variables: $y^0_T = 1, y^1_T = 1, y^f_T = 1$ and $y^r_T = 0$. In general the nature of interval-type of specification implies that $y^r_T \Rightarrow y^0_T y^1_T$, and $y^r_T y^f_T$ is not necessarily 0.

Logic of intervals assumes that in performing a required time analysis one needs to propagate backward both signal values and corresponding timing intervals. For the latter we define interval arithmetic in the conventional manner:

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

To illustrate propagation of intervals together with signal values let us consider 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^f_a (D^r_a), and corresponding delay for input b is given as D^f_b (D^r_b). Given these, by setting input b of AND gate to a non-controlling value '1', $a^r_T \iff y^r_{T+D^r_a}$.

Here is the full set of conditions for a 2-input AND gate (they can be derived for arbitrary Boolean function as well [5]).

$$\begin{aligned} y^1_T &= a^1_{T-(D^r_a \cup D^f_a)} b^1_{T-(D^r_b \cup D^f_b)} \\ y^0_T &= a^0_{T-(D^r_a \cup D^f_a)} + b^0_{T-(D^r_b \cup D^f_b)} \\ y^r_T &= a^r_{T-D^r_a} b^1_{T-(D^r_b \cup D^f_b)} + b^r_{T-D^r_b} a^1_{T-(D^r_a \cup D^f_a)} \\ y^f_T &= a^f_{T-D^f_a} b^1_{T-(D^r_b \cup D^f_b)} + b^f_{T-D^f_b} a^1_{T-(D^r_a \cup D^f_a)} \end{aligned} \quad (1)$$

Backward propagation of values and intervals through a combinational logic aims at representing a timing behav-

ior of a circuit through a single formula. The process terminates either at primary inputs (PIs) or at register outputs (PPIs). These points are characterized by arrival windows with at most one input transition per window. The single switch assumption is reasonable for primary inputs into a combinational logic cloud in a synchronous design methodology.

Once all transition and steady value intervals are propagated to inputs they could be pruned by taking an intersection with input arrival windows because a circuit is insensitive to input changes outside arrival windows.

2.2. Modeling noise in logic

The purpose of the worst case noise analysis is to determine the maximum slowdown/speedup that might occur at net transitions because of crosstalk. After analysis is done, switching windows $sw(v) = [e_v, l_v]$ for every net v are padded by the amount of slowdown (denoted by $\delta^+(v)$) and speedup (denoted by $\delta^-(v)$).

When constructing a timed boolean formula for a circuit behavior using logic of intervals and backward window propagation, there are two ways to take noise delays into account.

1. *Worst case noise delays*

   Backward propagation is performed from noisy switching windows. If the nominal pin a to output v fall delay is given as an interval $D_a^f = [d_{min}^f, d_{max}^f]$, then for the worst case noise this interval is modified as $Dn_a^f = D_a^f + \Sigma$, where $\Sigma = [-\delta^-(v), \delta^+(v)]$. Dn_a^f defines noisy pin-to-output delays which are used to propagate windows backward. A single step in analyzing noise slowdown is illustrated in Figure 2(a) by backward window propagation from output v to input a with solid arcs corresponding to delays with noise.

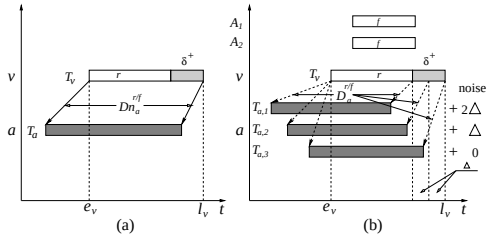


Figure 2. Propagating worst (a) and refined (b) slowdown noise delays

2. *Refined noise delays*

   For a signal at net v to arrive at time interval $T_v = [e_v, l_v]$, different arrival time intervals T_a at a fanin a of v are required depending on how much slowdown noise is obtained from v. It ranges from the one when no noise occur ($T_a = T_v - D_a^{r/f}$) to the one with the worst case noise ($T_a = T_v - \delta^+(v) - D_a^{r/f}$). The refined consideration of noise delays takes into account functional conditions under which a certain amount of noise is feasible. This is done through discretizing $\delta^+(v)$ with a pre-defined step Δ and deriving Boolean conditions for the feasibility of every interval.

