Fast Buffered Delay Estimation Considering Process Variations *

Tien-Ting Fang and Ting-Chi Wang

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
ttfang@tsmc.com, tcwang@cs.nthu.edu.tw

Abstract - Advanced process technologies impose more significant challenges especially when manufactured circuits exhibit substantial process variations. Consideration of process variations becomes critical to ensure high parametric timing yield. During the design stage, fast estimation of the achievable buffered delay can navigate more accurate and efficient wire planning and timing analysis in floorplanning or global routing. In this paper, we derive approximated first-order canonical forms for buffered delay estimation which considers the effect of process variations and the presence of buffer blockages. We empirically show that an existing deterministic delay estimation method will be over-pessimistic and thus result in unnecessary design rollback. The experimental results also show that our method can estimate buffered delay with 4% average error but achieve up to 149 times speedup when compared to a state-of-the-art statistical buffer insertion method.

I. Introduction

Buffer insertion has become an essential technique for interconnect optimization in physical synthesis. An industry study [1] predicts that at 65nm process technology, 35% of the cells on a chip will be buffers. Therefore, one must be able to efficiently and accurately estimate the impact of buffer insertion on a design in earlier stages, such as floorplanning. To this end, Alpert *et al.* [2] proposed a linear-time algorithm which is an extension of Otten's theory [3] in predicting interconnect delays for multifanout nets in the presence of buffer blockages. Accordingly, [2] can accurately assess the buffering impact in an early design stage without having to actually perform buffer insertion for nets. The estimation results of [2] are within 5% average error under the Elmore delay model when compared to a classic buffer insertion method, i.e., van Ginneken's algorithm [4]. According to [2], one could embed the fast estimation technique into a floorplanning or routing algorithm to estimate the timing cost for a net.

However, as critical dimensions are scaling quicker than the development of its controlling process technology, technology beyond 90nm exhibits significant variations [5]. Traditional analysis and optimization methods under nominal circuit parameters, e.g., [2] and [4], become too risky in the presence of process variations. Recently, there emerged many statistical static timing analysis (SSTA) approaches [6, 7], which greatly increase the analysis accuracy by propagating the distributions instead of single values. Based on these results, other statistical optimization techniques on gate sizing [8], power reduction [9], and even buffer insertion [10, 11, 12, 13, 14, 15] also surge to effectively alleviate the impact of circuit parameters deviation. To the best of our knowledge, none of any recent publications addresses the problem of variation-aware buffered delay estimation. We have conducted an experiment to see whether it is necessary to consider process variations in predicting buffered delay. We observed that the deterministic buffered delay estimation (*DBDE*) method [2] using the corner, $\mu+3\sigma$, which is the worst case corner for DBDE, will be too pessimistic. Take Figure 1 for example. If we choose the 99% timing yield, which is computed by a statistical buffer insertion (*SBI*) algorithm [15], as the timing constraint (as indicated by the blue straight line in Figure 1), the over-pessimistic result estimated by [2] in the worst case corner (as indicated by the red dot line in Figure 1) will force a designer to rollback design without knowing that there is 99% probability to satisfy the given constraint. The observation is reasonable because in recent technology generations, variability was dominated by the Back-End-of-the-Line (BEOL) or interconnect metallization and thus becomes more uncorrelated than before as mentioned in [5]. Consequently, traditional corner-based techniques, which perform optimization at some extreme values, are not applicable nowadays since the number of cases or corners required for confident coverage has grown tremendously. Therefore, it is necessary to develop a more effective approach for statistical delay estimation, which can give a designer a confidence value to decide whether he/she should rollback the design under some given constraint. As shown in Figure 1, the delay distribution computed by our statistical buffered delay estimation (*SBDE*) technique (to be presented in section III-B) is close to the one computed by *SBI*, and the timing yield of our distribution is 99.8% under the same given timing constraint as mentioned above.

In this paper, we study the problem of variation-aware buffered delay estimation. We base on the deterministic linear-time algorithm [2], and derive the first-order canonical forms for major operations in the algorithm by using some approximation techniques without losing the correlation between random variables. Equipped with the above techniques, we develop an efficient implementation of the variation-aware buffered delay estimation algorithm. The experimental results show that our method can estimate buffered delay with 4% average error but achieve up to 149 times faster than a state-of-the-art statistical buffer insertion method [15].

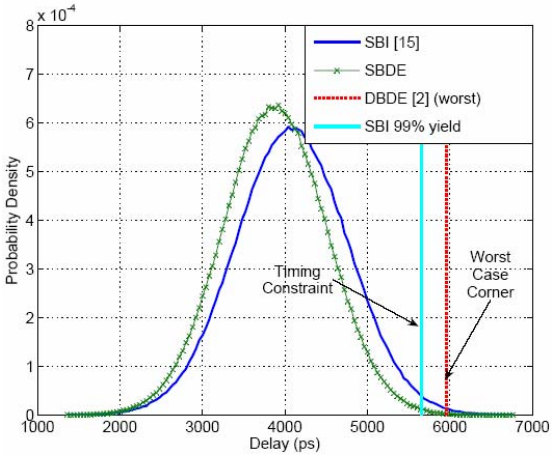

Figure 1: Comparison between deterministic and statistical buffered delay estimations.

