False Path Aware Timing Yield Estimation Under Variability

Lin Xie, Azadeh Davoodi, Kewal K. Saluja, and Abhishek Sinkar
Dept. of Electrical & Computer Engineering, University of Wisconsin at Madison
Phone: (608) 265-1145, Fax: (608) 262-1267, Email: adavoodi@wisc.edu

Abstract— **Effects of fluctuations in circuit timing due to process and environmental variations are becoming increasingly important as we move into sub-45nm technology. Since the delay of each gate is dependent on its input vectors, the timing yield, the probability that the circuit meets the given timing constraint, varies with different primary input patterns. Traditional timing yield estimation approaches assumed worst-case delay models for each gate over all its input vectors, which results in much pessimism. To overcome the aforementioned problems, this paper proposes a Monte Carlo based approach which can obtain a much tighter lower bound on the circuit timing yield compared to the existing timing yield estimation techniques. Specifically, our approach builds multiple input-vector-dependent variation-aware delay models for each logic gate, and considers the impact of false paths, both static and dynamic false paths, which are carefully selected from the likely timing-critical paths under variability. We demonstrate gradual improvement in the estimated timing yield in the simulation results, and show that the timing yield computed using traditional worst-case delay models is highly pessimistic.**

I. Introduction and Motivation

Process and environmental variations result in significant deviations from the expected delay of a gate and of the circuit timing. Statistical Static Timing Analysis (SSTA) aims to determine the probability density function (PDF) of the delay of a design by accounting for actual statistics of the variations. Based on the obtained PDF of the circuit timing, we can compute the timing yield using its definition which is the probability that a circuit meets the given timing constraint. The majority of the existing works assume one variation-aware delay model per gate, which is typically characterized for the worst-case gate delay under all its possible input vectors [1], [2], [3]. Recent works assume one variation-aware delay model for each pin of a logic gate but do not differentiate between its input falling and rising transitions [4]. *Please note that in this paper, we refer to a pattern of the primary inputs by "primary input pattern", and refer to input pattern of the logic gates by their "input vector".*

However, in practice, the delay of each gate is highly dependent on its input vectors. Its delay sensitivities with respect to the process or environmental parameters are also different as its input vector changes, which is forecasted to deteriorate with further technology scaling [5]. Therefore, the PDF of the gate delay alters with its varying input vectors. Consider NOR2X4 gate as an example. Fig. 1 gives its four possible delay distributions under variations. As this figure shows, the delay distributions are quite different under those four input vectors of NOR2X4 gate.

At the circuit level, the PDF of the circuit timing can be highly erroneous if we do not consider the primary input patterns. Here, we take the inverter chain as an example. Traditional SSTA approaches assume one variation-aware delay model for each inverter which was for the worst-case (assuming falling transition). However, the inverters' signals alternate between rising and falling transitions on a chain. Assuming falling transitions for all inverters results in overly-pessimistic circuit timing.

As shown above, assuming worst-case variation-aware delay models for all the gates in the circuit simultaneously is highly inaccurate. Examples of previous research works, which consider the impact of

This research was supported in part by National Science Foundation under Awards 0811082 and 6741641.

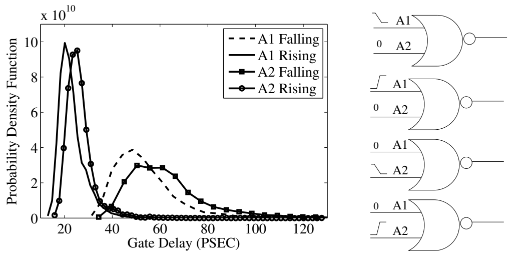

Fig. 1. Delay distributions of NOR2X4 under varying input vectors are quite different. Distributions were obtained using transistor-level Monte Carlo simulations on a 45nm commercial library [9], assuming variation in power supply, channel length, zero-bias threshold voltage with standard deviations of 10%, 5%, 10%, respectively, and nominal supply voltage of 1.1V.

primary input patterns under variations, are [7], [8]. Specifically, Liou *et al* [7] extracted a group of critical paths under variations, and focused on these paths to represent the entire circuit. However, the extracted paths might not be able to cover all the variation space. Lee and Wang [8] introduced "precision test vectors", which cause the worst-case circuit timing under variations. However, they only studied the impact of the input vectors on crosstalk effects but never considered that on the gate delays under variations.

To overcome the aforementioned problems, for each gate, we build different variation-aware delay models for its possible input vectors. We incorporate these input-vector-dependent variation-aware gate delay models into the existing Monte Carlo (MC) based SSTA framework. We investigate the impact of static and dynamic false paths on the timing yield. Our proposed approach can also be used to obtain the timing distribution. We list our contributions as follows.

1. For each gate, we develop variation-aware delay models, for all of its input vectors. We then extend MC based timing analysis to utilize these models for more accurate timing yield estimation.
2. We enhance our technique to accurately account for false paths. Since it is infeasible to extract all the false paths, we extract a subset of paths which are expected to be critical under variations [14]. We then identify and accurately capture the impact of false paths while accounting for primary input patterns. False path analysis includes both static false paths (identified once) and dynamic false paths (might vary for each variation sample). Note that dynamic false paths are also referred as timing false paths.
3. Our false-path analysis is accurate and accounts for all "complementary" paths in the circuit, unlike previous works on path-based SSTA that only focus on a group of extracted paths to represent the entire circuit [2], [7].

