False Paths Elimination in Statistical Static Timing Analysis

Masaki Uehata[1] , Masakazu Tanaka[2] , Masahiro Fukui[2] , and Shuji Tsukiyama[1]
[1]Dept. of EECE, Chuo University
Tokyo 112-8551, Japan
{masaki@tsuki., tsuki@}elect.chuo-u.ac.jp
[2]Matsushita Electric Industrial Co., Ltd.
Moriguchi 570-8501, Japan
{tanack, fukui}@mrg.csdd.mei.co.jp

Abstract: In this paper, we propose a technique to eliminate the effect of false paths in the calculation of the distribution of the maximum delay of a given CMOS combinatorial circuit, when distributions of interconnect delays and gate switching delays of the circuit are given. The technique can be incorporated into our algorithm for the statistical static timing analysis, which can take correlations of the delays into account.

1. Introduction

Since variability of design parameters increases in deep sub-micron era, the statistical static timing analysis[1], [2], [3], [4], [5], which analyzes the distribution of the maximum delay (critical delay) of a given combinatorial circuit, becomes important to design high speed and low power VLSIs. Because, designers often set excessive margins derived from the worst-case analysis in order to avoid the effect of delay time uncertainty, and such excessive margins bring over-design of circuits. If designers can estimate the distribution of critical path delay exactly, over-design of circuits may be eliminated so that high density and high performance VLSIs can be produced with high yield[6], [7], [8].

Several researches have been done on this topic, which can be divided into three types. The first type uses Monte Carlo simulation[1], [2]. This type is flexible, but time consuming, especially in the case where we must consider correlations of delays, such as correlations between interconnect delays in a net. Because, we must generate a large number of random values with correlations. The second type selects candidates for the critical paths and evaluates the distributions of those paths[4]. This type is efficient if the number of candidate paths is small. However, the number of paths may explode exponentially with respect to the number of gates, and the number of paths with a similar long delay tends to increase in the practical circuit. Therefore, if the number of candidate-paths is large, this type becomes inefficient, and if only a limited number of candidate paths are considered, the results may be inaccurate. The third type uses a systematic method to calculate the maximum delay to the output of each logic gate in a given combinatorial circuit[3], [5]. This type is most efficient with certain accuracy among these three types, but it is hard to remove the effect of the false paths. The first type also has a similar difficulty, if PERT-type algorithm[9] is used for calculating the maximum delay.

False path is a path which cannot be activated by any input vector[10], [11], [12], [13]. Paths other than false paths are called true paths. Since only the delays of true paths must meet the timing requirements, removing the effects of false paths is one of the most important issues in the static timing analysis.

False paths can be divided into two types; functional false paths and logical false paths. A path activated only by an input vector which never used in the specification of the circuit[11], and a multi-cycle false path[12] which need not to be activated within a single clock cycle, are functional false paths. For a path P, if there is no input vector I such that the logic value of the end terminal of P is determined by the logic values transmitted on P by I, then P is a logical false. Logical false paths can be divided into two types; delay independent logical false paths and delay dependent logical false path. Among these false paths, delay dependent logical false paths may be difficult to be identified, if the delay varies.

Functional false paths can be specified by designers. On the other hand, it is hard to find logical false paths[10], [13] by designers, and even by algorithms. However, we assume that logical false paths whose nominal delays are critical are specified in advance. Because, if the delay of each element (gate or interconnect) does not vary, there exists algorithms to determine whether a path is false or not, although they are not efficient, since they check the satisfiability of a Boolean expression written in a conjunctive normal form.

In this paper, given a CMOS combinatorial circuit and the specified false paths, we propose a technique to remove the effect of false paths in the statistical static timing analysis for the circuit. The technique is incorporated into our algorithm[5], which can take correlations into account.

2. Preliminaries

In order to find the distribution of the maximum delay of a CMOS combinatorial circuit, we represent the circuit by an acyclic graph $G = (V, E)$, as shown in Figure 1. In G, each terminal v of a circuit is represented by a pair of vertices $v0$ and $v1$, which are denoted in the figure by a white circle and a gray circle contained in an ellipse, respectively. Each primary input corresponds to a pair of sources in an ellipse, into which no edge comes, and each primary output corresponds to a pair of sinks in an ellipse, from which no edge goes out.

