Linear Complexity Pruning Rule Enables Efficient Buffer Insertion Considering Process Variations [*]

Jinjun Xiong Lei He
EE Department, University of California at Los Angeles
Los Angeles, CA, 90095

ABSTRACT

Advanced process technologies call for a proactive consideration of process variation to ensure effective timing yield and keep the manufacturing cost down. Buffer insertion critical for almost any high performance IC designs nowadays, however, has not gained enough attention in attacking the correlated process variations because of the lack of an efficient *pruning rule* that defines the dominance relationship between two randomly but interdependent solutions. The major contribution of this work is an efficient two-parameter (2P) based pruning rule for buffer insertion considering both inter-die and intra-die spatially correlated process variations. Under the normality assumptions on distribution, we theoretically prove that the 2P-based pruning rule has linear complexity (both time and space). Experiment results confirm the linear scalability of such an algorithm, and the capacity of the algorithm is also increased by more than thousands of times compared to the state-of-the-art technique. We further apply the algorithm for timing optimization considering correlated process variation. We show that buffer insertion without considering spatial correlated variation reduces the timing yield by more than 50%, thus alarm the importance of developing efficient algorithms for IC designs to attack the process variation effects actively.

1. INTRODUCTION

Advanced process technologies impose significant challenges for modern IC designers as we move into the ultra-deep submicron era where manufactured circuits exhibit substantial performance variations. The proactive consideration of process variation during the design stage is critical to ensure effective timing yield and keep the manufacturing cost down.

Current studies on process variations have been mainly focused on the variability modeling and statistical timing analysis (STA) [1, 2, 3]. There are very few papers that actively consider the process variation effects for design optimization. Critical for almost any high performance IC designs nowadays, the buffer insertion problem has been studied extensively with different objectives in literature [4, 5]. However, none of them has considered the effect of process variations yet. Moreover, almost all of these algorithms to some degree follow the same dynamic programming paradigm in solving the buffer insertion problem. This is mainly due to an important result owning to [4] that the dynamic programming based buffer insertion problem can be solved optimally in polynomial complexity by properly defining the *dominance relationship* (or *pruning rule*) between solutions.

However, when process variation is considered, solving the same buffer insertion problem "optimally" becomes tricky. One of the difficulties lies in the fact that solutions are no longer constant values, but random variables. Moreover, these solutions are not independent but *interdependent* with correlations. Different from the correlation discussed in the STA community, such correlation is not due to path reconvergence, but mainly due to the way we compute the solutions. In dynamic programming based buffer insertion, solutions are computed recursively from downstream nodes, rendering solutions from the same subtrees are correlated in nature. When intra-die spatial variation is considered, solutions from different subtrees will also exhibit certain correlations, which further complicates the problem. Another difficulty is how to define the *pruning rule* between different solutions in the presence of process variations. Because without a properly defined pruning rule, a straight-forward implementation of dynamic programming would make the algorithm complexity to increase exponentially.

To the best of our knowledge, there are three pieces of recent work in literature [6, 7, 8] that have attempted to solve the buffer insertion problem with consideration of different process variations. However, none of them has addressed the above two difficulties with definite answers. [6] studied the buffer insertion problem considering only the effect of wire length variation, albeit the fact that wire length variation is not a typical process variation. It was assumed that there was no correlation between different solutions. Moreover, three heuristic pruning rules were proposed with none of them can bound the complexity of the algorithm. The largest benchmark has 1260 sinks and no runtime was reported. [7] proposed to capture the correlation between solutions by using the joint probability density function (JPDF), and compute the JPDF recursively. A *four-parameter* based pruning rule was proposed to reduce the runtime complexity. However, the complexity and the effectiveness of the four-parameter based pruning rule is not clear, as only small benchmarks with no more than 9 sinks were reported. [8] studied a similar buffer insertion problem with a one-parameter based pruning rule, which is a simplified version of the four-parameter based pruning rule. Moreover, both [7] and [8] only considered the random device variations, but not the spatial correlated variation.

The contribution of this work is as follows. We propose an efficient two-parameter (2P) based pruning rule in the context of buffer insertion. Under the normality assumptions,

[*]Submission under review, please do not distribute.

we theoretically prove that the 2P-based pruning rule has linear complexity (both time and space), which enables efficient implementation of buffer insertion algorithm with consideration of both inter-die and intra-die spatial correlated variations. When solutions are not normally distributed, we give a theoretical necessary condition and experimentally verify that the two-parameter based pruning rule still has linear complexity in practice. Equipped with the efficient pruning rules, we employ a first-order process variation model to solve the buffer insertion problem considering both random device variation, inter-die variation and intra-die spatial correlated variations. Experiment results show that the algorithm based upon the 2P pruning rule can increase the buffer insertion capacity to more than sixty thousands of sinks, which is more than 60× improvement over [6] and 1000× improvement over [7]. Moreover, our experiment results also show that buffer insertion without considering spatial correlated variation reduces the timing yield by more than 50%, thus alarm the importance of developing efficient algorithms for IC designs to attack the process variation effects.

The rest of this paper is organized as follows. In Section 2, we propose a two-parameter based pruning rule with consideration of process variation. In Section 3, we present a first-order process variation model to consider different types of correlated variations. In Section 4, we discuss the buffer insertion algorithm with correlated process variations. We report experiment results in Section 5, and conclude in Section 6. Proofs of Theorems will be included in a technical report.