To the best of our knowledge, this is the first work to incorporate input-vector-dependent gate delay models for analysis under variability, and to accurately study the impact of static and dynamic false paths on the timing yield. Overall, our proposed approach can greatly reduce the pessimism in the existing timing yield estimation approaches. Simulation results show that on average, the timing yield increases from 54% using worst-case delay models to 80% using input-vector-dependent analysis. We show gradual improvement in estimated timing yield as we incorporate static/dynamic false paths.

The remainder of this paper is organized as follows. Section II introduces input-vector-dependent variation-aware gate delay models. Sections III and IV describe the Monte-Carlo based timing yield estimation using these delay models, and an overview of our proposed false path aware timing yield estimation approach, respectively. The details of our approach are discussed in Sections V-VI. We finish by presenting results and conclusions.

II. Input-Vector-Dependent Variation-Aware Gate Delay Models

We model variations similar to [3]: we express the variation of gate i in terms of G number of global die-to-die components (denoted by vector $\mathbf{X}_{1\times G}$), L number of within-die principle components (denoted by vector $\mathbf{Y}_{1\times L}$), and one zero-mean random term (i.e., Z_i). Note that the elements in $\mathbf{Y}$ and Z_i are obtained after Principle Component Analysis [10] or Independent Component Analysis [11] on within-die variations of all gates in the circuit. We also assume that $\mathbf{X}$, $\mathbf{Y}$, and Z_i are independent.

To simplify the notations, for each gate i, we represent its variations in a compact form $\mathbf{S}_i = [\mathbf{X}, \mathbf{Y}, Z_i]$. Existing SSTA approaches assume one variation-aware delay model per gate [1] (or per pin [4]), and ignore distinction between rise/fall transitions. However, this type of modeling might be too pessimistic. In this paper, we propose to build separate delay model for each pin and rising/falling combinations, **namely, input-vector-dependent variation-aware delay models**. For example, in Fig. 1, since we assume no multiple input switching, there are only four input vectors applicable to NOR2X4. This indicates that we should develop four input-vector-dependent variation-aware delay models for this gate. Specifically, we denote delay of gate i as a result of falling or rising transition at its output because of transition at input pin k with $D^F_{k\to i}$ and $D^R_{k\to i}$, respectively. Similar to [3], we use linear approximation to model the gate delay under variations:

$$D^{R/F}_{k\to i}(\mathbf{S}_i) \approx D^{k,R/F}_0 + \mathbf{A}^{k,R/F}_i \times \mathbf{S}_i \qquad (1)$$

To obtain the coefficients in Eq. (1), including $D^{k,R/F}_0$ and $\mathbf{A}^{k,R/F}_i$, we used Cadence Spectre to conduct transistor-level MC simulation for each logic gate in the library and the distributions of the process and environmental parameters. The simulation was conducted assuming rising or falling transitions at pin k. After obtaining the MC samples, we carried out linear regression to obtain these coefficients.

To show the pessimism in the existing worst-case variation-aware gate delay models, we apply the above procedures to obtain the input-vector-dependent delay models for the gates in a 45nm library [9]. We use 10K samples for each MC simulation, and the assumptions are given in the caption of Fig. 1. Each gate has one variation-aware delay model per input vector, with a corresponding mean and variance. Table I reports the comparisons on the mean and standard deviation of the delays of a gate over all its possible input vectors. Assuming a transition at pin i, we consider μ^i_w, μ^i_R, and μ^i_F corresponding to averages of worst-case, low-to-high, and high-to-low timing distributions, respectively. We report $\min_i(\mu^i_w)$, $\max_i(\mu^i_w)$, $\min_i(\mu^i_R - \mu^i_F)$, and $\max_i(\mu^i_R - \mu^i_F)$ over the pins of the gate. We also report $\min_i(\sigma_i)$, the minimum standard deviation of delay distributions over the pins of the gate.

As shown in Table I, for each gate type, the worst-case gate delays over different pins have a large variation. In addition, the rising gate delays are much larger than the falling gate delays. We also report $\min_i(\sigma_i)$, the minimum standard deviation of gate delays over all its input vectors, to show the impact of variations on gate delays.

TABLE I
THE VARIABILITY IN INPUT-VECTOR-DEPENDENT DELAY MODELS (PSEC)

Gate	$\min\limits_i(\mu^i_w)$	$\max\limits_i(\mu^i_w)$	$\min\limits_i(\mu^i_R-\mu^i_F)$	$\max\limits_i(\mu^i_R-\mu^i_F)$	$\min\limits_i(\sigma)$
INVX1	35.70	35.70	16.07	16.07	4.73
INVX16	32.35	32.35	11.19	11.19	7.40
NAND2X1	44.20	50.62	16.40	19.86	8.09
NAND2X4	38.35	45.50	13.24	17.09	6.57
NOR2X1	63.43	72.66	38.83	44.97	5.53
NOR2X4	53.84	63.25	30.65	36.93	5.96
NAND3X1	51.67	68.47	18.65	25.84	8.45
NAND3X4	44.95	63.85	14.50	23.20	10.55
NOR3X1	88.47	120.13	58.94	84.47	6.42
NOR3X4	74.83	110.60	46.99	72.94	6.99