We denote the set of sources by S and that of sinks by T. Each logic gate is represented by a box, and each left and right ellipse in a box corresponds to an input and output terminals of the logic gate corresponding to the box, respectively.

Vertex $v0$ ($v1$) is called 0-vertex (1-vertex) of terminal v, and represents logic value 0 (1) transmitted to terminal v. Namely, we will represent the maximum delay $d(v,0)$ ($d(v,1)$) spent for transmitting logic value 0 (1) from a primary input to terminal v by the longest path length $d(v0)$ ($d(v1)$) from a source to 0-vertex (1-vertex) of terminal v. In order to do so, we generate an edge $e = (v,w)$ and assign delay $t(e)$ to the edge, if a logic value of w is determined by the logic value of v. It is easy to generate edge(s) corresponding to a net.

For a given logic gate, an input logic value is called a control signal, if the value of output terminal of the logic gate is determined by the input logic value, whatever values other input terminals have. Otherwise, it is called a non-control signal. For AND and NAND (OR and NOR) gates, logic value 0 (1) is the control signal and 1 (0) is the non-control signal. For an inverter, both 0 and 1 are control signals, and for an XOR gate, both 0 and 1 are non-control signals.

Let us consider the case when a non-control signal b is transmitted to all inputs v_i ($1 \leq i \leq k$) of a logic gate, and let b' be the logic value of the output w of the gate determined by b. In this case, the maximum delay $d(w,b')$ is calculated by taking the maximum as shown below,

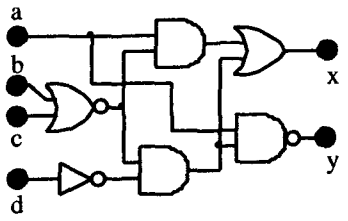
$$d(w,b') \quad = \quad max\,[\, d(v_i,b) + t(v_i,b) \mid 1 \leq i \leq k \,]$$

where $t(v_i,b)$ is the gate switching delay for setting w to be b' by the signal b of input v_i. Therefore, if we generate an edge $e_ib = (v_ib, wb')$ from each vertex v_ib to vertex wb', we can represent $d(w,b')$ by the longest path length $d(wb')$ to wb'.

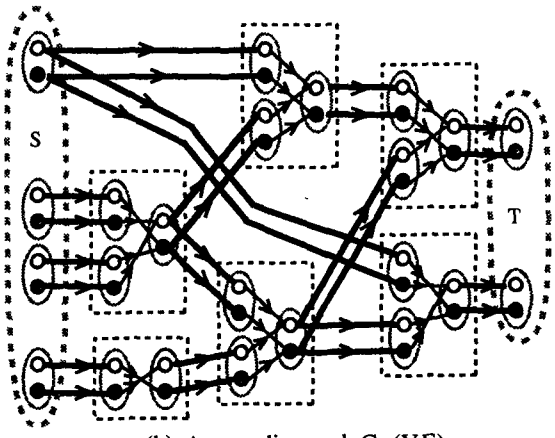
On the other hand, consider the case when a control signal c is transmitted to an input terminal v_i of a gate. In this case, the logic value c' of the output w of the gate is determined as soon as v_i becomes c. However, if all inputs other than v_i are non-control signal, w is set to c' after the delay equal to $d(v_i,c) + t(v_i,c)$, where $t(v_i,c)$ is the gate switching delay for setting w to be c' by the signal c of input v_i. Hence, the maximum delay $d(w,c')$ for w to be c' is obtained by taking the maximum among these delays $d(v_i,c)+t(v_i,c)$, $(1 \leq i \leq k)$. Thus, we can calculate $d(w,c')$ by the longest path length $d(wc')$, if we generate an edge $e_ic = (v_ic, wc')$ from each vertex v_ic to wc'.

Delay $t(e)$ of edge e thus generated is a stochastic variable, and is modeled by a normal distribution $N(\mu(e), \sigma^2(e))$, where $\mu(e)$ and $\sigma^2(e)$ are the mean and variance of the delay, respectively. Since the distributions of delays of a certain edges are not independent[5], [14], we introduce a correlation coefficient $\rho(e', e") \neq 0$ between delays $t(e')$ and $t(e")$ of edges e' and $e"$ such that delays $t(e')$ and $t(e")$ are not independent.