* This work was partially supported by National Science Council under Grant No. NSC-94-2220-E-007-010, and Ministry of Economic Affairs under Grant No. MOEA-95-EC-17-A-01-S1-031.

The rest of this paper is organized as follows. Section II gives the preliminaries including delay and variation modeling, problem formulation, and some useful properties in statistics. Section III first reviews the deterministic buffered delay estimation method and then presents our statistical estimation technique. Section IV reports the experimental results and section V concludes this paper.

II. Preliminaries

A. Delay Model

Assume that the wire resistance and capacitance per unit length are denoted as R_w and C_w. For each buffer type b, we annotate the input capacitance as C_b, intrinsic delay as D_b, and output resistance as R_b. For simplicity, we adopt the Elmore delay model for delay computation and model wires by their equivalent π-models.

B. Process Variation Modeling

We adopt the first order approximation to characterize the impact of process variations on both device and interconnect characteristics, i.e., R_w, C_w, R_b, C_b, D_b. Such an approximation has been widely used in statistical timing analysis [6, 7] and statistical buffer insertion [12, 13, 15]. We represent device and interconnect characteristics, R_w, C_w, R_b, C_b, D_b, as random variables in a standard or canonical first-order form below:

$$A = A_0 + \sum_{i=1}^{k} a_i X_i = A_0 + \alpha^T X \quad (1)$$

where A is the characteristic, k is the number of variation sources, and X_i represent the sources of variations, such as inter-die global variation, intra-die spatial variation, and local random variation. The mean or nominal value of A is given by A_0, while the sensitivity of A with respect to X_i is given by a_i. For simpler representation, X is the vector of X_i's and α is the corresponding coefficient vector of X. By scaling the sensitivity coefficients, we can assume that X_i is in the standard normal or Gaussian distributions $N(0,1)$. We also assume all random variables X_i are mutually independent because the mechanisms of causing those variations are different. In practice if a random variable is correlated to another, by using a principal component analysis like [6], one can always transform the set of correlated random variables into a set of independent ones.

C. Problem Formulation

The input to our problem is a buffer library B, a set Blk of rectangular buffer blockages, a set of process parameters (i.e., R_w, C_w, R_b, C_b, D_b) with variations, and a routed topology of a net which is modeled as a routing tree $T = \{V, E\}$ and T may pass through buffer blockages. The problem aims to estimate the worst path delay distribution of the net under optimal buffering but without actually performing any buffer insertion.

D. Useful Properties in Statistics

Here we present three set of properties which are used to derive our equations in the next section.

Property 1. We are given two vectors θ_1 and θ_2 in R^n. $tr(\cdot)$ is the trace operation of a square matrix and equals the sum of the diagonal elements in the matrix. We have:

a. $tr\left(\theta_1\theta_2^T\right) = \theta_1^T\theta_2$

Property 2. Let X and Y be random variables and k be a scalar. $E(\cdot)$ is the expected value of a random variable. We have:

a. $E\left(k\right) = k$

b. $E(kX)=kE(X)$

c. $E(X+Y)=E(X)+E(Y)$

Property 3. We are given a random vector X in R^n where elements in X are all random variables in the standard Gaussian distribution and are mutually independent. $E(\cdot)$ is the expected value of a random variable. For any vector θ in R^n and any square matrix A in $R^{n\times n}$, we have:

a. $E\left(X\right) = 0$

b. $E\left(X^TAX\right) = tr\left(A\right)$

c. $E\left(X^TAX\theta^TX\right) = 0$

d. $E\left(\left(X^TAX\right)^2\right) = 2tr\left(A^2\right) + tr\left(A\right)^2$

The proofs of Property 3(b)(c)(d) can be found in [16], and it has also been adopted in [15].

III. Buffered Delay Estimation

A. Deterministic Buffered Delay Estimation

The linear-time estimation technique proposed in [2] is demonstrated that it only produces few percents error when compared to a buffer insertion method [4]. However, we observed that by ignoring intrinsic buffer delay (which is the case in [2]), the estimation may lead to 26.42% average error in practice[1]. As a result, in the following review of the method in [2], we incorporate the intrinsic buffer delay in the estimation, and denote the modified version as *DBDE* (which stands for Deterministic Buffered Delay Estimation).

In order to fast and simply predict interconnect delays for multi-fanout nets in the presence of blockages, *DBDE* is based on the following key assumptions which actually impose small tolerable error when compared to an actual buffer insertion solution. The reasons for making these assumptions, as listed in Table I, will be explained below.

TABLE I
Assumptions in *DBDE*

(a) Single buffer type.
(b) Infinitesimal decoupling buffers.
(c) Smaller blockages ignored.
(d) Larger blockages front-and-back buffering.

DBDE's main idea is to compute the worst path delay at the source in a single bottom-up tree traversal by decomposing the tree edges as out-blockage and in-blockage ones. Whenever an edge intersects with the boundary of a blockage, the edge will be broken into two edges. As a result, each edge lies either completely inside (as an in-blockage edge) or outside blockages (as an out-blockage edge). Similarly, a tree node lies inside a blockage is defined as an in-blockage node; otherwise, it is defined as an out-blockage one. Note that a tree node lies on the boundary of a blockage is identified as an out-blockage node.