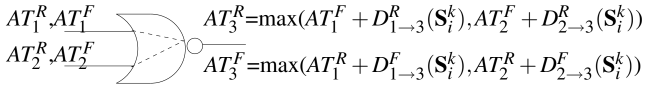
AT^R_1, AT^F_1
AT^R_2, AT^F_2
$AT^R_3 = \max(AT^F_1 + D^R_{1\to 3}(\mathbf{S}^k_i), AT^F_2 + D^R_{2\to 3}(\mathbf{S}^k_i))$
$AT^F_3 = \max(AT^R_1 + D^F_{1\to 3}(\mathbf{S}^k_i), AT^R_2 + D^F_{2\to 3}(\mathbf{S}^k_i))$

Fig. 2. At each MC analysis step, falling/rising arrival times are computed for a gate by evaluating its input-vector-dependent variation-aware delay models in that variation sample, $\mathbf{S}^k_i$.

III. Input-Vector-Dependent Monte-Carlo Analysis

A. Extending Traditional Monte Carlo Based Timing Analysis

We extend the traditional MC analysis to incorporate the input-vector-dependent variation-aware delay models obtained in Section II, and denote this framework as input-vector-dependent MC analysis. *Note that, in this paper, we refer to the MC timing analysis with incorporation of input-vector-dependent gate delay models by "input-vector-dependent MC analysis".*

Specifically, we first generate samples for the local and global random variables (i.e., $\mathbf{X}$, $\mathbf{Y}$, Z_i in Eq. (1)) according to their distributions. At each MC analysis step, we evaluate the input-vector-dependent variation-aware delay models in Eq. (1), and compute the delays of the logic gates in the circuit. We then compute the circuit timing for that variation sample according to the following procedures: we perform block-based timing analysis similar to the deterministic approach of [12] to incorporate input-vector-dependent delay models. At each gate i, we store a falling and a rising arrival time, denoted by AT^F_i and AT^R_i respectively. These arrival times of gate i are computed using the rising/falling arrival times at each of its fanins (which are already computed) and the corresponding falling or rising gate delay for that fanin pin. Fig. 2 illustrates an example.

B. Timing Yield Computation

Our objective is to compute the timing yield (i.e., Y), the probability that circuit timing T is smaller than given T_{cons}:

$$Y \triangleq Pr(T < T_{cons}) \qquad (2)$$

To compute the timing yield, at each MC analysis step, we compute the circuit timing using the described approach, and evaluate if T_{cons} is met. We compute the yield as the percentage of the times that circuit timing satisfies T_{cons} over all MC analysis steps.

Please note our MC approach is more time-consuming than block-based SSTA [1]. We target our approach at the sign-off stage for more optimistic analysis, similar to [8]. However, our MC approach can be implemented in parallel using multi-core and GPU platforms [13]. We also make the assumption that all primary input patterns and their switchings are equally probable.

C. Pruning of Input-Vector-Dependent Gate Delay Models

We can further improve our proposed input-vector-dependent MC by making the observation that an arbitrary gate in the circuit might never (logically) get excited by a subset of its input vectors. Consequently, the corresponding variation-aware models of those

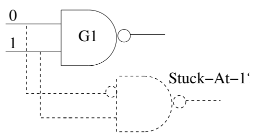

Fig. 3. Use of stuck-at-fault testing to detect the input transition at G1.

input vectors can get pruned out for a less pessimistic MC analysis. For example, in Fig. 2, if the transition $D^R_{1\to 3}$ never occurs, we can simplify $AT^R_3 = AT^F_2 + D^R_{2\to 3}$. As shown in Fig. 3, to test if a transition can occur for a gate, we can synthetically modify the netlist and conduct a stuck-at-fault test, which is done once and prior to analysis.

This type of pruning can identify the cases, when all the paths causing a certain rise or fall transition through a specific pin of a gate are static false paths because they can never get excited logically. In the previous example, all the paths going through pin 1 of gate 3 causing a rising transition are static false paths. In practice, it is more likely that only a few (and not all) paths going through pin 1 and causing a rising transition, to be static false paths. However, this situation can never get detected using a-priori gate input vector pruning. Therefore, instead of handling static false paths in such a specific manner, we will consider false paths more generically.

IV. Overview of Our Proposed Approach

The overview of our proposed false path aware timing yield computation approach is enumerated below and elaborated with an example in Fig. 4. Since it is challenging to extract *all* false paths in the circuit, our objective in this paper is to extract the false paths from a small set of paths which we expect to be critical under variations (have path yield less than a threshold Y_{th}). We denote this set as P_{crit}. We then conduct an accurate timing analysis with false path detection to *assess the impact* of these paths on the circuit timing yield.