The idea of the refined noise propagation is illustrated by Figure 2(b). By choosing a discretization step $\Delta = \delta^+/2$ three intervals for a victim v are considered: $T_1 = [e_v - 2\Delta, l_v - 2\Delta]$, $T_2 = [e_v - \Delta, l_v - \Delta]$ and $T_3 = [e_v, l_v]$. These windows are propagated backward to input a using **nominal** (non-noisy) range of delays $D_a^{r/f}$ (shown by dashed timing arcs) giving rise to propagated windows $T_{a,1}$, $T_{a,2}$ and $T_{a,3}$.

Suppose that slowdown δ^+ is induced by falling transitions at aggressors A_1 and A_2 which have an equal coupling and zero offset alignment with v. Then the Boolean conditions for rising transition at v happening in intervals T_1, T_2 and T_3 are respectively $B_{2\Delta} = A_{1,T_1}^f * A_{2,T_1}^f$ (both aggressors are required to switch to provide a $\delta^+ = 2\Delta$ slowdown), $B_\Delta = A_{1,T_2}^f + A_{2,T_2}^f$ (at least one aggressor is required to switch to provide a Δ slowdown) and $B_0 = 1$ (with no noise no aggressor is required to switch).

In general, for a given slowdown $\delta^+(v)$ of a rising transition at net v, discretization step Δ and an interval $T_v = [e_v, l_v]$ to be propagated one could describe the timed behavior of v as:

$$v_{T_v}^r = \sum_{j=0,k:\delta_j=\delta^+(v)-j*\Delta} v_{T_v-\delta_j}^r * B_{\delta_j} \qquad (2)$$

Formula (2) explicitly distinguishes the delay component of v with silent aggressors ($v_{T_v}^r$) and different levels of induced noise: starting from the maximal slowdown $\delta^+(v)$ when all aggressors impact v ($B_{\delta^+(v)} = \bigwedge Aggressor(v)_{T_v}^f$) to no aggressor impacting v ($B_0 = 1$). The derivation of B_{δ_j} can be found in [5].

For slowdown analysis only the noise occurring at the interval $[l_v - \delta^+(v), l_v]$ impacts the latest arrival time l_v. Due to that only the timing interval $[l_v - \delta^+(v), l_v]$ is checked for aggressor alignment and noise calculation.

With refined noise delay propagation, we explicitly check how much noise could occur at specified intervals for every net. Therefore the correlations between different stages along a path/cone are naturally taken into account to obtain "true" number for noise accumulation at the path/cone end.

3. Delay analysis

3.1. Cone and path delays in the presence of noise

Sections 2.1-2.2 give a way to specify a behavior of a combinational logic by timed Boolean formula. This formula takes into account functional and temporal conditions for all noise delays (see Section 2.2). The conditions are feasible if there exists an input pattern that makes the formula satisfiable. This could be efficiently determined using SAT-based technique.

For a given path (or cone of logic) L finding its "true" delay reduces to derivation of a SAT formula for L and every interval for slowdown or speedup at L, and then finding the maximum slowdown or speedup for which the formula is satisfiable.

This analysis takes as an input a set of noisy switching windows sw^n for all nets together with their noisy pin-to-output delays (these are available as a result of conventional worst-case noise analysis).

In addition for every net g_i of L it requires:

- the set of strong aggressors $Aggr$,
- worst case slowdown $\delta^+(g_i)$ (or speedup $\delta^-(g_i)$),
- a nominal window of g_i $[e_{g_i}, l_{g_i}]$ and a step Δ to discretize slowdown or speedup on g_i.

The procedure starts from a terminal net g_n of L and recursively constructs SAT formula using expression (2) over variables $g^p_{n,T}$, where $T = [l_{g_n}, l^n_{g_n}]$ representing the interval due to the worst-case noise accumulated at net g_n (between the nominal latest arrival time l_{g_n} with no slowdown noise considered and noisy latest arrival time $l^n_{g_n}$ with worst-case slowdown noise considered) and $p = \{r, f\}$ depends on the polarity of g_n. Note, that in unfolding SAT formula backward from g_n, for efficiency the refined noise delay propagation (see Section 2.2) is applied only to nets from L, while for "side" inputs the worst case noise delay propagation is considered (Algorithm 1). A binary search is used on T to find the latest satisfiable arrival time delay at g_n.

