Fast Min-Cost Buffer Insertion under Process Variations*

Ruiming Chen and Hai Zhou
Electrical Engineering and Computer Science
Northwestern University, Evanston, IL

Abstract

Process variation has become a critical problem in modern VLSI fabrication. In the presence of process variation, buffer insertion problem under performance constraints becomes more difficult since the solution space expands greatly. We propose efficient dynamic programming approaches to handle the min-cost buffer insertion under process variations. Our approaches handle delay constraints and slew constraints, in trees and in combinational circuits. The experimental results demonstrate that in general, process variations have great impact on slew-constrained buffering, but much less impact on delay-constrained buffering, especially for small nets. Our approaches have less than 9% runtime overhead on average compared with a single pass of deterministic buffering for delay constrained buffering, and get 56% yield improvement and 11.8% buffer area reduction, on average, for slew constrained buffering.

Categories & Subject Descriptors

B.7.2 [Hardware]: Integrated Circuits-Design aids;
J.6 [Computer-Aided Engineering]: Computer-aided design

General Terms

Algorithms, performance, reliability

Keywords

Buffer insertion, variation, statistical optimization

1 Introduction

Interconnect delays have become dominant in modern deep sub-micron (DSM) designs with the continuously shrinking VLSI feature sizes. Buffer insertion is widely used to reduce the interconnect delays [2,3,6,19,23,32]. These researches are focusing on the buffering on a single net. Saxena *et al.* [22] predicted that 70% of cells will be dedicated to buffers within a few process generations. An effective buffer insertion in a whole circuit is needed. Recently, Sze *et al.* [26] proposed a path-based buffering technique; Chen and Zhou [7] proposed a network flow based buffering technique; Waghmode *et al.* [29] proposed a look-ahead approach to handle the buffering of a whole circuit. But none of them considered process variations.

Process variations become prominent in fabrication. The traditional corner-based analysis and optimization techniques become prohibitive. Nowadays, many statistical static timing analysis (SSTA) approaches [5, 28] emerged. Propagating distributions instead of single values, these techniques are much more efficient. Based on these, some statistical buffering techniques [8, 10, 17, 30, 31] also emerged. But all of them are considering delay minimization. For the cost minimization problem, the solution space is much bigger, and [24] has proved that the problem even without process variations is NP-complete. The pruning approaches in [10, 17, 31] are expensive, and [30] was shown to have worse solutions than the deterministic buffering [8].

Buffer insertion is generally a discrete optimization problem: there are only limited types of buffers in a library, and the possible buffering locations are restricted because of forbidden areas. For discrete optimization problems, statistical optimization techniques as in [9, 12, 25] may not get much better solutions than deterministic optimization techniques. Sinha *et al.* [25] analyzed the situation where the statistical gate sizing is necessary. It is well-known that statistical optimization is much less efficient than its deterministic counterpart in general. Therefore, for discrete optimization problem, if we do the deterministic optimization first, and then an iterative refinement based on the deterministic results may get decent results efficiently.

The slew constrained buffering is done before the delay constrained buffering in the IBM physical synthesis methodology, and most nets may already satisfy the delay constraints after the slew constrained buffering [20]. Recently, some slew constrained buffering researches [14, 21] emerged. But they did not consider process variations. He *et al.* [13] considered the slew propagation, but it did not consider the variances on the interconnects and the correlations among parameter variations.

This paper considers both the delay constrained and the slew constrained min-cost buffering problems. With the huge number of buffers, the power consumption of those buffers becomes prominent, and is proportional to the total buffer area [19]. Thus, our objective is to minimize the total buffer area. Our approaches can be easily extended to the optimization of other cost metrics. The rest of this paper is organized as follows. Section 2 presents the delay modeling of interconnects, and Section 3 proposes our buffering approaches to handle the delay constrained and the slew constrained cost minimization problem in buffering with the consideration of process variations. Section 4 proposes our buffering approach for a combinational circuit with the consideration of process variations. Then, Section 5 shows the experimental results on our approaches, and finally, the conclusion is drawn in Section 6.

2 Preliminary

Given a routing tree with a distributed RC network, the classic van Ginneken's algorithm [27] computes non-inferior solutions bottom-up from the sinks to the root during the buffering. The objective is to insert buffers such that the maximal delay from the root to sinks is minimized.