Algorithm 1: Overview of Our Proposed False Path Aware Timing Yield Computation Approach

```
1. Extract $P_{crit}$=$\{P_i | Y_i \le Y_{th}; \forall P_i \in P_{cir}\}$ of critical paths.
2. Remove static false paths formed by path segments in $P_{crit}$:
   2.1 $P^{(static)}_{crit} = P_{crit} - P^{(excitable)}_{crit}$
   2.2 $P_{combined} \triangleq \cup P^{(static)}_{crit}$
   2.3 $P_E \triangleq P_{combined} - P^{(static)}_{combined}$
3. Compute the timing yield using input-vector-dependent MC analysis
   while assessing the impact of $P_E$ as well as the complementary circuit
   paths that are not in $P_E$.
```

At Step 1, we identify a set of paths (i.e., P_{crit}), which include all the paths with path yield smaller than a specified threshold Y_{th}. By setting Y_{th}, we can control the size of P_{crit}. The proposed path extraction approach is based on our previous work [14], in which we discuss bound-based identification of critical paths under variability. Even though in this paper we use the method of [14], we can still use any other critical path extraction algorithms, such as [15].

We then assess the impact of the extracted paths on the circuit timing. First, at Step 2, we remove static false paths (SFPs) from the selected critical paths to obtain a set of remaining paths P_E. Then, at Step 3, we conduct an accurate input-vector-dependent MC analysis on P_E with consideration of dynamic false paths, and account for the impact of the "complementary" portion of the circuit. The complementary circuit contains paths that do not belong to P_E.

Remark: Unlike previous works, such as [2], [7], we do not assume that the circuit can be solely represented by the set P_E. Instead, since the total number of paths in the circuit is large, we capture the impact of both P_E and complementary circuit during the input-vector-dependent MC analysis.

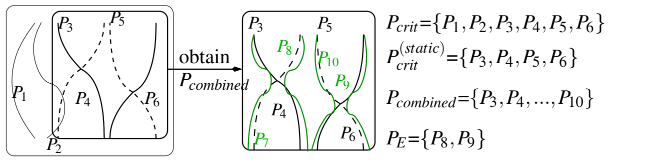
P_1
P_2
P_3
P_4
P_5
P_6
obtain
$P_{combined}$
P_3
P_5
P_8
P_{10}
P_9
P_4
P_7
P_6
P_{crit}=$\{P_1,P_2,P_3,P_4,P_5,P_6\}$
$P^{(static)}_{crit}$=$\{P_3,P_4,P_5,P_6\}$
$P_{combined}$=$\{P_3,P_4,...,P_{10}\}$
P_E=$\{P_8,P_9\}$

Fig. 4. Obtaining a set of extracted paths P_E from a subset of critical paths. We accurately assess the *impact* of P_E in computing the timing yield.

At Step 2, we identify the static false paths from the set P_{crit}, and denote these paths by $P^{(static)}_{crit}$ in Step 2.2. These are the *obvious* static false paths. However, there exist more static false paths if we consider all possible paths that can be formed by joining the segments of the paths which are extracted. Therefore, at Step 2.2, we define $P_{combined}$ as the set of all possible paths formed by joining the segments in $P^{(static)}_{crit}$. The intuition is that combined paths of static false paths are more likely to form static false paths themselves.

We use Fig. 4 as an example to illustrate the notations in Algorithm 1. We assume that $P^{(static)}_{crit} = \{P_3, P_4, P_5, P_6\}$, which indicates that P_3, P_4, P_5, P_6 are the static false paths among our extracted critical paths. Therefore, we can have the combined set $P_{combined} = \{P_7, P_8, P_9, P_{10}\}$. We further assume that among all the paths in $P_{combined}$, only P_7 and P_{10} are static false paths, which indicates that $P^{(static)}_{combined} = \{P_7, P_{10}\}$. We remove all the identified false paths and define our extracted set of paths as P_E=$\{P_8, P_9\}$ by end of Step 2.

Remark: As seen in this example (also Step 2.2 in Algorithm 1), our combined circuit does not include all excitable critical paths that were initially extracted (P_1, P_2 in the example). *If we define $P_{combined}$ as a set of paths formed by joining the segments in P_{crit}, we can get more accurate timing yield. However, since the size of P_{crit} is large, using the above approach to define P_{crit} will make our timing analysis framework to be very time-consuming.* Here, we define our combined paths by merging the segments of the "obvious" static false paths (i.e., $P^{(static)}_{crit}$) since they are more likely to form static false paths. Our objective in this paper is to accurately incorporate the impact of a small portion of important paths on the timing yield. This is in addition, to our input-vector-dependent MC analysis which will be applied to the entire circuit. We plan to analyze the impact of other static false paths that can be formed for future research.

Given this overview, next, in Section V, we discuss incorporation of static false paths during yield computation. In Section VI, we discuss consideration of dynamic false paths.

V. Impact of Extracted Paths on Timing Yield

In this section, we discuss the details of Step 3 in Algorithm 1. After identifying the set P_E of extracted paths which can get logically excited and are likely to be critical, we need to incorporate their impact on the timing yield. We use an input-vector-dependent MC analysis for this assessment. At each analysis step, if none of the paths in P_E fail the timing constraint for that variation sample, we still need to make sure that the "complementary" part of the circuit will not fail the timing.