2. VARIATION AWARE PRUNING

2.1 Review of Deterministic Pruning

For a given buffered routing tree, two figure-of-merits are associated with every legal buffer position t in the tree: i.e., the *input loading capacitance* (or *downstream loading capacitance*) L_t and the *required arrival time* T_t. The basic buffer insertion problem formulation is to find the locations of buffers in the given routing tree such that the required arrival time (RAT) at the root is maximized. In the context of dynamic programming, we first traverse the routing tree in the reverse topological order and propagate solutions bottom up while book-keeping all intermediate solutions. At the end (root), we pick the optimal solution with the largest RAT. By backtracking the chosen optimal solution, we can determine the optimal solutions for each sub-tree recursively.

The complexity of dynamic programming based buffer insertion is mainly resulted from the merging of two sets of solutions obtained from two different sub-trees. In general, the total number of possible combinations for merging is $n \times m$, where n and m are the number of solutions from two sub-trees, respectively. If all combinations are kept at each merging node, the complexity will increase exponentially. However, by properly defining the *dominance relationship* (or *pruning rule*) between two solutions, i.e., solution (L_1, T_1) dominates solution (L_2, T_2) if condition $L_1 < L_2$ and $T_1 > T_2$ are satisfied, [4] proved that only $n + m$ number of solutions need to be kept instead of $n \times m$.

This is made possible by recognizing that there exists a *strict ordering* between solutions. That is, if $T_1 > T_2$ and $T_2 > T_3$, then $T_1 > T_3$. Similarly, if $L_1 < L_2$ and $L_2 < L_3$, then $L_1 < L_3$. Given there is N solutions, by pre-sorting

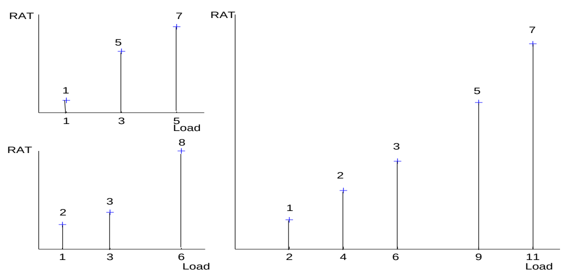

Figure 1: Linear merging $O(n + m)$.

the N solutions according to either L or T, we can prune out all dominated solutions in linear time $O(N)$, instead of quadratic time $O(N^2)$.

Even though pruning based upon the dominance relation can help reduce the total number of solution, in general we still have to pay the price of $O(n \times m)$ locally in order to obtain all possible combinations for merging. In deterministic buffer insertion, such a procedure can be reduced to $O(n + m)$ by using a merging sort like operation on the two sets of already sorted solutions. To see this, we give an example in Figure 1, where there are three solutions from each sub-tree, i.e., $n = 3$ and $m = 3$, and they are strictly sorted in terms of both L and T. By using a merging sort like operation on both solution sets while following the dominance rule, we can obtained all non-dominated solutions $O(m+n)$ after merging in linear time. Moreover, the solutions after merging is also strictly sorted as shown in Figure 1.

Based upon the above two linear operations on pruning and merging, [4] proved that by keeping only dominating solutions at every node, the dynamic programming based algorithm can solve the buffer insertion problem in polynomial time without loosing optimality[1].

2.2 Review of Four-parameter Based Pruning

Even though the conventional deterministic buffer insertion problem can be solved optimally via dynamic programming [4], solving its variation-aware counterpart "optimally" is not that easy. The difficulty lies in the fact that L_t and T_t are no longer constant values, but two interdependent random variables, as both are recursive functions of their downstream random variations. Moreover, different solutions, (L_1, T_1) and (L_2, T_2), no matter whether or not they come from the same subtree, can be also interdependent in the presence of spatial correlated variations.

By relating the dominance relation with designers' willingness of accepting uncertainty for a given design, [7] proposed the following *four-parameter* based (4P) dominance rule. Each parameter π_α gives a measure of designers preference for certainty in choosing the design parameter x in the presence of variations, such that the final design would have x less than π_α with $(100\alpha)\%$ certainty. Mathematically, this is given by

$$\alpha = \int_{-\infty}^{\pi_\alpha} f(x)dx, \qquad (1)$$

where $f(x)$ is the probability density function (PDF) of x.

[1]Note that the recently proposed predicative pruning [9] also exploits the same strictly ordering property of deterministic solutions.

Given four parameters, π_{α_l} and π_{α_u} for L_t, and π_{β_l} and π_{β_u} for T_t, such that $0 \leq \alpha_l < \alpha_u \leq 1$ and $0 \leq \beta_l < \beta_u \leq 1$, solution (L_1, T_1) is said to dominate solution (L_2, T_2) if the following conditions are satisfied:

$$\pi^{(1)}_{\alpha_u} < \pi^{(2)}_{\alpha_l} \tag{2}$$
$$\pi^{(1)}_{\beta_l} > \pi^{(2)}_{\beta_u}. \tag{3}$$

Despite of its intuitive definition, such a dominance relation is hard to use in practice for large designs, because under such a dominance relation, it is very difficult to keep the strict ordering of solutions as the deterministic case does, thus loosing both the linear complexity of merging and pruning operations. In other words, we still have to pay the price of $O(n\times m)$ in computing all possible combinations for merging solutions, and $O(N^2)$ for pruning, which makes it less efficient for large designs as we will show in the experiment parts.

2.3 Two-parameter Based Pruning

In the following, we propose a new variation aware pruning rule that would enable us to keep both merging and pruning operations in linear complexity even in the presence of process variations.