The delay of an out-blockage edge is determined by a closed-form formula as shown in Equation (2) where L_e is the wirelength of the out-blockage edge. Note that the intrinsic buffer delay has been incorporated into Equation (2).

$$D(e)=L_e(R_wC_b+R_bC_w+\sqrt{2R_wC_w\left(R_bC_b+D_b\right)}) \quad (2)$$

One can see that the delay is a linear function of the edge wirelength because the Elmore delay becomes linear to the wirelength after optimal buffering is performed. Note that the more buffer types there are actually in the library, more precisely the single buffer type approximation[2] can be. Consequently, Equation (2) only uses a

[1] We suspect that why the delay error reported in [2] is very small is because the buffer insertion method [4] might also ignore the buffer intrinsic delay as [2] does.

[2] We empirically use the largest buffer in our estimation.

single buffer type in predicting optimal buffered delay, which corresponds to assumption (a). Additionally, Equation (2) is derived from a two-pin net, and thus need to set up the assumption (b) for a multi-fanout net. In the derivation of Equation (2), one can also derive the optimal spacing L_{opt} between buffers. We have

$$L_{opt} = \sqrt{\frac{2(R_b C_b + D_b)}{R_w C_w}} \quad (3)$$

As for the delay of an in-blockage edge, there are two scenarios to be considered. 1) For the edge wirelength L_e smaller than the optimal spacing L_{opt}, we can treat the edge as an out-blockage one and use Equation (2) for delay estimation because buffers can be placed (e.g., the two buffers are placed at the front and back of blockage b_1 in Figure 2 (a)) or potentially sized (e.g., the two buffers placed at the front and back of blockage b_2 are downsized in Figure 2 (a)) in such a way as to avoid the blockage, and doing so only suffers a negligible delay penalty. This scenario corresponds to assumption (c). 2) For the edge wirelength L_e larger than or equal to the optimal spacing L_{opt}, the buffers will be best placed right before and right after the blockage (e.g., the two buffers are placed at the front and back of blockage b_3 in Figure 2 (a)) to minimize the quadratic effect of delay, and it corresponds to assumption (d). Based on these assumptions, the original configuration of buffer insertion with multiple buffer types, as depicted in Figure 2 (a), can be translated to a simpler one, as depicted in Figure 2 (b), which greatly simplifies and accelerates the estimation.

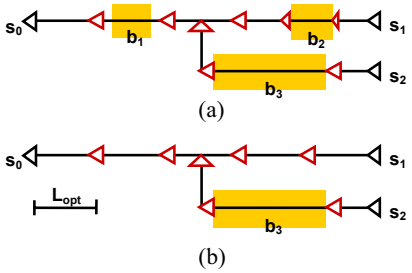

Figure 2: Configuration after buffer insertion (a) without assumptions, (b) with assumptions.

The pseudo code of *DBDE* is illustrated in Figure 3, which is adopted from [2] and has been slightly modified for more accurate delay calculation. The modifications are as follows: First, rather than merely recording in-blockage wire capacitance, the algorithm records the entire downstream capacitance in the variable $c(v)$. In addition, we reset $c(v)$ as the buffer input capacitance when entering or exiting blockages according to assumption (d), as shown in lines 11 and 17 in Figure 3. Finally, we incorporate the intrinsic buffer delay in both the linear and Elmore delay calculations corresponding to lines 8 and 12 in Figure 3. Note that the buffer intrinsic delay should be subtracted if the source is located out of a blockage, as shown in line 20 in Figure 3. As noted by [2] the complexity of *DBDE* is $O(|V|)$.

In *DBDE*, the delay is computed piecewise in the following eight steps: During the bottom-up tree traversal, **step 1**, corresponding to line 3, initializes $d(v)$ to zero and $c(v)$ to C_b when v is a sink. For each children of v, **step 2** in line 5 prepares $c(v)$ as zero for cumulating the downstream capacitance in the loop. When the edge locates out of a blockage, **step 3** in line 8 calculates the linear delay of the edge. As for the in-blockage edge, **step 4** first resets $c(v)$ as C_b as shown in line 11 if v is the entry of a blockage, and then calculates the Elmore delay of the edge which corresponds to line 12. After calculating the delay of an edge, **step 5**, corresponding to line 13, cumulates the downstream capacitance into $c(v)$. After traversing all the children of v, **step 6** picks the maximum delay for $d(v)$ as shown in line 14. If v is an exit of a blockage, **step 7** adds the delay resulted from a buffer as shown in line 16 and resets $c(v)$ as C_b as shown in line 17. Finally, **step 8** subtracts D_b from $d(v)$ if v is the source without locating in blockages as shown in line 20. The reason of taking **step 8** is that the delay of a net excludes the source intrinsic delay which is included in the calculation of Equation (2).