To better understand the complementary circuit, we consider the example of Fig. 5. Here, we have shown four complementary paths, each having at least one different edge shared with the extracted paths. Some paths like P_1 and P_2 are completely non-overlapping with P_E, while other paths such as P_3 and P_4 have edges that are same with P_E. We formally define complementary paths P_C as the paths that have at least one edge which does not belong to any paths in P_E:

$$P_C = \{P_i | P_i \notin P_E \ and \ (P_i \cap P_j = \emptyset \ or \ P_i \cap P_j \in P_E; \forall j \in P_E)\} \quad (3)$$

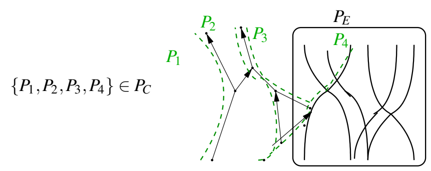

Fig. 5. Complementary paths P_C have at least one edge that is different from the edges in the extracted path set P_E.

The details of our approach (Step 3 in Algorithm 1) is given below in Algorithm 2. During MC analysis, we compute the timing yield by counting the number of the analysis steps (or variation samples) that the circuit fails the timing constraint. At each step, if we find at least one path that fails the timing constraint, we declare this step as failing, skip testing the remaining paths and move to the next step.

Algorithm 2: Input-Vector-Dependent MC Incorporating Impact of P_E (Step 3 in Algorithm 1)

```
1. Failure=0
2. For each MC sample from 1 to NumSamples:
    2.1 If at least one path in complementary circuit $P_C$ fails the timing
        constraint: Failure++;
    2.2 Else for each path $P_e \in P_E$
            If $P_e$ fails timing constraint and is not dynamic false path:
            Failure++;
3. Timing yield = 1 - Failure/NumSamples;
```

More specifically, at each analysis step, we first analyze if a failure can happen in at least one path in the complementary circuit P_C (Step 2.1). We will soon show that this step can be done very efficiently. If no failure is detected in the complementary circuit, we check for timing failure in the extracted paths P_E (Step 2.2). If we detect failure in one path $P_e \in P_E$, we further test that P_e is not a dynamic false path for this variation sample. We discuss dynamic false path detection in detail in the next Section. In the end, the timing yield is computed from the number of failing samples (See Step 3).

Efficient Detection of Failure on Complementary Circuit:

At Step 2.1, we need to evaluate if there is at least one path in P_C that will fail the timing constraint. Enumerating all the paths in P_C is an infeasible task since the size of our extracted paths P_E is small and therefore the size of P_E is large. In this paper, we propose an efficient approach to detect failing P_Cs using two rounds of "forward" and "backward" block-based timing analysis. At each MC analysis step, we follow the procedures below:

- We conduct input-vector-dependent MC analysis using a forward traversal (See Section III), and store worst-case rising/falling arrival times (ATs) at the output of each gate.
- We then conduct a reverse timing analysis from the outputs to inputs and store rising/falling backward arrival times (denoted by RATs) at the output of each gate. To find the RAT at a gate's output, we find the maximum of the falling RATs and the maximum of the rising RATs of its fanouts. When traversing the RATs backward from a gate's output to its fanins, we use one gate delay model corresponding to the input pin that we determined to have maximum AT in the forward traversal.
- For each edge e_{ij} connected from node i to node j in the circuit, we compute AT_i and RAT_j for both rising and falling cases. We consider $AT_i + RAT_j$, the longest path delay that contains e_{ij}, for each edge e_{ij} that is not a segment in any of the extracted paths (i.e., $e_{ij} \notin P_E$).
- We examine all edges that are outside the P_E set and as soon as we find one edge e_{ij} in which $AT_i + RAT_j$ is larger than the timing constraint, we have detected timing failure. We then move to the next MC analysis step.

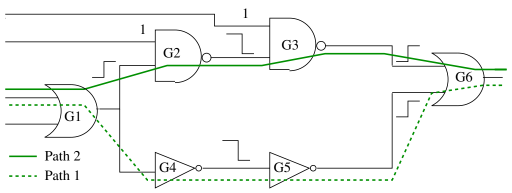

Fig. 6. An example of dynamic false path

Please note that based on the definition of P_C, all the paths that contain e_{ij} should be in C_p. Next, we discuss checking if a path is dynamic false path as required in Step 2.2 in Algorithm 2.

VI. Incorporation of Dynamic False Paths

In this Section, we first discuss the impact of dynamic false paths (DFPs, also referred as timing false paths) on the circuit timing and the related challenges in presence of variations. We then introduce our approach to identify DFPs and incorporate them in the timing yield computation to elaborate Step 2.2 in Algorithm 2.

A. Impact and Challenges Under Variability

Dynamic false paths are the paths, which *can be* logically sensitized but are "masked" by other paths because of circuit timing. Take Fig. 6 as an example. Two paths (i.e., P_1, P_2) belong to set P_E. We assume no variations, and that P_2 violates the timing constraint due to rising transition at the input of G1. We also assume that the timing constraint is satisfied for the remaining cases–when there is a falling transition at input of P_2 and when there is a rising or a falling transition at input of P_1. Using path-by-path timing analysis on these extracted paths, we observe that the circuit fails timing constraint due to the P_2 rising transition. However, P_1 and P_2 diverge at G1 and reconverge at G6. Since G6 is an OR gate, a rising transition at the output of G6 is generated at $\min(AT_3^R + D_{3\to 6}^R, AT_5^R + D_{5\to 6}^R)$. In other words, when considering two paths, the rising arrival time of G6 and consequently at output G6 is always decided by P_1, which always satisfies the timing constraint. We conclude that P_2 is a DFP, even though it can get logically excited. The above example is based on the nominal case. However, in presence of variations, it is possible that P_1 and P_2 both violate the timing constraint for some variation samples. Therefore, P_2 will not always be masked by P_1. Consequently, for each variation sample, we need to detect DFPs separately.