3.2. Critical timing with "true" noise

For practical purposes the main objects in noise analysis are the delay faults that produce timing violations. These violations (hold or setup) are revealed through the arrival time analysis of endpoints (PO/PPOs). The above cone delay analysis is naturally adopted here for its simultaneous consideration of all the paths ending at a specified endpoint. The maximum accumulated slowdown or speedup at an endpoint, which may affect critical timing, can be obtained. The delay fault of the path can be safely ignored unless it makes the path critical and produces timing violation.

Algorithm 2 gives the analysis flow. We start from a sorted endpoint list Θ_{ep} based on their slacks. We then check each endpoint one by one until the critical timing cannot be improved.

Algorithm 1 Path (cone) delay noise analysis

```
prev_lo := l_{g_n};
lo := (l_{g_n} + l^n_{g_n})/2;
hi := l^n_{g_n};
while (hi − lo) > T_th do
  // when building CNF, use refined noise analysis for
  // nets in L and use worst case noise for the rest of nets
  build CNF for g^p_{n,[lo,hi]};
  sol := solve();
  if sol = SAT then
    prev_lo := lo;
    lo := (hi + lo)/2;
  else
    hi := lo;
    lo := prev_lo;
  end if
end while
// hi is the satisfiable path(endpoint) delay
return hi;
```

Algorithm 2 Critical timing noise analysis

```
// sort endpoints by their slacks from worst-case analysis
Θ_ep := sort(endpoints);
Slack_{sat,max} := −∞;
repeat
  Slack_{sat,ep} := cone_analysis (ep); // endpoint analysis
  Slack_{sat,max} := max(Slack_{sat,ep}, Slack_{sat,max})
  ep := next_ep(Θ_ep);
until Slack_{sat,max} < Slack_{worst,ep}
```

3.3. Complexity issues

The size of the circuit that needs to be processed for path/cone L is defined by the combinational logic in the transitive fanin of all nets from L together with their aggressors. Moving backward, propagated windows are multiplied because of discretizing the slowdown and due to re-convergent paths. This results in exponential number of switching windows and timed Boolean variables propagated. To keep the size of the SAT formula under control two techniques are used:

- *Temporal pruning.*
  If a backward propagated window for required times is outside the bounds of switching windows for arrival times then the propagated window is dropped.
- *Interval merging.*
  This suggest to selectively merge a pair of disjoint switching windows T_a and T_b to create a new window $T_{new} = T_a \cup T_b$. We assume that a pre-defined threshold on the maximal number of windows at the output

of a single net is imposed. Interval merging is applied to keep the number of windows not exceeding a threshold.

Interval merging introduces an over-approximation in SAT analysis. However our experimental results have shown that increasing the threshold beyond 10 has a negligible impact on improving the accuracy of SAT analysis (see Section 4).

Thanks to interval merging the overall complexity of SAT formula is **linear** to the size of combinational cloud in the transitive fanin of L because the number of timed boolean variables for a net is kept below the pre-defined threshold.

4. Experiments

Effectiveness and scalability. Several industry circuits (see Table 1) were checked using *CeltIC* commercial analyzer [1] with further post-processing by our tool. The experiments were performed on an Intel P4 2.0GHz machine with 1G memory running Linux. We used zChaff [8] as the SAT solver and the time limit for a single SAT solving was set to 5 minutes.

Circuit	Tech (μm)	#Nodes	#Nets	f_{max}(MHz)
ind1	0.25	15k	17k	400
ind2	0.18	122k	139k	250
ind3	0.18	234k	273k	160
ind4	0.13	5.9k	6.5k	100
ind5	0.13	35k	41k	125
ind6	0.13	78k	86k	200
ind7	0.13	114k	116k	300
ind8	0.13	115k	140k	125
ind9	0.13	203k	220k	312