Inputs: $T(V, E)$ – Given Steiner tree
Blk – set of blockages
R_b, C_b, D_b – output resistance/input capacitance/intrinsic delay of buffer b
R_w, C_w – wire resistance/capacitance per unit length

Variables: $d(v)$ – delay at node v
$d_i(v)$ – delay at node v on path to child u_i
$c(v)$ – capacitance downstream from v

Let: $\alpha = R_w C_b + R_b C_w + \sqrt{2 R_w C_w (R_b C_b + D_b)}$

```
1  for each v ∈ V (in bottom-up order) do
2  if v is a sink,
3    set d(v) = 0 and c(v) = C_b
4  if v has children u_1,…, u_k for k ≥ 1, then
5    set c(v) = 0
6    for i = 1 to k do
7      if (u_i, v) ∉ Blk, then
8        set d_i(v) = d(u_i) + α · l(u_i, v)
9      if (u_i, v) ∈ Blk, then
10       if u_i ∉ Blk, then    /* enter a blockage */
11         set c(u_i) = C_b
12       set d_i(v) = d(u_i) + R_w · l(u_i, v) · (C_w · l(u_i, v) / 2 + c(u_i))
13     set c(v) = c(v) + C_w · l(u_i, v) + c(u_i)
14   set d(v) = max_{1≤i≤k} {d_i(v)}
15   if v ∉ Blk and there exists u_i s.t. (u_i, v) ∈ Blk, then
16     set d(v) = d(v) + R_b · c(v) + D_b /* exit a blockage */
17     set c(v) = C_b
18 if v is the source, then
19   if v ∉ Blk, then
20     set d(v) = d(v) - D_b
21   return d(v)
```

Figure 3: *DBDE* Algorithm.

B. *Statistical Buffered Delay Estimation*

In order to incorporate the effect of process variations into the *DBDE* algorithm, we express the characteristics of wire and interconnect as random variables in the first-order canonical form, i.e.,

$$R_w = R_{w0} + \gamma_w^T X \quad , \quad C_w = C_{w0} + \varepsilon_w^T X \quad , \quad R_b = R_{b0} + \gamma_b^T X \quad ,$$
$$C_b = C_{b0} + \varepsilon_b^T X \text{, and } \quad D_b = D_{b0} + \eta_b^T X \text{ .}$$

We apply these random variables to the major operations in *DBDE*, and thus the delay and capacitance variables, $d(v)$ and $c(v)$, propagated to each node v also become random variables. However, due to the non-linear operations (multiplication, maximum, and square-root operations) involved in the calculation, newly computed $d(v)$ and $c(v)$ are no longer in the first-order canonical forms. In order to keep $d(v)$ and $c(v)$ in the first order canonical forms without loss of much accuracy and make the computation more efficiently, we use some approximation techniques in the following computation.

In the bottom-up tree traversal, we represent the $d(u)$ and $c(u)$ of the child node u of node v by the following first-order canonical forms:

$$d(u) = d_{u0} + \alpha_u^T X \quad (4)$$

$$c(u) = c_{u0} + \beta_u^T X \quad (5)$$

If the edge (u, v) lies in a blockage, we use the same idea of [6] to modify the Elmore delay in line 12 in Figure 3 for process variations as follows:

$$\begin{aligned} d(v) &= d_{u0} + \frac{1}{2} l(u,v)^2 R_{w0} C_{w0} + l(u,v) R_{w0} c_{u0} \\ &\quad + \left[\alpha_u + \frac{1}{2} l(u,v)^2 (R_{w0}\varepsilon_w + C_{w0}\gamma_w) + l(u,v)(R_{w0}\beta_u + c_{u0}\gamma_w) \right]^T X \\ &\quad + X^T \left[\frac{1}{2} l(u,v)^2 \gamma_w \varepsilon_w^T + l(u,v)\gamma_w \beta_u^T \right] X \\ &= d_0 + \lambda^T X + X^T \Omega X \end{aligned} \tag{6}$$

where $d_0 = d_{u0} + \frac{1}{2} l(u,v)^2 R_{w0} C_{w0} + l(u,v) R_{w0} c_{u0}$

$$\lambda = \alpha_u + \frac{1}{2} l(u,v)^2 (R_{w0}\varepsilon_w + C_{w0}\gamma_w) + l(u,v)(R_{w0}\beta_u + c_{u0}\gamma_w)$$

$$\Omega = \frac{1}{2} l(u,v)^2 \gamma_w \varepsilon_w^T + l(u,v)\gamma_w \beta_u^T$$

Due to the quadratic term $X^T \Omega X$, $d(v)$ does not remain in the first-order canonical form as $d(u)$. However, we can still derive the first-order approximation of $d(v)$, which is denoted as $d(v)$', by applying the moment matching technique. First, we compute the first and second moments of $d(v)$ in (6) by using Property 1 , Property 2, and Property 3 in section II. We have

$$E(d(v)) = d_0 + \lambda^T \cdot E(X) + E(X^T \Omega X) = d_0 + tr(\Omega) \tag{7}$$

$$\begin{aligned} E(d(v)^2) &= d_0^2 + E(X^T \lambda \lambda^T X) + E\left((X^T \Omega X)^2 \right) + 2 d_0 \lambda^T E(X) \\ &\quad + 2E(X^T \Omega X \lambda^T X) + 2 d_0 E(X^T \Omega X) \\ &= \left(d_0^2 + tr(\Omega) \right)^2 + \lambda^T \lambda + 2tr(\Omega^2) \end{aligned} \tag{8}$$

Afterwards we compute the mean and variance of $d(v)$' by matching the first and second moments of $d(v)$. We have

$$\mu(d(v)') = E(d(v)) = d_0 + tr(\Omega) \tag{9}$$

$$\sigma(d(v)')^2 = E(d(v)^2) - E(d(v))^2 = \lambda^T \lambda + 2tr(\Omega^2) \tag{10}$$