B. Framework Overview and Identification of DFPs

Recall that in our yield computation framework, at each MC analysis step, we need to verify whether the extracted path $P_2 \in P_E$, which violates the timing constraint, is DFP or not (Step 2.2 in Algorithm 2).

Here, we first give an overview of our framework to identify the DFPs. We propose the following procedure to deal with DFPs. At each MC analysis step, we divide the extracted path set P_E into two subsets, 1) paths that meet the timing constraint, 2) paths that violate the timing constraint. Next, for each violating path P_2 in subset 2, we verify if any of the paths P_1 in subset 1 can mask this violating path. As soon as we find one such path in subset 1, we know P_2 is DFP and we move to verifying if the next path in subset 2 is DFP. *After going through all the paths in subset 1, if P_2 still violates the timing constraint, we can claim that the circuit violates timing constraint at this MC analysis step and do not need to go through all other paths in subset 2.*

We then discuss the simplified problem if at a MC analysis step, a violating path P_2 can be masked by a non-violating path P_1.

We first identify all reconvergent paths formed by P_1 and P_2 (e.g., reconvergence formed by G1 and G6 in Fig. 6). These reconvergences can be easily identified between *consecutive* overlapping gates since we are dealing with only two paths. For an identified reconvergence, we consider the gate at the converging point (i.e., G6 in Fig. 6). We update the rising/falling arrival times at the output of this gate as defined by both P_1 and P_2 while incorporating the logic function of the converging gate. In Fig. 6, since G6 is OR gate, we conclude $AT_6^R = \min(AT_3^R + D_{3\to6}^R, AT_5^R + D_{5\to6}^R)$ is the worst-case rising arrival time at G6. In our path-based analysis for P_2, we have priorly determined the rising arrival time of G6 to be $AT_3^R + D_{3\to6}^R$. Now if we find $AT_5^R + D_{5\to6}^R$ is a smaller quantity, we will update AT_6^R. Once we update the rising/falling arrival time of the reconverging gate, we update the rising/falling arrival times of P_2. Note the arrival time of P_1 has already met the timing constraint.

After the update, if we observe that both the rising and falling arrival times of P_2 meet the timing constraint, we conclude that P_2 is a DFP, and move to the next path that violates the timing constraint. If not, we consider the next reconvergence between P_1 and P_2, and if no other reconvergence exists, we evaluate whether P_2 can be masked by other paths which meet the timing constraint.

In our example, the reconverging gate was an OR gate, therefore, we were able to decrease the *rising* arrival times of its fanins. For other types of reconverging gates, we could do similar analysis based on their logic functions. For example, for AND gate, we can determine the worst-case *falling* arrival time at the reconverging gate to be the minimum of falling arrival times defined by the two paths. Note that we were able to do this analysis since we simplified our procedure to two paths at a time, one meeting the timing constraint, and the other violating the timing constraint. We were unable to incorporate this analysis in general block-based MC framework.

C. Summary and Discussions

Our proposed timing yield computation approach is based on assessing the impact of a small set of extracted critical paths. We consider all the possible paths that can be formed by combining these paths and removing static false paths. Next, at each step of our MC analysis, we apply our input-vector-dependent block-based analysis to the complementary part of the circuit, and then incorporate the impact of extracted paths via input-vector-dependent path-based timing analysis combined with analysis for DFPs.

Since the size of extracted paths is small, identifying their combinations and the static false paths is done efficiently. Furthermore, our DFP analysis is very efficient in practice because of reducing the problem to the simplified model of comparison between a violating path P_2 and a non-violating path P_1. Working with two paths makes identification of reconvergences trivial. Furthermore, as soon as we find P_2 to be masked by P_1, we stop the analysis for P_2. Also as soon as we find P_2 cannot get masked, we go to next MC analysis step. This is because our goal is to determine the timing yield. We only need to verify if P_2 violates the timing constraint.

VII. Simulation Results

A. Simulation Setup

We synthesize ISCAS'85 benchmarks using a 45nm library [9] and Synopsys Design Compiler. The synthesis is fairly conducted for minimum area under a timing constraint to ensure that the design is optimized and many paths are critical after this optimization. We build input-vector-dependent variation-aware delay models for this library with the assumptions given in Section II and the hierarchical model in [3] to capture the spatial correlations among the variations.