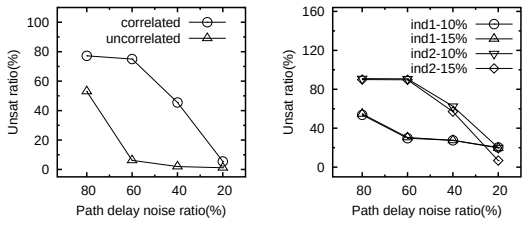
Table 1. Industry circuit profile

For each example from Table 1 the most noisy PPO/POs were chosen (400 in total) and SAT-based analysis was applied to their paths/cones. The noisy windows was discretized into 5 intervals and the threshold for merging propagated windows was set to 5 (meaning that every net had at most 5 propagated windows). Table 2 shows the results, where r_n is the average ratio of noise delay vs propagation delay, and CPU is the average runtime for a single path/cone analysis. Columns "$>$80%", "$>$60%", "$>$40%" and "$>$20%" give the percentage of paths/cones that were proved unsatisfiable for having noise delay larger than 80%, 60%, 40% and 20% of the worst-case determined through *CeltIC*. From table 2 follows that up to 90% (more than 50% on average) of paths/cones cannot experience the worst-case delay noise. In general cone analysis gives more pruning but consumes more CPU time. The run times for SAT analysis are acceptable and show the scalability of the suggested technique.

Circuit	r_n (%)	CPU (s) (path/cone)	unsatisfiable cases (%) (path/cone)			
			$> 80\%$	$> 60\%$	$> 40\%$	$> 20\%$
ind1	13.1	2.2/13.4	48.3/53.8	11.5/29.8	5.8/27.5	1.5/20.3
ind2	18.2	10.3/13.4	77.3/91.0	75.0/90.8	45.5/62.5	5.5/20.0
ind3	8.7	35.2/41.4	30.3/34.0	6.0/11.5	2.8/6.5	0.8/4.0
ind4	18.1	8.5/12.8	46.0/43.8	27.0/27.3	6.5/24.5	5.0/6.8
ind5	8.8	4.6/8.7	51.8/68.3	21.8/41.5	10.3/25.3	6.3/17.0
ind6	10.7	3.5/4.7	30.3/40.0	11.3/21.8	7.5/17.5	3.5/11.8
ind7	4.8	5.8/5.4	70.3/73.8	61.8/65.5	56.5/61.3	50.5/55.5
ind8	9.6	36.5/67.5	1.8/5.8	0/1.0	0/0.3	0/0
ind9	15.2	7.6/9.6	57.8/58.8	33.5/37.0	19.5/24.5	4.8/6.5
AVG	11.9	22.4/34.2	46.0/52.1	27.5/36.2	17.2/27.8	8.7/15.8

Table 2. Path(cone) delay noise analysis

Accuracy. Figure 3(a) shows the difference in the amount of pruned paths for correlated and uncorrelated analysis for circuit ind2. In uncorrelated path analysis, for every noisy net of a path a separate SAT formula was solved to determine the maximal satisfiable delay noise of a stage. In that way the amount of delay noise at every stage was pruned, however no correlation between stages was considered. Correlated path analysis was performed according to Algorithm 1 from Section 3. Figure 3(a) clearly shows the advantage of the proposed approach to the previously known technique from [5] and all other circuits show the similar behavior.



(a) correlated vs. uncorrelated analysis (ind2)

(b) delay variability

Figure 3. Accuracy factors: correlation (a) and variability (b)

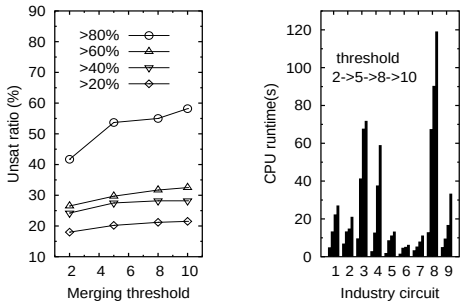
Figure 3(b) shows the impact of increasing the variability of net delays from 10% to 15%. Though in most of industrial examples the difference in pruning false faults was less than 5%, for circuit ind2 variability resulted in 14% difference in one region. This tells about an importance of variability modeling for modern technology.