Finally we successfully derive $d(v)$ to its first-order approximation $d(v)$' in the canonical form as follows:

$$d(v)' = \left(d_0 + tr(\Omega) \right) + \sqrt{1 + \frac{2tr(\Omega^2)}{\lambda^T \lambda}}\, \lambda^T X \tag{11}$$

By using the similar technique discussed above, we can apply the same approximation on all the random variables which are not in the first-order canonical forms. Almost all the operations listed in Figure 3 can be modified to consider process variations except the computation of linear delay, which is corresponding to line 8 in Figure 3. For the calculation of linear delay, we need to add a more advanced treatment in our approach for the presence of the square-root operation (which is involved in calculating α). The linear delay per unit wirelength, α, in statistical term can be expressed as follows:

$$\begin{aligned} \alpha &= R_{w0} C_{b0} + R_{b0} C_{w0} + \left(R_{w0}\varepsilon_b + C_{w0}\gamma_b + R_{b0}\varepsilon_w + C_{b0}\gamma_w \right)^T X \\ &\quad + X^T \left(\gamma_w \varepsilon_b^T + \gamma_b \varepsilon_w^T \right) X + f(X) \end{aligned} \tag{12}$$

where $f(X) = \sqrt{A + BX + CX^2 + DX^3 + EX^4}$

$$A = 2 R_{w0} C_{w0} \left(R_{b0} C_{b0} + D_{b0} \right)$$

$$B = 2 \left[R_{w0} C_{w0} R_{b0} C_{b0} \left(\frac{\gamma_w}{R_{w0}} + \frac{\varepsilon_w}{C_{w0}} + \frac{\gamma_b}{R_{b0}} + \frac{\varepsilon_b}{C_{b0}} \right) + R_{w0} C_{w0} D_{b0} \left(\frac{\gamma_w}{R_{w0}} + \frac{\varepsilon_w}{C_{w0}} + \frac{\eta_b}{D_{b0}} \right) \right]$$

$$C = 2 \left[\begin{aligned} &R_{w0} C_{w0} R_{b0} C_{b0} \left(\frac{\gamma_w \varepsilon_w}{R_{w0} C_{w0}} + \frac{\gamma_w \gamma_b}{R_{w0} R_{b0}} + \frac{\gamma_w \varepsilon_b}{R_{w0} C_{b0}} + \frac{\varepsilon_w \gamma_b}{C_{w0} R_{b0}} + \frac{\varepsilon_w \varepsilon_b}{C_{w0} C_{b0}} + \frac{\gamma_b \varepsilon_b}{R_{b0} C_{b0}} \right) \\ &+ R_{w0} C_{w0} D_{b0} \left(\frac{\gamma_w \varepsilon_w}{R_{w0} C_{w0}} + \frac{\gamma_w \eta_b}{R_{w0} D_{b0}} + \frac{\varepsilon_w \eta_b}{C_{w0} D_{b0}} \right) \end{aligned} \right]$$

$$D = 2 \left[\begin{aligned} &R_{w0} C_{w0} R_{b0} C_{b0} \left(\frac{\gamma_w \varepsilon_w \gamma_b}{R_{w0} C_{w0} R_{b0}} + \frac{\gamma_w \varepsilon_w \varepsilon_b}{R_{w0} C_{w0} C_{b0}} + \frac{\gamma_w \gamma_b \varepsilon_b}{R_{w0} R_{b0} C_{b0}} + \frac{\varepsilon_w \gamma_b \varepsilon_b}{C_{w0} R_{b0} C_{b0}} \right) \\ &+ \gamma_w \varepsilon_w \eta_b \end{aligned} \right]$$

$$E = 2 \left[\gamma_w \varepsilon_w \gamma_b \varepsilon_b \right]$$

We want a power series expansion of $f(x)$ as (13), so as to integrate the expansion with the remaining parts in α.

$$f(X) = a_0 + a_1 X + a_2 X^2 + \ldots \tag{13}$$

Here we apply the Taylor series expansion on $f(x)$ with respect to $X = 0$ and truncate it until the second order. We have

$$f(X) \approx \sqrt{A} + \left(\frac{B}{2\sqrt{A}} \right)^T X + X^T \left(\frac{C}{2\sqrt{A}} - \frac{B^2}{8\sqrt{A^3}} \right) X \tag{14}$$

The final step is to apply moment matching we discussed above on α and derive its corresponding first-order canonical form as (11) with the following parameters:

$$\begin{aligned} d_0 &= R_{w0} C_{b0} + R_{b0} C_{w0} + \sqrt{A} \\ \lambda &= R_{w0}\varepsilon_b + C_{w0}\gamma_b + R_{b0}\varepsilon_w + C_{b0}\gamma_w + \frac{B}{2\sqrt{A}} \\ \Omega &= \gamma_w \varepsilon_b^T + \gamma_b \varepsilon_w^T + \frac{C}{2\sqrt{A}} - \frac{B^2}{8\sqrt{A^3}} \end{aligned} \tag{15}$$