TABLE II
THE CONFIGURATIONS IN OUR TIMING YIELD ESTIMATION APPROACH

Bench	#G	Y_{th} (%)	$\#P_{crit}$	$\#G_{crit}$	$\#P_{crit}^{(static)}$	$\#G_{crit}^{(static)}$	$\#P_{combined}$
C432	266	87.78	2054	112	1989	112	5184
C499	739	87.99	2018	184	483	138	2088
C880	696	99.33	1966	399	62	78	98
C1355	692	87.89	2024	269	459	213	2794
C1908	759	86.39	2015	210	1385	210	5224
C2670	954	98.57	1870	325	20	45	20
C3540	1286	92.05	2056	200	1241	172	2121
C5315	2053	93.85	1664	283	552	207	656
C7552	2431	90.75	1539	346	396	266	1618

Our simulation setup is given in Table II. In our proposed input-vector-dependent MC analysis, we always generate 10,000 samples for variations, which is shown to be a reasonable number. We first extract all the paths P_{crit} for which the probability of satisfying the timing constraint is smaller than the value of Y_{th} reported in Column 3. The number of extracted paths (i.e., $\#P_{crit}$) is given in Column 4. The number of gates covered by these extracted paths (i.e., $\#G_{crit}$) is given in Column 5 which can be compared with the total number of gates in the circuit given (i.e., $\#G$) in Column 2. We then use the academic ATPG tool [16] to detect the SFPs from the extracted path (Step 2.1 of Algorithm 1). We report the number of SFPs (i.e., $\#P_{crit}^{(static)}$) and the number of gates covered by them in Columns 6 and 7. Finally, we consider the total number of paths that can be formed by combining the SFPs (i.e., $\#P_{combined}$) (Step 2.2 of Algorithm 1). We finally remove all the SFPs from this combined set of paths (Step 2.3 of Algorithm 1). We discuss Table II below:

- Some benchmarks, such as C3540, are highly *unbalanced*. The number of paths in G_{crit} is a small percentage of G (the total number of gates), unlike the *balanced* benchmarks, such as C1355.
- The number of SFPs highly varies from one benchmark to another. In C432, many of the paths in P_{crit} are SFPs, while in others, like C2670, only 20 out of 1870 paths are SFPs.
- We compare the number of SFPs before and after combination (i.e., $P_{crit}^{(static)}$ and $P_{combined}$). In some cases, such as C432, the number of paths in $P_{combined}$ is much larger than that in $P_{crit}^{(static)}$, while in other benchmarks, such as C2670 or C5315, the number of paths in $P_{combined}$ does not change much.

B. Discussions on Timing Yield Comparisons

To show the improvement of timing yield estimation using our proposed algorithm, we apply four different approaches which are listed as follows and make comparisons among each other.

1. WC: We use worst-case variation-aware gate delay models with pin incorporation [4] in a MC timing analysis framework and ignore false paths.
2. RF: We use input-vector-dependent variation-aware delay models and store worst-case rising and falling arrival times in our MC analysis framework (See Section III).
3. RF_SFP: We consider static false paths in the previous RF approach. This is essentially our discussed framework without consideration for dynamic false paths.
4. RF_SDFP: We consider the RF approach followed by both static and dynamic false path analysis.

Table III lists the circuit timing yield of these implemented approaches in Columns 2 to 5. We make the following observations:

- The RF approach improves timing yield on average from 54% to 76% compared to WC. This is solely due to the consideration of input-vector-dependent variation-aware delay models.

TABLE III
DEMONSTRATING IMPROVEMENT IN TIMING YIELD AS WE CONSIDER INPUT-VECTOR-DEPENDENCE AND STATIC AND DYNAMIC FALSE PATHS

	Timing Yield (%)				Yield Loss (%)	
BENCH	WC	RF	RF_SFP	RF_SDFP	YL_CMPL	YL_EXT
C432	50.07	70.55	83.90	90.14	9.86	0
C499	50.14	74.71	74.71	74.71	25.29	0
C880	50.07	73.27	73.48	73.48	26.52	0
C1355	50.11	75.78	76.12	76.12	23.87	0.01
C1908	50.13	73.31	73.34	77.05	15.19	7.76
C2670	50.03	71.41	71.41	71.41	28.59	0
C3540	66.40	85.00	86.80	89.27	10.68	0.05
C5315	66.45	86.33	90.43	90.50	8.97	0.53
C7552	50.09	73.57	73.59	79.64	20.36	0
ave	53.72	75.99	78.20	80.26		

- The incorporation of SFPs in RF_SFP can further improve the timing yield for C432, C3540, and C5315 by 13.90%, 1.80%, and 4.10%, respectively.
- After considering DFPs over the subcircuit composed by SFPs, the timing yield increases by 6.24%, 3.71%, 2.47%, 6.05% for C432, C1908, C3540, and C7552, respectively.

Note that the percentage comparisons denote the *actual* increase in the value of timing yield (and not an improvement ratio). We conclude that it is necessary to incorporate input vectors and false path analysis during timing yield estimation.

We next discuss the data for C432 benchmark in detail. Previously from Table II, we observe that in this benchmark, 1989 out of 2054 extracted paths (about 96.84%) are static false paths. Further, the number of paths in $P_{combined}$ is 5184, out of which about 5000 paths are static false paths. Therefore, we are able to get a 13.90% timing yield improvement from RF to RF_SFP when we consider static false paths in timing yield estimation. Furthermore, using dynamic false path analysis, we are able to improve yield by 6.24%.