The new pruning rule is based upon the following observation. To eliminate uncertainty when comparing two solutions, we can extend the deterministic dominance relation between (L_1, T_1) and (L_2, T_2) by enforcing $P(L_1 \leq L_2) = 1$ and $P(T_1 \geq T_2) = 1$. The physical interpretation of this extension is that solution (L_1, T_1) has 100% chances (almost always) to result in a larger required arrival time but with a less loading capacitance when compared to solution (L_2, T_2). However, in practice, such a 100% chance requirement is too conservative. Therefore, we relax such a requirement by adding two parameters such that Solution (L_1, T_1) is said to dominate solution (L_2, T_2) if the following two conditions hold:

$$P(L_1 < L_2) > \overline{p_L}, \tag{4}$$
$$P(T_1 > T_2) > \overline{p_T}, \tag{5}$$

where $\overline{p_L}$ and $\overline{p_T}$ are two parameters between 0.5 and 1. In other words, the probability of L_1 being less than L_2 is greater than $\overline{p_L}$, while the probability of T_1 being greater than T_2 is less than $\overline{p_T}$. We call the pruning rule as defined by (4) and (5) as *two-parameter* (2P) based pruning rule in the following.

By utilizing the new dominance rule as defined in (4) and (5), we can order the solutions much the same way as in the deterministic cases, thus achieving similar complexity as the deterministic counterpart. To see this argument, we have the following Theorems[2].

THEOREM 1. *Given T_1, T_2 and T_3 as three* **independent** *random variables with arbitrary distributions, if $P(T_1 > T_2) > p$, $P(T_2 > T_3) > p$, then $P(T_1 > T_3) > p$.*

THEOREM 2. *Given T_1, T_2 and T_3 as three* **interdependent** *random variables with arbitrary distributions, if $P(T_1 > T_2) > p$, $P(T_2 > T_3) > p$, then $p > 0.5$ is the necessary condition for $P(T_1 > T_3) > p$ to hold.*

[2]For simplicity, we only use T to illustrate the idea. It is understood that same results can be applied to L as well.

THEOREM 3. *Given T_1, T_2 and T_3 as three* **interdependent** *random variables with normal distributions, if $P(T_1 > T_2) > 0.5$, $P(T_2 > T_3) > 0.5$, then $P(T_1 > T_3) > 0.5$*

THEOREM 4. *Given T_1 and T_2 as two* **interdependent** *random variables with normal distributions, for $P(T_1 > T_2) > 0.5$ to hold, it is necessary and sufficient to have $\mu_{T_1} > \mu_{T_2}$ with μ_{T_1} and μ_{T_2} being the mean for T_1 and T_2, respectively.*

Theorem 1 says that if the inter-dependency between solutions are ignored, we can have a strict ordering between solutions in terms of (4) and (5), If the inter-dependency between solutions cannot be ignored, setting $\overline{p_L} = 0.5$ is the necessary condition to keep the strict ordering according to Theorem 2. If normal distribution is observed for each solution, setting $\overline{p_L} = 0.5$ is also the sufficient condition to keep the strict ordering according to Theorem 3. Moreover, according to Theorem 4, the ordering of solutions can be purely determined by mean values. By keeping the strict ordering between random solutions, we can simplify the $O(n \times m)$ merging operations into $O(n + m)$ while doing pruning in $O(n)$ based upon the similar arguments as in Section 2.1. Such a statement is also experimentally verified in our experiments, and we can achieve more than 20× runtime speedup compared to the 4P-based pruning.

When $\overline{p_L}$ and $\overline{p_T}$ are greater than 0.5, we cannot prove theoretically that the above strict ordering still holds. However, under the normal distribution assumption, we will show that the above dominance rule still gives a reasonable good ordering of solutions. To see this, we assume both T_1 and T_2 (similarly L_1 and L_2) are normal, and we have the following closed forms to evaluate the dominance relation between them according to [10]:

$$P(T_1 > T_2) = \Phi(\frac{\mu_{T_1} - \mu_{T_2}}{\sigma_{T_1,T_2}}) > \overline{p_T}, \tag{6}$$

where Φ is the cumulative density function (CDF) of standard normal distribution, μ_{T_1}, and μ_{T_2} are the mean values of T_1 and T_2, respectively. σ_{T_1,T_2} can be computed by

$$\sigma_{T_1,T_2} = (\sigma^2_{T_1} - 2 \cdot \rho_{T_1,T_2}\sigma_{T_1}\sigma_{T_2} + \sigma^2_{T_2})^{1/2}, \tag{7}$$

where $\sigma^2_{T_1}$ and $\sigma^2_{T_2}$ are variance of T_1 and T_2, respectively; and ρ_{T_1,T_2} is the correlation coefficient of T_1 and T_2.

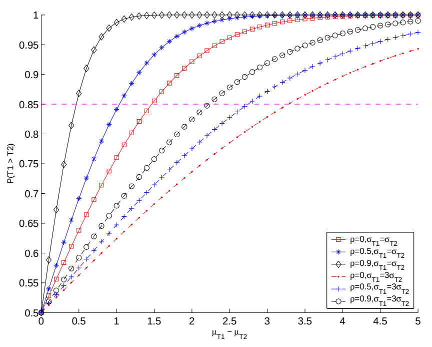

Figure 2: Probability of T_1 is greater than T_2.