For the acyclic graph $G = (V,E)$ representing a given combinatorial circuit and the information of the edge delays, we want to find the mean $Exp[MaxD]$ and variance $Var[MaxD]$ of the longest delay $MaxD = max[\, d(w) \mid w \in T \,]$ from a source to a sink. Our algorithm proposed in [5] can compute these values and treat correlations between delays $t(e)$ and $t(e')$ of edges e and e' as well as correlations between the maximum delays $d(w)$ and $d(w')$ to vertices w and w'. The algorithm computes mean $Exp[d(w)]$ and variance $Var[d(w)]$ of the longest path length $d(w)$ to each vertex w selected in topological order, and also necessary correlation coefficients between $d(w)$ and $d(v)$ of other vertex v and between $d(w)$ and $t(e)$ of an edge e. Although the worst-case time complexity of the algorithm is $O(|E|^2)$, we have found that the algorithm is efficient and effective to the practical circuits, where $|E|$ is the number of edges.



(a) A given combinatorial circuit.



(b) An acyclic graph G=(V,E).

Figure 1. The graph representing a given circuit

3. Elimination of False Paths

Let us describe a technique to modify graph $G = (V,E)$, so that the specified false paths are eliminated but true paths still remain.

A path is a directed path, and the first and the last edges (vertices) of a path are the starting and ending edges (vertices) of the path, respectively. A cause partial path P is a minimal path such that all the paths from a source to a sink containing P are false paths[11]. We assume that all false paths to be eliminated are specified by the list $CP = \{\, P_i \mid 1 \leq i \leq f \,\}$ of cause partial paths P_i.

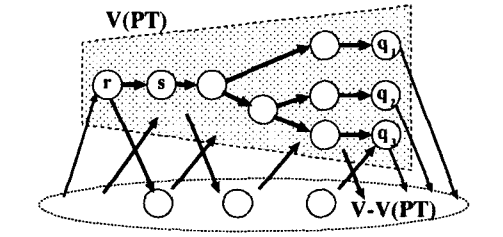
Since two false paths with a common edge may contain a true path, it is difficult to eliminate all false paths in CP at once. Therefore, we first select paths in CP with the same starting edge or the same ending edge, which satisfy the following conditions, and delete those paths from CP. Namely, let SP be the set of the selected paths satisfying the following conditions, and set $CP = CP - SP$.

(A) The paths in SP form a rooted tree PT, such that PT is an out-tree (in-tree) with only one edge going out from (coming into) the root, if the paths contain the same starting (ending) edge.

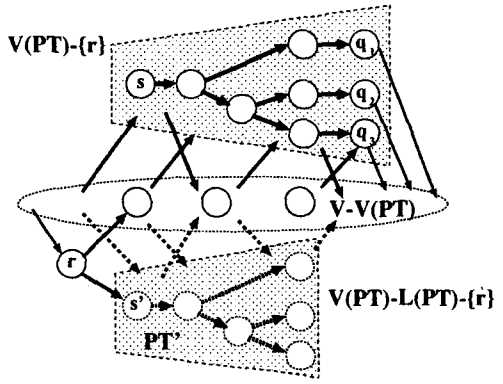
(B) There is no edge in G, which connects two vertices in PT but is not contained in PT.

Let $V(PT)$ and $E(PT)$ be the set of vertices and edges of PT, respectively. From the definition of PT, we can see that any path in G other than the paths containing a path P_i in SP includes a vertex not contained in $V(PT)$. Moreover, we can easily find such set SP and tree $PT = (V(PT), E(PT))$.

In order to show a technique to eliminate all paths in SP, we consider the case where PT is an out-tree (see Fig. 2). Let $L(PT)$ and $LE(PT)$ be the sets of the leaves and the edges coming into leaves in PT, respectively.



(a) Graph G



(b) Graph $G(PT)$

Figure 2. Elimination of Paths in PT

(1) First, we make a copy of PT, and modify it by deleting the root r, the starting edge (r, s), $L(PT)$ and $LE(PT)$. We denote the modified copy by PT', and call the edges contained in PT' duplicate edges.

(2) Next, we add PT' to G and obtain the graph $G(PT)$.