To reveal the inaccuracy coming from interval merging, Figure 4(a) compares the difference in the number of unsatisfiable end-points under threshold values 2/5/8/10 for circuit ind1. The curves become flat after the threshold values larger than 5. Experiments showed that the difference in number of unsatisfiable cases between threshold value 5 and 10 (going to threshold values larger than 10 has a negligible impact on improving SAT results) does not exceed 10% for the considered circuits. The CPU runtime increases

Circuit	# Nodes	CPU (s)	T_{nom} (ns)	T_{noise} (%)	T_{sat} (%)	noise red (%)
alu4	2973	36.3	4.1	17.4	4.3	75.0
apex2	3915	4.1	6.8	7.6	1.4	81.4
apex4	2787	45.0	4.2	7.8	2.9	62.5
bigkey	5416	1.4	3.1	17.1	0.6	96.3
clma	17204	0.6	10.8	0.3	0.0	100.0
des	5596	7.4	4.1	3.3	3.3	0.0
diffeq	3621	46.7	5.8	4.3	0.6	86.8
dsip	4226	7.7	2.7	18.8	2.4	87.5
elliptic	9312	66.0	10.8	4.0	3.0	25.6
ex1010	9379	86.7	6.3	12.4	4.6	62.5
ex5p	2537	12.8	4.2	7.6	7.6	0.0
frisc	8869	459.7	12.0	2.3	0.0	100.0
misex3	2978	9.2	3.7	5.7	4.8	16.4
pdc	9857	107.7	9.0	8.6	4.3	50.0
s298	3543	45.4	7.0	2.6	2.6	0.0
s38417	14495	29.0	6.0	4.8	4.2	13.2
s38584.1	13654	33.5	5.1	7.3	5.5	24.9
seq	3669	0.8	6.0	9.5	7.1	25.1
spla	7785	4.2	6.9	5.5	2.1	62.1
tseng	3191	25.4	6.0	3.5	1.3	62.2
AVG		51.5		7.5	3.1	51.6

Table 3. Satisfiable critical timing

quickly with larger threshold values due to the fast growth of the size of SAT formula (Figure 4(b)).



(a) ind1 (b) Runtime

Figure 4. Effects of merging threshold on analysis

Impact on critical timing. Due to the limited access to timing information in industry circuits, we performed the experiments with MCNC benchmarks. The experimental flow started from synthesizing MCNC benchmarks and mapping them into a pre-characterized 0.13μm library. After placement and routing, we made a layout for each circuit and extracted aggressors and victims. The conventional crosstalk aware timing analysis then was performed to get the circuit timing and worst-case noise on each net. Algorithm 2 was applied to those circuits to check how much critical timing we can gain by eliminating false noise. Table 3 shows the results. Column "CPU" gives runtime for the entire analysis. T_{nom} is the nominal-case critical timing, T_{noise} is the noise increase with the conventional noise analysis and T_{sat} is the satisfiable critical timing. Column "noise red" gives the percentage of noise reduction from conventional worst-case analysis method. Two important observations might be made:

1) Crosstalk noise has a significant impact of critical timing (might increase it up to 18%).

2) The suggested approach recover on average 50% of the slowdown reported by noise analysis tools.

Since in most cases only a few end-points need to be analyzed, the entire analysis on average only takes tens of seconds.

5. Conclusion

This paper presented a method to check the "true" noise impact on path/cone delay. It uses four-variable Boolean logic, with which rise and fall transitions can be distinguished in a given timing interval. This has been used to eliminate noise faults that cannot take place when both timing and functional correlations are considered. The experiments show that on the average more than 50% of critical delay faults can be eliminated with this technique.

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] M. Abramovici, M. Breuer, and A. Friedman. *Digital Systems Testing and Testable Design*. IEEE Press, 1994.

[4] R. Arunachalam, K. Rajagopal, and L. Pileggi. TACO: timing analysis with coupling. In *Proceedings of DAC*, pages 266–269, 2000.

[5] D. Chai, A. Kondratyev, Y. Ran, K. H. Tseng, Y. Watanabe, and M. Marek-Sadowska. Temporofunctional crosstalk noise analysis. In *Proceedings of DAC*, 2003.

[6] P. Chen and K. Keutzer. Towards true crosstalk noise analysis. In *Proceedings of ICCAD*, pages 132–137, 1999.

[7] A. Glebov, S. Garrilov, D. Blaauw, S. Sirichotiyakul, C. Oh, and V. Zolotov. False-noise analysis using logic implications. In *Proceedings of ICCAD*, pages 515–521, 2001.

[8] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of DAC*, pages 530–535, 2001.

[9] 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 *Proceedings of DAC*, pages 233–238, 2000.

[10] T. Xiao and M. Marek-Sadowska. Worst delay estimation in crosstalk aware static timing analysis. In *Proceedings of ICCD*, pages 115–120, 2000.