As for the maximum operation in line 14, we appeal to the concept of *tightness probability* in [7] or called the *binding probability* in [17, 18]. Given two random variables P and Q in the first-order canonical form as follows:

$$P = P_0 + \sigma_P^T X \tag{16}$$

$$Q = Q_0 + \sigma_Q^T X \tag{17}$$

The tightness probability $T_{P,Q}$, which represents the probability of P larger than Q, is computed by

$$\begin{aligned} T_{P,Q} &= \Phi\left(\frac{P_0 - Q_0}{\theta} \right) \\ \text{where } \theta &= \sqrt{\sigma_P^2 + \sigma_Q^2 - 2\rho\sigma_P\sigma_Q} \end{aligned} \tag{18}$$

Φ is the cumulative density function (CDF) of the standard normal distribution, and ρ is the correlation coefficient of P and Q. The values of operation Φ have been tabulated and can be found in [19]. After computing the mean and variance via the moment generating function provided in [20], the first-order canonical form of $max(P, Q)$ can be expressed as

$$\begin{aligned} \max(P,Q) &= T_{P,Q} P_0 + T_{Q,P} Q_0 + \theta \phi\left(\frac{P_0 - Q_0}{\theta} \right) \\ &\quad + \left(T_{P,Q}\sigma_P + T_{Q,P}\sigma_Q \right)^T X \end{aligned} \tag{19}$$

ϕ is the probability density function (PDF) of the standard normal distribution.

The closed formed of the operation in line 16 is as (11) with the following parameters:

$$\begin{aligned} d_0 &= d_{v0} + R_{b0} c_{v0} + D_{b0} \\ \lambda &= \alpha_v + R_{b0}\beta_v + c_{v0}\gamma_b + \eta_b \\ \Omega &= \gamma_b \beta_v^T \end{aligned} \tag{20}$$

Due to the page limit, we do not list the other operations in lines 3, 5, 11, 13, 17, and 20, in the paper because their closed forms are already first-ordered and can be derived straightforwardly. The complexity of our statistical buffered delay estimation technique is linear as the deterministic one shown in Figure 3.

IV. Experimental Results

We call our variation-aware estimation algorithm as *Statistical Buffered Delay Estimation* (*SBDE*) and call the deterministic algorithm modified from [2], as shown in Figure 3, as the *Deterministic Buffered Delay Estimation* (*DBDE*). We also implemented the variation-aware buffer insertion algorithm in [15], called *Statistical Buffer Insertion* (*SBI*), to verify the accuracy of our *SBDE* algorithm. All the algorithms were implemented in C++ language and compiled by g++ version 3.4.2 on a Linux x86_64 machine with 2G Processor/4GB RAM.

We ran the three different algorithms (*DBDE*, *SBDE*, *SBI*) on two sets of testcases reported in [2]. One set of testcases includes eight groups of randomly generated nets. We summarize their results in Table II in which each row reports the average values of each group which is grouped by the same number of sinks. The other is a set of industrial cases which are reported in Table III on individual basis. For all the testcases, we set the parameters of driver and receivers to be equal to those of the buffer by which we used in predicting buffered delay. In each case, we set each sink with equal criticality and then used the C-tree algorithm [21] to generate its Steiner topology. Moreover, we forced each of the random variables, i.e., R_w, C_w, R_b, C_b, D_b, has 5% deviation from its nominal value. A same blockage configuration was applied on each test case. We created 64 square blockages, each 4 mm on a side, and arranged them in a regular pattern on a square layout that has 40 mm on a side. Accordingly, there is sufficient space with 1-mm wide between blockages to allow buffer insertion.

Table II and Table III summarize the results of each algorithm. The columns "*#sink*", "*wirelength*" and "*%blk*" give the number of sinks, wirelength and the percentage of the net that was blocked. For each algorithm (*DBDE*, *SBDE*, *SBI*), the corresponding columns "*delay*", "*delay-sd*", "*#buf*", and "*runtime*" give the mean delay[3], the standard deviation of delay, the number of buffers inserted, and the CPU time, respectively. We report the comparison results of *SBDE* with *SBI* in the last three columns in which the "*delay*" and "*delay-sd*" of *SBI* are normalized as 100% and the "*runtime*" of *SBDE* is normalized as 1.

From Tables II and III, we can see that the delay estimated by *DBDE* is always smaller than the mean delay estimated by *SBDE*. Thus, we can say that in the presence of process variations, *SBDE* tightens the lower bound and gives a more accurate estimation than *DBDE* can do. By comparing *SBDE* with *SBI*, we observed the error of the mean delay value given by *SBDE* for the first set of testcases (for the second set of testcases, respectively) is only 4.31% (4.3%, respectively) on the average and the error of the standard deviation is within 5.62% (8.98%, respectively) on the average. We also observed that the accuracy of *SBDE* does not vary much with the size of a testcase. One can see the average error rates in Tables II and III are very close. In order to illustrate the error bound of *SBDE*, we pick two extreme testcases, mcu0 and n313, one of which commits the minimum error and the other commits the maximum one. Figure 4 depicts the delay distributions computed by *SBDE* and *SBI* for these two extreme testcases, as shown in Figures 4 (a) and (b), respectively. Although the distribution computed by *SBDE* in Figure 4 (b) does not perfectly match the one by *SBI*, *SBDE* is still more accurate than *DBDE* because the delay estimated by *DBDE* always locates at the left of the mean value computed by *SBDE*. Moreover, the runtime of *SBDE* can achieve 10~149 times faster than *SBI* while only 2.5~6 times slower than *DBDE*.