We plot the probability of T_1 greater than T_2 in Figure 2, where the x-axis is the mean difference of T_1 and T_2 (μ_{T_1}-μ_{T_2}), and the y-axis is $P(T_1 > T_2)$. Three correlation coefficients, $\rho = 0$, $\rho = 0.5$ and $\rho = 0.9$, between T_1 and T_2

are plotted. The first three plots have $\sigma_{T_1}=\sigma_{T_2}$, and the rest three plots have $\sigma_{T_1}=3\sigma_{T_2}$. According to Figure 2, we can see that when the difference between μ_{T_1} and μ_{T_2} becomes larger, the probability of T_1 being greater than T_2 is also increasingly larger. For a given required probability, say $\overline{p_L}$=0.85, it only requires μ_{T_1} being greater than μ_{T_2} by less than $4ps$. In practice, for a general routing tree, such a small delay difference will likely exist among different solutions, either due to the difference in routing or due to the difference in buffering. Therefore, we conclude that ordering by T_i's mean value will not loose much accuracy even when different $\overline{p_L}$ values are taken. Moreover, such an approximation becomes even better when two solutions have similar variance and have higher correlations, which are the cases for our buffer insertion problem because solutions from the same sub-tree or nearby sub-trees are highly correlated in nature. Such a statement has also been verified experimentally in our experiment parts (Section 5).

3. PROCESS VARIATIONS MODELING

As has been shown in the STA community that block-based STA analysis is suitable for increment computation of statistical timing. In the course of buffer insertion, such an increment computation feature is a must. Therefore, in the following, we discuss a first-order process variation model that incorporates both random device variation, inter-die variation and intra-die spatial correlated variations. Such a first-order modeling has been used for STA [3, 1], but none of them has explicitly considered all the above three type of variations. Moreover, their focus is mainly for timing analysis, while our focus in this paper is on design optimization.

3.1 Random Variation

We characterize a device (or buffer) in terms of its gate capacitance (C_b), intrinsic delay (T_b) and output resistance (R_b). Due to process variations, these values will no longer be fixed values. To simplify the model, we lump all variation effects into C_b and T_b while keeping R_b as a constant for a given device size. In general, devices characteristics are complicated (nonlinear) functions of the underlying physical parameters and sometimes are even hard to described in a closed form. Therefore, we resort to first-order approximation. The rational is that if the underlying parametric variations is small, any nonlinear relationship can be reasonably captured by a first-order approximation. Mathematically it can be described as:

$$C_b = C_{b0} + \sum \alpha_i \cdot X_i, \qquad (8)$$

$$T_b = T_{b0} + \sum \beta_i \cdot X_i, \qquad (9)$$

where C_{b0} and T_{b0} are nominal values of C_b and T_b, respectively; and X_i are the underlying parametric variations such as channel length, doping density, and gate oxide thickness. The coefficients α_i and β_i are sensitivity of C_b and T_b to the variation of X_i, respectively.

We run SPICE simulations to verify the accuracy of the above first-order modeling. For illustration purpose and also because of the lack of access to the real sources of foundry process variations, we only model the random L_{eff} variation using $65nm$ BSIM model in this section. We assume the variation of L_{eff} to be a symmetric normal distribution with the standard deviation as 10% of its mean value in our experiments. After extracting devices characteristics from SPICE simulation, we then use a least square curve-fitting technique to obtain (8) and (9).

Because of the nonlinear relationship between parametric variations (like channel length, doping density, gate oxide thickness, etc.) and the device characteristics, the latter's distributions are unlikely to be normal even if the underlying parametric variations are assumed to be normal. However, just as we have discussed above, if the underlying parametric variations is assumed to be small, the nonlinear relationship can be approximated by a first-order linear equation. Therefore, the device characteristics can also be approximated by a normal distribution. We validate this argument via Monte Carlo simulation. Figure 3 shows both the SPICE extracted PDFs of T_b and the normal distribution approximation from (9). It clearly shows that normal distribution is a reasonably good approximation of the real distribution because the two PDFs are very close to each other.

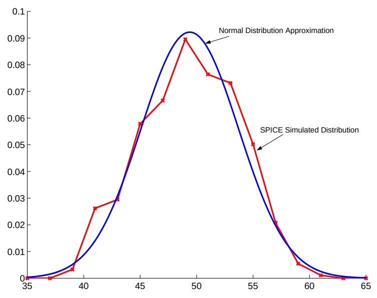

Figure 3: Normal distribution approximation of T_b.

3.2 Intra-die Variation

Current wafer manufacturing methods employ multiple identically oriented chips (usually four) within multi chip reticles during the stepping imaging process. Because of the optical lens distortion and wafer manufacturing processes, characteristics of devices on the same die are generally different and may take on the form of radial distortion [11], i.e., differences depending on distance from the center of the lens. Moreover, devices that are physically close to each other have higher correlation than devices that are far apart.

Therefore, to properly predict device characteristics, the following first-order model is proposed to capture the intra-die spatial variations on device characteristics. We partition the die area into different regions, and associate each region with one independent random variables Y_i. When the variance of Y_i, σ_{Y_i}, are the same for all regions, we call such a model as *homogeneous* spatial variation model. When the variance of Y_i are different for different regions (and may even exhibit certain patterns), we call such a model as *heterogeneous* spatial variation model. Note that the spatial model proposed here is different from that of [12] in the sense that we use less number of independent variables but with more flexibility in capturing different spatial variations. We further note that associating each region with an independent random variable is also different from the principle component analysis (PCA) technique as employed in [1].