(3) Then, in $G(PT)$, we delete edge $e = (r, s)$ and add edge $e' = (r, s')$ instead, where s' is the vertex in PT' corresponding to s. The delay $t(e')$ of e' is the same as that of e.

(4) Moreover, for each edge $e_i = (v, w)$ coming into a vertex w in $V(PT)$ from a vertex v in the outside of PT and for each edge $e_o = (w, v)$ coming into a vertex v in the outside of PT from a vertex w in $V(PT)$, we add new edges $e_i' = (v, w')$ and $e_o' = (w', v)$, where w' is the vertex in PT' corresponding to w. We call $e_i' = (v, w')$ and $e_o' = (w', v)$ duplicate edges.

(5) Finally, if a new source or a new sink exists in $G(PT)$, we delete such sources and sinks together with incident edges, until all the sources and sinks become same as G.

(6) In order to apply our algorithm in [5], we assign to each duplicate edge the same delay as the original edge, and to the pair of delays of an original edge and the duplicate edge the correlation coefficient equal to 1.

In the graph $G(PT)$ thus obtained, there is no path containing a cause partial path P_i in SP, but any path in G which does not contain a path P_i in SP is not eliminated. Since we duplicated edges, $G(PT)$ may have two paths corresponding to a path in G, but no new path is created in $G(PT)$, that is, any path in $G(PT)$ corresponds to a path in G.

The elimination process described above is repeated until CP becomes empty. In this repetition, if a path P_i in CP not contained in SP is duplicated, then the duplicate path must be added to CP.

4. Experimental Results

Since the proposed technique duplicates a path, one path contributes more than one time in the calculation of the distribution of the maximum. Hence, we first examine the effect of this duplication. To do so, we calculated the following values by our algorithm in [5] under the condition that $Exp[x] = Exp[y] = 20$, $10 \leq Exp[z] \leq 30$, and $Var[x] = Var[y] = Var[z] = 2$. This shows the situation such that a path with the length of $x + y$ is duplicated 10 times.

$$t_1 = max[\, x,\ z \,] + y$$
$$t_i = max[\, t_{i-1},\ x + y \,]\ (i = 2, 3, ..., 10)$$

Since x and y in t_i are same as those in t_1, all of the values t_i must have the same distribution, but if we ignore the correlations, that is, if we treat x and y in t_i ($1 \leq i \leq 10$) are independent, then the mean $Exp[t_i]$ and the standard deviation $\sqrt{Var[t_i]}$ of t_i have errors up to 11% and -46%, respectively, in the range of $Exp[z] = 10$ through $Exp[z] = 30$. However, since our algorithm can treat correlations, the errors of the mean and the standard deviation are less than -0.5% and $+5\%$, respectively.

We have applied our algorithm to a few circuits in ISCAS85 benchmark data and a circuit denoted by

"c_{exam}" in Table 1, which has a false path created manually. Table 1 shows the size of the graph representing a given circuit and the modified graph in which all specified false paths are eliminated. Since circuit c499 has many false paths, the modified graph could not be stored in the main memory of 512[*MByte*]. Therefore, we applied the proposed algorithm to c880 and c_{exam}, under the condition that the means $\mu(e)$ of each edge-delay $t(e)$ corresponding to an interconnect delay and gate switching delay are 30[*psec*] and 20[*psec*], respectively, and the standard deviation $\sigma(e)$ of each edge delay is equal to $\sigma(e) = 0.1\mu(e)$ or $\sigma(e) = 0.2\mu(e)$. Table-2 shows the mean and the standard deviation of the critical delay of c_{exam} calculated before and after false paths elimination. For c880, no difference was obtained under the condition of our experiments, and hence we omit the results. From the results, we can see that the distribution of the critical delay can be affected by the false paths, especially when the percentage of false paths is large in the paths with long delays which influence on the critical delay.