Our next experiment aims to observe whether running *DBDE* in the worst case corner (μ+3σ) is useful in predicting buffered delay under process variations. We choose 11 different timing yields (ranging from 99.99% to 90%) from the results reported by *SBI* to get different timing constraints. Note that the timing constraint gets more relaxed as the timing yield increases. In Figure 5, the X-axis gives the timing yield and the Y-axis shows the number of testcases passing the given timing constraint with the probability no less than the corresponding timing yield. Undoubtedly, all testcases estimated by *SBDE* can pass the given timing constraints. As for *DBDE* run in the worst case corner, denoted by *DBDE_worst*, the estimation becomes too pessimistic. For example, when we choose the 97% timing yield to set the timing constraint, none of the testcases estimated by *DBDE_worst* passes and thus all need a design rollback. It shows that *DBDE_worst* is over-pessimistic in predicting buffered delay with process variations.

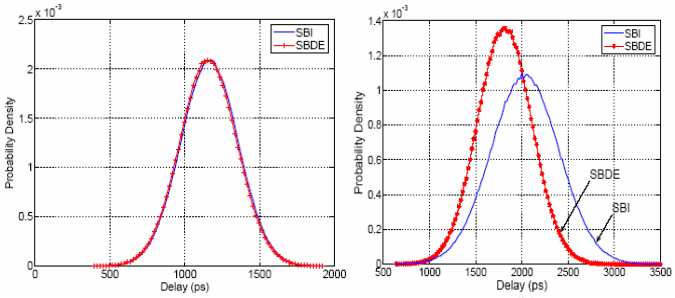

Figure 4: The delay distribution of *SBDE* and *SBI* (a) the case with minimum inaccuracy, (b) the case with maximum inaccuracy.

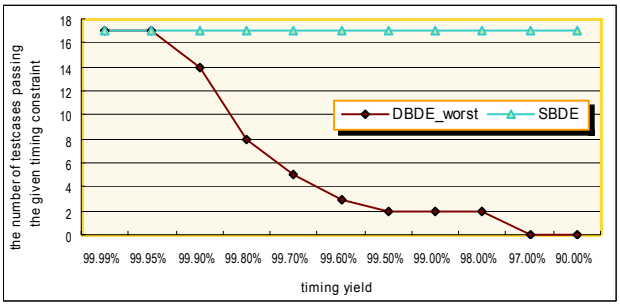

Figure 5: The over-pessimism of *DBDE*.

V. Conclusions

In this paper, we have derived the approximated first-order canonical forms for buffered delay estimation which considers the effect of process variations and the presence of buffer blockages. We empirically show that the deterministic buffered delay estimation using the worst case corner, i.e., μ+3σ, will be over-pessimistic. Promising experimental results are reported to demonstrate the efficiency and accuracy of our statistical estimation technique.

Before closing this paper, we would like to point out some possible future works. One possible future work is to adopt a more suitable matching or approximation technique for more accurate delay estimation. Another one is to embed the fast estimation technique into a floorplanning or routing algorithm to estimate the timing cost for a net under the consideration of process variations.

References

[1] P. Saxena, N. Menezes, P. Cocchini, and D. A. Kirkpatrick, "Repeater scaling and its impact on CAD," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 4, pp. 451-463, Apr. 2004.

[2] C. J. Alpert, J. Hu, S. S. Sapatnekar, and C. N. Sze, "Accurate estimation of global buffer delay within a floorplan," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1140-1146, Jun. 2006.

[3] The column "*delay*" in *DBDE* is the delay estimated by *DBDE* under nominal circuit parameters.

[3] R. H. J. M. Otten, "Global wires harmful?" in *Proc. Intl. Symp. on Physical Design*, pp. 104-109, 1998.

[4] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-Tree network for minimal Elmore delay," in *Proc. Intl. Symp. on Circuits and Systems*, pp. 865-868, 1990.

[5] C. Visweswariah, "Death, taxes and failing chips," in *Proc. Design Automation Conf.*, pp. 343-347, 2003.

[6] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 621-625, 2003.

[7] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "Frist-order incremental block-based statistical timing analysis," in *Proc. Design Automation Conf.*, pp. 331-336, 2004.

[8] M. R. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov, "Gate sizing using incremental parameterized statistical timing analysis," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 1029-1036, 2005.

[9] M. Mani, A. Devgan, and M. Orshansky, "An efficient algorithm for statistical minimization of total power under timing yield constraints," in *Proc. Design Automation Conf.*, pp. 309-314, 2005.

[10] V. Khandelwal, A. Davoodi, A. Nanarati, and A. Srivastava, "A probabilistic approach to buffer insertion," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 560-567, 2003.

[11] L. He, A. B. Kahng, K. Tam, and J. Xiong, "Simultaneous buffer insertion and wire sizing considering systematic CMP variation and random Leff variation," in *Proc. Intl. Symp. on Physical Design*, pp. 78-85, 2005.

[12] A. Davoodi, and A. Srivastava, "Variability-driven buffer insertion considering correlations," in *Proc. Intl. Conf. on Computer Design*, pp. 425-430, 2005.