For a device located at a particular region R_t, we have

$$C_{b,t} = C_{b0} + \sum \alpha_i \cdot X_i + \sum_{i \in \mathcal{I}_t} \gamma_i \cdot Y_i, \quad (10)$$

$$T_{b,t} = T_{b0} + \sum \beta_i \cdot X_i + \sum_{i \in \mathcal{I}_t} \theta_i \cdot Y_i. \quad (11)$$

The index set $\mathcal{I}_t$ defines the set of regions that spatial correlations matter for devices located at R_t, and the coefficient γ_i and θ_i further determine how strong the correlations are. In general, the larger the coefficients, the larger the correlation. Because the index set $\mathcal{I}_t$, and the coefficients γ_i and θ_i are region-dependent and different regions will have different $\mathcal{I}_t$, γ_i, and θ_i, by properly setting up these values, we can capture the spatial correlations between devices at different regions. For example, for two devices located at two nearby regions, they will share more number of common regions as decided by their common indexes in $\mathcal{I}_{t1}$ and $\mathcal{I}_{t2}$, and thus would have larger corresponding correlations.

For example, Figure 4 shows a layout with regular defined regions. If we assume that each region is correlated with its closest neighboring regions (3×3 grids) with each region having an independent random variables Y_i, then the device's characteristics will be affected by Y_1 to Y_9 as given by (10) and (11). For two buffers that are physically closer to each other (e.g., buffer $B1$ and buffer $B2$), they share two number of correlated regions as defined by their respective correlation regions. On the other hand, for two buffers that are physically apart from each other (e.g., buffer $B1$ and buffer $B5$), they do not share any number of correlated regions, hence their spatial correlation is negligible.

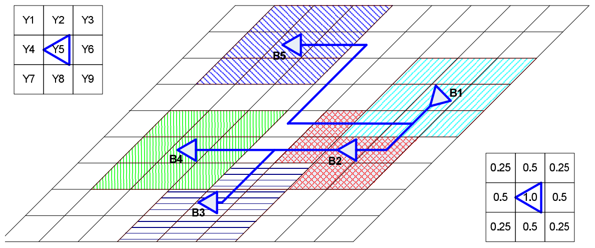

Figure 4: Modeling of the spatial correlations.

3.3 Inter-die Variation

As the inter-die variation affects all devices within the same die uniformly, we can model this variation by introducing another independent random variable, G and modify (10) and (11) as follows:

$$C_{b,t} = C_{b0} + \sum \alpha_i \cdot X_i + \sum_{i \in \mathcal{I}_t} \gamma_i \cdot Y_i + \xi \cdot G, \quad (12)$$

$$T_{b,t} = T_{b0} + \sum \beta_i \cdot X_i + \sum_{i \in \mathcal{I}_t} \theta_i \cdot Y_i + \eta \cdot G. \quad (13)$$

4. BUFFER INSERTION CONSIDERING PROCESS VARIATIONS

4.1 Key Operations for Buffer Insertion

In addition to the dominance rule between solutions, three other key operations are needed in order to solve the dynamic programming based buffer insertion problem. We review them briefly as follows. Denote c and r as interconnect's *unit length capacitance* and *sheet resistance*, respectively, we model each interconnect segment in the routing tree with length l as a π model. Under the Elmore delay model, the L_t and T_t can be computed recursively if we know the downstream node solutions.

If the solution at node t is obtained by adding a wire of length l at its direct downstream node n, then

$$L_t = L_n + c \cdot l \quad (14)$$

$$T_t = T_n - r \cdot l \cdot L_n - \frac{1}{2} \cdot r \cdot c \cdot l^2. \quad (15)$$

If the solution at node t is obtained by adding a buffer at its direct downstream node n, then

$$L_t = C_b \quad (16)$$

$$T_t = T_n - T_b - R_b \cdot L_n. \quad (17)$$

If the solution at node t is obtained by merging two solutions from its two sub-trees rooted at nodes m and n, respectively, then

$$L_t = L_n + L_m \quad (18)$$

$$T_t = min(T_n, T_m). \quad (19)$$

4.2 Variation Aware Key Operations

In contrast to [7] where the process variation is modeled by the numerical JPDF, we employ the first order process variation model as discussed in Section 3 in this paper. We will report our comparison with [7] in Section 5. Below, we describe the changes that we need to make to the three key operations in solving the dynamic programming based buffer insertion problem.

When solutions at downstream nodes are modeled as random variables, the newly computed solutions (L_t, T_t) are also random variables. To make the recursive computation efficient, we keep the same first order form for L_t and T_t as their downstream nodes. In the following, we assume all downstream node solutions are known and can be represented by the following first order forms:

$$L_n = L_{n0} + \sum \alpha_{n,i} \cdot X_i \quad (20)$$

$$T_n = T_{n0} + \sum \beta_{n,i} \cdot X_i, \quad (21)$$

where X_i can be any kind of defined variations. We then compute the solution at the current node t as follows. If the solution at node t is obtained by adding a wire of length l at its direct downstream node n, then

$$L_t = (L_{n0} + c \cdot l) + \sum \alpha_{n,i} \cdot X_i, \quad (22)$$

$$T_t = (T_{n0} - r \cdot l \cdot L_{n0} - \frac{1}{2} \cdot r \cdot c \cdot l^2) + \sum (\beta_{n,i} - r \cdot l \cdot \alpha_{n,i}) \cdot X_i. \quad (23)$$

If the solution at node t is obtained by adding a buffer at its direct downstream node n, then

$$L_t = C_{b0} + \sum \alpha_{b,i} \cdot X_i, \quad (24)$$

$$T_t = (T_{n0} - T_{b0} - R_b \cdot L_{n,0}) + \sum (\beta_{n,i} - \beta_{b,i} - R_b \cdot \alpha_{n,i}) \cdot X_i. \quad (25)$$