Table 1. Modified graph

Circuit	Graph	False Paths	Modified Graph
c499	$\|V\| = 1302$ $\|E\| = 2048$	181	too large
c880	$\|V\| = 2345$ $\|E\| = 2916$	4	$\|V\| = 2457(+4.78\%)$ $\|E\| = 3158(+8.30\%)$
c_{exam}	$\|V\| = 78$ $\|E\| = 84$	1	$\|V\| = 79(+1.28\%)$ $\|E\| = 87(+3.57\%)$

Table 2. Distribution of the critical delay

		with false paths		no false paths	
	σ(e)	mean[psec]	s.d.[psec]	mean[psec]	s.d.[psec]
c_{exam}	0.1μ	211	4.55	211	4.57
	0.2μ	223	9.11	223	9.13

Since we could not obtain distinguishing results for c499 and c880 by the proposed algorithm, we selected candidate true paths for the critical paths by assuming that all edge delays are constant, and calculated the distribution of the critical delay for these selected true paths. Table 3 shows the results. The numbers of the selected paths are 43 and 52 for c499 and c880, respectively. Table 3 also shows the results obtained from the original graph containing a false path, from which we can see that the result of this method is not accurate enough. Because, the results for c880 with false paths can be regarded as the correct results.

Table 3. Distribution of the critical delay

		with false paths		true paths	
	σ (e)	mean[psec]	s.d.[psec]	mean[psec]	s.d.[psec]
c499	0.1μ	595	4.73	574	6.31
	0.2μ	640	9.46	598	12.6
c880	0.1μ	1230	12.7	1220	11.8
	0.2μ	1260	25.4	1250	23.5

5. Conclusions

In this paper, we proposed a technique to eliminate the specified false paths in the statistical static timing analysis for a CMOS combinatorial circuit, and showed the effect of the technique through a few experiments. Since the technique duplicates edges in the graph representing a given circuit, the size of the graph may explode in the case when the number of false paths is large. In such a case, a practical method may be to select the candidate of critical paths (true paths) and to evaluate the distribution of the maximum delays of these paths. However, since two true paths with a common edge may contain a false path, we must take false paths into consideration somehow. If the number of true paths to be considered is manageable size, we may treat each true path separately. But, the number of paths may explode exponentially, and hence this method may fail in such case. We are now investigating these problems, and will report a practically best way to treat false paths in the statistical static timing analysis.

References

[1] H.-F. Jyu, S. Malik, S. Devadas, and K. Keutzer, *IEEE Trans. VLSI Systems*, 1(2):126–137, 1993.

[2] J. Rung-Bin and W. Meng-Chiou, *Proc. 11th Int. Conf. on VLSI Design*, pages 507–513, 1997.

[3] M. Berkelaar, *Proc. Int. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU97)*, pages 15–24. ACM, 1997.

[4] H. Matsunaga, D. Mizoguchi, and H. Yasuura, *Proc. DA Symp. '99*, pages 77–82, JIPS, 1997 (in Japanese).

[5] S. Tsukiyama, M. Tanaka, and M. Fukui, *IEICE Trans. Fundamentals*, vol.E84-A, no.11, pp.2746-2754, 2001.

[6] M. Hashimoto and H. Onodera, *Proc. Workshop on Synthesis And System Integration of MIxed Technology (SASIMI 2000)*, pages 77–82, 1999.

[7] E.T.A.F.Jacobs and M.R.C.M.Berkelaar. *Proc. Design Automation and Test in Europe (DATE2000)*, pages 283–290, 2000.

[8] S. Nishimoto, S. Tsukiyama, M. Tanaka, and M. Fukui, *Tech. Report of IEICE, VLD 2000-130*, pp.21–26, 2001 (in Japanese).

[9] T. Sekine, *PERT · CPM*, Nikka-Giren, 1992. (in Japanese)

[10] H. Chen and D. Du, *IEEE Trans. Computer-Aided Design of ICs and Systems*, vol.12, no.2, pp.196–207, 1993.

[11] K. Nishida, K. Okada, M. Ohnishi, A. Yamada, T. Kambe, false-path detection," *Proc. the 13th Workshop on Circuits and Systems in Karuizawa*, pp.257–262, 2000 (in Japanese).

[12] P. Ashar, S. Dey, S. Malik, *IEEE Trans. Computer-Aided Design of ICs and Systems*, vol.14, no.9, pp.1067–1075, 1995.

[13] S.T. Huang, T.M. Parng, and J.M. Shyu, *IEEE Trans. Circuits and Systems - 1: Fundamental Theory and Applications*, 43,5:386–396, 1996.

[14] D. Yanagi and S. Tsukiyama, *This Proceedings*, 2002.