Next, we discuss an alternative way to analyze our data. Consider the total yield loss in the circuit which corresponds to the MC steps when there is a failure in timing.

We consider total yield loss to be the summation of the yield loss in our extracted paths (i.e., P_E), as well as the yield loss in the complementary circuit (P_C which includes all the paths that are different from the set of extracted paths). Column 6 of Table III lists the timing yield loss caused by the complementary circuit, which we denote by YL_CMPL. To compute YL_CMPL, during our MC analysis, we count the percentage of the times that a timing failure happens in the complementary circuit. Note if a failure happens both at the extracted paths and the complementary circuit, we still include it in the YL_CMPL. We then report the quantity YL_EXT$\triangleq$100-(RF_SDFP+YL_CMPL) (%). Here, YL_EXT is an estimation of the degree of yield loss solely among the extracted paths, assuming RF_SDFP and YL_CMPL are not subject to high errors. Note that in many cases, YL_EXT is 0 which means after doing dynamic and static false path analysis, we find out that there is no failure on our extracted paths during MC analysis. However, if the yield was estimated using RF, we would have assumed a considerable degree of timing failure happening on the same extracted paths. The only benchmark with a significant value of YL_EXT (i.e., 7.76%) is C1908, but even for this benchmark, we are able to improve the timing yield from 73% to about 77% after considering dynamic false paths in the extracted paths.

Finally, we conduct a quick check to determine if false path analysis is really necessary. Consider the following examples. In benchmarks C499 and C2670, the summation RF+YL_CMPL is 100%. Therefore, we know that further analysis for false path would not provide any further improvement. We could have skipped false path analysis on these two benchmarks. Similarly in C5315, the summation RF_SFP+YL_CMPL is very close to 100%. We could skip analysis for dynamic false paths.

VIII. CONCLUSIONS AND FUTURE WORKS

We propose an input-vector-dependent Monte Carlo approach, which can obtain the timing yield with much less pessimism than current practice based on worst-case gate delay models. We focus our attention on the impact of a small but important portion of expected critical paths for which we conduct static and dynamic false path analysis. Overall, we show that we can improve the timing yield on-an-average from 54% to 80% by incorporating input-vector dependent gate delay models. We also show step-by-step improvement to elaborate the impact of static and dynamic false paths.

In the future, we plan to extend our framework to incorporate multiple input switching and crosstalk. We also plan to improve our path extraction algorithm to more effectively identify a set of critical paths for dynamic and static false path analysis.

REFERENCES

[1] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," *IEEE Trans. Computer-Aided Design*, vol. 27, no. 4, pp. 589–607, 2008.
[2] M. Orshansky and A. Bandyopadhyay, "Fast statistical timing analysis handling arbitrary delay correlations," in *Proc. IEEE Design Automation Conf.*, 2004, pp. 337–342.
[3] F. Najm, N. Menezes, and I. Ferzli, "A yield model for integrated circuits and its application to statistical timing analysis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 3, pp. 574–591, 2007.
[4] S. V. Kumar, C. V. Kashyap, and S. S. Sapatnekar, "A framework for block-based timing sensitivity analysis," in *Proc. IEEE Design Automation Conf.*, 2008, pp. 668–693.
[5] (2007) International technology roadmap for semiconductors. [Online]. Available: http://www.ITRS.net
[6] D. Tadesse, D. Sheffield, E. Lenge, R. I. Bahar, and J. Grodstein, "Accurate timing analysis using sat and pattern-dependent delay models," in *Proc. IEEE Design Automation & Test in Europe*, 2007, pp. 1018–1023.
[7] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng, "False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation," in *Proc. IEEE Design Automation Conf.*, 2002, pp. 556–559.
[8] L. Lee and L.-C. Wang, "An efficient pruning method to guide the search of precision tests in statistical timing space," in *Proc. IEEE International Test Conf.*, Oct. 2006, pp. 1–10.
[9] (2008) Nangate 45nm open cell library. [Online]. Available: http://www.nangate.com
[10] H. Chang and S. S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 9, pp. 1467–1482, 2005.
[11] J. Singh and S. S. Sapatnekar, "A scalable statistical static timing analyzer incorporating correlated non-gaussian and gaussian parameter variations," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 1, pp. 160–173, 2008.
[12] S. S. Sapatnekar, Ed., *Timing*. Boston: Kluwer Academic Publishers, 2004.
[13] K. Gulati and S. P. Khatri, "Accelerating statistical static timing analysis using graphics processing units," in *Proc. IEEE Asia South-Pacific Design Automation Conf.*, 2009, pp. 260–265.
[14] L. Xie and A. Davoodi, "Bound-based identification of timing-violating paths under variability," in *Proc. IEEE Asia South-Pacific Design Automation Conf.*, 2009, pp. 278–283.
[15] L.-C. Wang, J.-J. Liou, and K.-T. Cheng, "Critical path selection for delay fault testing based upon a statistical timing model," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 11, pp. 1550–1565, 2004.
[16] H. Lee and D. Ha, "On the generation of test patterns for combinational circuits," *Technical Report, Dept. of ECE, Virginia Polytechnic Institute and State University*, no. 12, 1993.