If the solution at node t is obtained by merging two solutions from its two sub-trees rooted at nodes m and n, respectively, then

$$L_t = (L_{n0} + L_{m0}) + \sum(\alpha_{n,i} + \alpha_{m,i}) \cdot X_i. \qquad (26)$$

To express T_t after the *min* operation still to be a linear combination of underlying variations, we resort to the tightness probability idea from [3], and we have

$$\begin{aligned} T_t &= t_{n,m} \cdot T_{n0} + (1 - t_{n,m})T_{m0} - \sigma_{n,m} \cdot \phi(\frac{\mu_{T_m} - \mu_{T_n}}{\sigma_{n,m}}) \\ &+ \sum(t_{n,m} \cdot \beta_{n,i} + (1 - t_{n,m}) \cdot \beta_{m,i}) \cdot X_i, \qquad (27) \end{aligned}$$

where $t_{n,m}$ is the probability of T_n less than T_m, and can be computed by

$$t_{n,m} = \Phi(\frac{\mu_{T_m} - \mu_{T_n}}{\sigma_{n,m}}). \qquad (28)$$

ϕ and Φ are the PDF and CDF of the standard normal distribution, respectively; and $\sigma_{n,m}$ can be computed by

$$\sigma_{n,m} = (\sigma_n^2 - 2 \cdot \rho_{n,m} \cdot \sigma_n \cdot \sigma_m + \sigma_m^2)^{1/2}, \qquad (29)$$

where σ_n^2 and σ_m^2 are variance of T_n and T_m, respectively; and $\rho_{n,m}$ is the correlation coefficient of T_n and T_m. Knowing the first-order representation of T_n and T_m, we can compute σ_n^2, σ_m^2, and $\rho_{n,m}$ as follows.

$$\begin{aligned} \sigma_n^2 &= \sum \beta_{n,i}^2 \sigma_{X_i}^2, \qquad (30) \\ \sigma_m^2 &= \sum \beta_{m,i}^2 \sigma_{X_i}^2, \qquad (31) \\ \rho_{n,m} &= \frac{\sum \beta_{n,i}\beta_{m,i}\sigma_{X_i}^2}{\sigma_n \sigma_m}, \qquad (32) \end{aligned}$$

where $\sigma_{X_i}^2$ is the variance of X_i.

5. EXPERIMENT RESULTS

5.1 Experiment Setting

Two sets of benchmarks are obtained from the public domain for our experiments [9]. The characteristics of the benchmarks are shown in Table 1.

Bench	Sinks	Buffer Positions
p1	269	537
p2	603	1205
r1	267	533
r2	598	1195
r3	862	1723
r4	1903	3805
r5	3101	6201

Table 1: Characteristics of benchmarks.

Because of the lack of access to the real wafer data, we derive the process variation data based upon the literature that addresses similar process variation issues but in the context of statistical timing analysis [12]. In our experiment, the 65nm BSIM technology is assumed. We budget the random device variation, inter-die variation, and intra-die variation all to be 5% of its nominal value, respectively. For the homogeneous spatial variation model, we uniformly distribute the budgeted 5% variation into different regions. For the heterogeneous spatial variation model, we distribute the budgeted 5% variation on the die from the South-West corner to the North-East corner in a linearly increasing fashion, i.e., devices located at the South-West corner has smaller spatial variation while devices located at the North-East corner has larger spatial variations. We divide the chip layout into different grids with the length of each grid as 500μm. For devices located at a particular grid, their characteristics are affected by a set of nearby grids whose contribution weights form an isotropic stationary Gaussian process with the value tapers off at a distance about 2mm.

5.2 Runtime Comparison

We compare the runtime between our algorithm and [7] based upon RAT optimization (Section 5.3) as follows. Because there is no detailed runtime data from [7], we first compare the two algorithms runtime complexity in terms of the largest benchmark that each algorithm can handle. As reported in [7], the largest routing tree has only nine (9) sinks. In contrast, our algorithm can easily handle routing trees with more than three thousand sinks (> 3000), and it seems that nothing prevents our algorithm from handling even larger benchmarks[3].

Bench	$4P$	$2P$	Speedup
p1	25.4	1.5	17.3×
p2	-	6.9	-
r1	-	25.0	-
r2	-	63.8	-
r3	-	131.5	-
r4	-	396.9	-
r5	-	922.8	-

Table 2: Runtime comparison in second.

Furthermore, we speculate that one of the main reasons in preventing [7] from trying larger benchmarks is that their 4P-based pruning rule is not very effective in both pruning and merging solutions. Another reason is due to the numerical complexity in computing JPDF, which requires the computation of high dimensional integral numerically. To verify this speculation, we have also reimplemented the algorithm of [7] based upon the same process variation models as used in this paper. In other words, we avoid [7]'s high complexity in computing the JPDF numerically, but rather to use the same first order process variation model to represent the JODI implicitly. Therefore, the only difference between our algorithm and that of [7] is the dominance rule. For the algorithm in [7], the 4P-based pruning rule as discussed in Section 2 is used, while our algorithm employs the newly proposed 2P-based pruning rule.

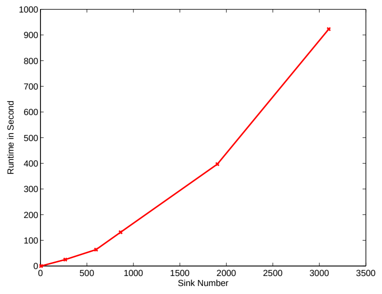

Figure 5: Runtime versus total number of sinks.