[13] L. Deng, and M. D. F. Wong, "Buffer insertion under process variations for delay minimization," in *Proc. Intl. Conf. on Computer-Aided Design*, pp. 317-321, 2005.

[14] J. Xiung, K. Tam, and L. He, "Buffer insertion considering process variation," in *Proc. Design, Automation and Test in Europe*, pp. 970-975, 2005.

[15] J. Xiung and L. He, "Fast buffer insertion considering process variations," in *Proc. Intl. Symp. on Physical Design*, pp. 128-135, 2006.

[16] L. Zhang, W. Chen, Y. Hu, J. A. Gubner, and C. C.-P. Chen, "Correlation-preserved non-Gaussian statistical timing analysis with quadratic timing model," in *Proc. Intl. Conf. on Design Automation*, pp. 83-88, 2005.

[17] J. Jess, "DFM in synthesis," research report, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598, Dec. 2001.

[18] J. A. G. Jess, K. Kalafala, S. R. Naidu, R. H. J. M. Otten, and C. Visweswariah, "Statistical timing for parametric yield prediction of digital integrated circuits," in *Proc. Design Automation Conf.*, pp. 932-937, 2003.

[19] E. Kreyszig, *Advanced engineering mathematics*, 8th edition, Peter Janzow, 1999.

[20] M. Cain, "The moment-generating function of the minimum of bivariate normal random variables," *The American Statistician*, vol. 48, May 1994.

[21] C. J. Alpert, G. Gandham, M. Hrkic, J. Hu, A. B. Kahng, J. Lillis, B. Liu, S. T. Quay, S. S. Sapatnekar, and A. J. Sullivan, "Buffered Steiner trees for difficult instances," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 1, pp. 3-14, Jan. 2002.

TABLE II
Experimental Results of Randomly Generated Nets (on Average Basis)

Testcase				DBDE		SBDE			SBI				SBDE vs. SBI		
net	#sink	wirelength	%blk	delay	runtime	delay	delay-sd	runtime	delay	delay-sd	#buf	runtime	delay	delay-sd	runtime
r3	3	28849	55.50	1550.75	0.0009	1560.02	246.43	0.0024	1605.33	255.05	12	0.3644	97.18%	96.62%	149x
r4	4	36727	66.42	1771.43	0.0013	1782.11	282.17	0.0039	1837.98	292.77	16	0.4037	96.96%	96.38%	104x
r5	5	33210	72.53	1497.81	0.0013	1507.15	241.56	0.0039	1603.84	260.84	13	0.2901	93.97%	92.61%	75x
r6	6	39967	48.49	1627.52	0.0014	1637.19	258.88	0.0037	1667.90	265.55	20	0.4652	98.16%	97.49%	127x
r7	7	42645	62.72	1632.03	0.0016	1641.93	261.30	0.0051	1725.14	278.69	18	0.4014	95.18%	93.76%	79x
r8	8	51207	59.29	1946.99	0.0019	1958.55	310.58	0.0057	2040.27	328.48	23	0.4943	95.99%	94.55%	87x
r9	9	49267	62.93	1561.62	0.0020	1570.75	247.17	0.0061	1658.93	267.96	21	0.5266	94.68%	92.24%	86x
r10	10	57807	65.30	1745.86	0.0021	1756.23	278.64	0.0063	1880.92	304.91	25	0.5247	93.37%	91.38%	83x
													95.69%	**94.38%**	**99x**

TABLE III
Experimental Results of Industry Nets (on Individual Basis)

Testcase				DBDE		SBDE			SBI				SBDE vs. SBI		
net	#sink	wirelength	%blk	delay	runtime	delay	delay-sd	runtime	delay	delay-sd	#buf	runtime	delay	delay-sd	runtime
mcu0	18	39920	65.53	1149.70	0.0020	1157.39	190.32	0.0070	1165.96	190.98	21	0.7889	99.26%	99.65%	113x
mcu1	19	41380	78.13	1849.71	0.0020	1861.91	304.38	0.0080	1917.39	311.12	18	0.3329	97.11%	97.83%	42x
n107	17	12790	41.75	639.99	0.0010	643.36	102.66	0.0060	720.07	131.80	13	0.5789	89.35%	77.89%	96x
n189	29	58100	59.86	1727.20	0.0030	1738.50	285.20	0.0110	1768.46	316.06	30	0.5858	98.31%	90.24%	53x
n313	19	55840	82.65	1801.14	0.0030	1812.44	293.51	0.0080	2034.04	367.97	22	0.4369	89.11%	79.77%	55x
n786	32	54520	83.64	3842.54	0.0050	3867.33	628.67	0.0140	4082.36	677.60	20	0.3050	94.73%	92.78%	22x
n869	21	43270	81.42	3137.13	0.0030	3156.68	506.37	0.0090	3271.70	528.52	15	0.2440	96.48%	95.81%	27x
n873	20	49720	37.05	1418.88	0.0020	1427.59	227.48	0.0080	1455.53	262.77	25	0.4456	98.08%	86.57%	56x
poi3	20	63960	46.75	3516.91	0.0030	3537.35	554.27	0.0090	3578.88	561.88	39	0.0905	98.84%	98.64%	10x
													95.70%	**91.02%**	**53x**