[3]In fact, the largest benchmark we have tested in house is an eight-level H-tree clock network with more than 64,000 sinks.

In Table 2, we report the runtime for both algorithms based upon the benchmarks we have tested. According to Table 2, we can see that the newly implemented algorithm of [7] now can handle much larger benchmarks than what was originally reported in [7], and the largest tested benchmark is $p1$ with 269 sinks. This improvement is mainly due to the avoidance of computing JPDF explicitly. However, we still fail to use the improved algorithm of [7] to run larger benchmarks. In fact, for the rest of tested benchmarks, it fails due to exceeding either memory capacity ($2G$) or tolerable time limit (4 hours in our setting). This observation is expected, because as we have explained in Section 2, the 4P-based pruning rule only imposes partially ordering between solutions, rendering the complexity of merging as $O(n \times m)$ and pruning as $O(N^2)$ for both memory and runtime. In contrast, by using the newly proposed 2P-based pruning rules, our algorithm can easily run through all benchmarks and for the largest benchmark $r5$, the runtime is less than 16 minutes. This significant runtime speedup is achieved because the 2P based pruning rule as discussed in Section 2 enforces a strict ordering between solutions, thus enables a linear complexity for both merging and pruning. This is further confirmed by Figure 5 which shows roughly the linear runtime scalability of our algorithms in terms of the number of sinks.

5.3 RAT Optimization

Enabled with the efficient implementation of buffer insertion considering both inter-die and intra-die variations, we can run our buffer insertion algorithm on the benchmarks for RAT optimization and study the effect of process variation on buffered interconnect design.

Before we report the experiment results, we first verify the accuracy of our model in predicting the RAT under process variations via the Monte Carlo simulation. Given a buffered routing tree with process variations, we run the algorithm to compute the PDF of RAT at the root. Such a procedure only requires the three key operations according to Section 4.2. After we obtain the RAT at the root represented in the form of (21), its PDF can be obtained by computing its mean and variance and it is approximated to be normally distributed. Figure 6 shows both the model predicted PDF and the Monte Carlo simulated PDF for the RAT at the root for one of the largest benchmarks ($r5$) in our experiments. We can clearly see that the first order process variation model is very accurate in predicating the PDF of RAT, and therefore it can indeed be employed for RAT optimization.

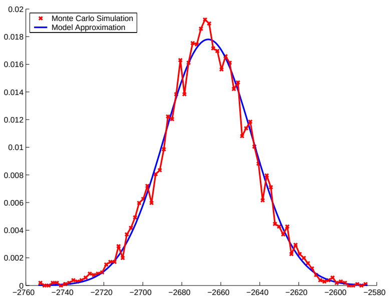

Figure 6: RAT at the root predicted by our model versus Monte Carlo simulation.

We compare three buffer insertion algorithms for RAT optimization in the following. The first one is the deterministic buffer insertion algorithm where all design variables are assumed to be nominal without variation, and we denote it as NOM. The second one is the variation-aware buffer insertion considering only random device variation and inter-die variation, but without considering the spatial variations. We denote such algorithm as $D2D$. The last one is the variation-aware buffer insertion algorithm considering all above variations, and it is denoted as WID. We run each algorithm on the given benchmarks and obtain the buffered routing tree. Two figure-of-merits are used to compare different algorithms. The first one is the 95% timing yield for RAT, which is defined as the 5%-tile RAT on the PDF such that the final RAT has the 95% chances of being larger than the 5%-tile RAT. The second one is the timing yield for a given required RAT at the root.

Table 3 compares the three algorithms using the heterogeneous spatial variation model as defined in Section 5.1. Column 2, 4 and 6 are the RAT at the 95% timing yield rate. The value in parenthesis are relative RAT degradation for NOM and $D2D$ when compared to WID, respectively. According to Table 3, we can see that deterministic buffer insertion (NOM) without considering process variations always results in designs that require much higher RAT than WID and the increase can be as high as 23.1%. On average, NOM increases RAT by 9.7% compared to WID. From Table 3 we can also see that variation-aware design without considering spatial variation ($D2D$) also increases the RAT compared to WID, and the average increase is about 8.4%. It is interesting to note that compared to NOM, $D2D$ achieves only about 2% improvement on average for RAT optimization. This confirms the common wisdom that global inter-die variation shifts the design space in the same direction while random device variation effect tends to diminish for a large design due to the variation canceling effects.

We also compare the timing yield for the three algorithms in Table 3. As we already observe that RAT from WID is about 10% better on average than both NOM and $D2D$, we set the mean RAT of WID with 10% reduction as the targeted RAT for all designs. We then compute the timing yield for all algorithms and report the results under Column 3, 5 and 7 in Table 3. According to Table 3, we see that NOM and $D2D$ on average only achieve 42.2% and 46.7% timing yield and the yield loss is more than 50% when compared to WID.

Bench	NOM	$D2D$	WID
p1	60 (1.09×)	60 (1.09×)	55
p2	156 (1.08×)	155 (1.08×)	144
r1	65 (1.44×)	61 (1.36×)	45
r2	135 (1.44×)	131 (1.39×)	94
r3	187 (1.42×)	187 (1.42×)	132
r4	375 (1.38×)	374 (1.38×)	272
r5	608 (1.32×)	598 (1.30×)	459
Avg	1.15×	1.13×	1×

Table 5: Number of buffers under different variation models.

We also run the same set of experiments but under the homogeneous spatial variation model as defined in Section 5.1 and report the experiment results in Table 4. Similar trends as in Table 3 can be observed. Moreover, by comparing Table 4 and Table 3, we observe that homogeneous

Bench	*NOM*		*D2D*		*WID*	
	RAT (%)	Yiled	RAT (%)	Yiled	RAT	Yiled
p1	-2673.5 (-2.4%)	99.6%	-2673.5 (-2.4%)	99.6%	-2611.7	100%
p2	-3791.3 (-7.7%)	99.9%	-3713.4 (-5.5%)	99.8%	-3519.3	100%
r1	-1240.7 (-15.9%)	0.5%	-1193.7 (-11.5%)	14.8%	-1070.3	100%
r2	-1808.1 (-23.1%)	0.1%	-1751.8 (-19.2%)	2.3%	-1469.2	100%
r3	-1658.8 (-9.3%)	48.5%	-1658.8 (-9.3%)	48.5%	-1518.0	100%
r4	-2475.7 (-10.7%)	27.6%	-2475.7 (-10.7%)	27.6%	-2236.0	100%
r5	-2934.9 (-8.6%)	83.5%	-2934.9 (-8.6%)	83.5%	-2703.3	100%
Avg	-9.7%	45.0%	-8.4%	47.0%	100%	100%

Table 3: RAT optimization under the *heterogeneous* spatial variation model.

Bench	*NOM*		*D2D*		*WID*	
	RAT (%)	Yiled	RAT (%)	Yiled	RAT	Yiled
p1	-2620.2 (-1.8%)	99.6%	-2620.2 (-1.8%)	99.6%	-2574.1	100%
p2	-3684.9 (-0.9%)	99.9%	-3685.9 (-0.9%)	99.8%	-3652.4	100%
r1	-1181.0 (-11.0%)	0.5%	-1141.0 (-7.3%)	14.8%	-1063.5	100%
r2	-1612.3 (-11.7%)	0.1%	-1575.9 (-9.2%)	2.3%	-1443.6	100%
r3	-1602.0 (-4.8%)	48.5%	-1602.0 (-4.8%)	48.5%	-1529.2	100%
r4	-2280.7 (-5.0%)	27.6%	-2280.7 (-5.0%)	27.6%	-2172.6	100%
r5	-2653.1 (-3.1%)	83.5%	-2653.1 (-3.1%)	83.5%	-2572.6	100%
Avg	-4.8%	45.0%	-4.0%	47.0%	100%	100%

Table 4: RAT optimization under the *homogeneous* spatial variation model.

spatial variation leads to in general a larger RAT for every design. We further report the total number of buffers inserted for each design in Table 5. We find that for all designs, WID uses the least number of buffers compared to NOM and $D2D$. This results show that our WID algorithm can wisely insert buffers in the routing tree such that the final buffered solution achieves real RAT optimization in the presence of process variations.

Finally, we experimentally verify that different choices of $\overline{p_L}$ and $\overline{p_T}$ in (4) and (5) indeed have little impact on RAT optimization in practice. We have tried different combinations of $\overline{p_L}$ and $\overline{p_T}$ (from 0.5 to 0.95) for the same set of experiments as reported in Table 3 and 4. However, among all tested experiments, we see less than 0.1% difference in the final optimal RAT at the root. This observation is expected as we have discussed in Section 2.3.

6. CONCLUSION AND DISCUSSION

An efficient two-parameter (2P) based pruning rule has been proposed for dynamic programming based buffer insertion. Under the normality assumptions on distribution, we have theoretically proved that the 2P-based pruning rule has linear complexity (both time and space), which enables efficient implementation of the buffer insertion algorithm considering both inter-die and intra-die spatially correlated process variations. Experiment results have confirmed the linear scalability of our algorithm, whose capacity has been increased by more than thousands of times compared to [7]. We have applied the algorithm for timing optimization considering correlated process variations and concluded that process variation must be considered for real optimal designs, and buffer insertion without considering spatial correlated variation would reduce the timing yield by more than 50%, this calls for developing efficient algorithms for IC designs to attack the process variation effects actively.

In this work, we only considered timing optimization for signal nets. In the future, we intend to apply the same 2P-based pruning rule and develop efficient algorithms for clock skew minimization.

7. REFERENCES

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

[2] A. Agarwal, D. Blaauw, V. Zolotov, and S. Vrudhula, "Computation and refinement of statistical bounds on circuit delay," in *DAC 03*, Jun 2003.

[3] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. Design Automation Conf*, Jun 2004.

[4] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 865–868, 1990.

[5] J. Lillis, C. K. Cheng, and T. T. Y. Lin, "Simultaneous routing and buffer insertion for high performance interconnect," in *Proc. the Sixth Great Lakes Symp. on VLSI*, 1996.

[6] V. Khandelwal, A. Davoodi, A. Nanavati, and A. Srivastava, "A probabilistic approach to buffer insertion," in *Proc. Int. Conf. on Computer Aided Design*, 2003.

[7] J. Xiong, K. Tam, and L. He, "Buffer insertion considering process variation," in *Proc. Design Automation and Test in Europe*, 2005.

[8] 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. Int. Symp. on Physical Design*, April 2005.

[9] W. Shi and Z. Li, "An o(nlogn) time algorithm for optimal buffer insertion," in *DAC*, Jun 2003.

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

[11] T. Wilder, "Multi chip mask die rotation and mirror to minimize lens exposure aberrations on chip performance," in *The IBM Technical Disclosure Bulletin, on-line http://www.priorartdatabase.com/*, Dec. 2004.

[12] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Proc. Int. Conf. on Computer Aided Design*, pp. 900 – 907, Nov. 2003.