Lillis *et al.* [19] extended van Ginneken's algorithm to consider the cost minimization. Let (P_v, D_v, C_v) represent a buffering solution of the subtree rooted at node v, where P_v represents the cost, D_v represents the maximal delay to sinks, and C_v is the downstream capacitance. If there are two solutions (P_1, D_1, C_1) and (P_2, D_2, C_2) satisfying $P_1 \leq P_2$, $D_1 \leq D_2$ and $C_1 \leq C_2$ at one node, (P_2, D_2, C_2) is inferior,

*This work was supported by NSF under CCR-0238484.

DAC 2007, June 4-8, 2007, San Diego, California, USA

and (P_1, D_1, C_1) *dominates* (P_2, D_2, C_2).

When process variations are considered, the wire length, wire width and wire height are no longer deterministic values. [15] shows that the introduced variation on the interconnect can be more than 30% of the nominal value. The parameters (e.g., Leff) of buffers are no longer deterministic values either. We assume that all the random variables have Gaussian distributions. Through principal component analysis (PCA) [5, 18], each random variable X is represented as a canonical form: $x_0 + \sum_{i=1}^{n} x_i \epsilon_i + x_{n+1} R_x$, where ϵ_i's are independent random variables with the standard Gaussian distribution, x_0 is the mean value of X, x_i's are coefficients, and R_x is an independent random variable with the standard Gaussian distribution.

When a wire or a buffer is attached to a node, the delay computation of the new solution involves the multiplication of two Gaussian random variables for the Elmore delay model. Assuming that the result still has the Gaussian distribution, [8, 30] get analytic formula for this operation. We will stick to the approach in [8]. For the D2M delay model, the approach in [1] is used.

In this paper, we are considering the following three related buffering problem.

Problem 1 *Delay constrained min-cost buffering on a net: given a routing tree where possible buffering locations are specified, insert buffers such that the probability that the maximal delay from the root to sinks is no greater than a given number is no less than a given constraint η and the cost (e.g., total buffer area) is minimized.*

Problem 2 *Slew constrained min-cost buffering on a net: given a routing tree where possible buffering locations are specified, insert buffers such that the probability that the input slew at each buffer or sink is no greater than a given number is no less than a given constraint η and the cost (e.g., total buffer area) is minimized.*

Problem 3 *Delay constrained min-cost buffering in a combinational circuit: given a routing combinational circuit where possible buffering locations are specified, insert buffers such that the probability that the maximal delay from the primary inputs to the primary outputs is no greater than a given number is no less than a given constraint η and the cost (e.g., total buffer area) is minimized.*

Note that it is not necessary to consider the delay constraints and the slew constraints simultaneously, because most of nets satisfy the delay constraints when the slew constraints are satisfied, and those nets violating the delay constraints after the slew constrained buffering can easily satisfy the delay constraints through the delay constrained buffering [14].

3 Min-cost buffering on a net

3.1 Delay constrained buffering

With process variations, a straight-forward extension of the optimality condition of the deterministic min-cost buffering is that (P_1, D_1, C_1) is **inferior** iff there exist another solution (P_i, D_i, C_i) on the same node such that

$$Pr(P_1 \geq P_i, D_1 \geq D_i, C_1 \geq C_i) = 1.$$

In reality, using this prune criteria cannot prune many solutions since it is difficult for Gaussian random variables to satisfy the 100% probability, so we relax it to

$$Pr(P_1 \geq P_i, D_1 \geq D_i, C_1 \geq C_i) \geq \eta,$$

where η is a given real number in $(0.5, 1)$.

Xiong *et al.* [31] ignored correlations between delays and capacitances, and proved that the transitive ordering property is satisfied. Now we prove that the transitive ordering property does not always hold when the correlations between delay and capacitance are considered.

Theorem 1 *Suppose random variables $C1$, $C2$, $C3$, $D1$, $D2$, and $D3$ having a joint Gaussian distribution satisfy*

$$Pr(D_3 \geq D_2, C_3 \geq C_2) \geq \eta \text{ and } Pr(D_2 \geq D_1, C_2 \geq C_1) \geq \eta,$$

where $\eta \in [0.5, 1)$. Then $Pr(D_3 \geq D_1, C_3 \geq C_1) \geq \eta$ does not always hold.

Proof: Let A, B, C and D be random variables with a joint Gaussian distribution. We need to prove that if

$$Pr(A \leq 0, B \leq 0) \geq \eta \quad and \quad Pr(C \leq 0, D \leq 0) \geq \eta,$$

where $\eta \in [0.5, 1)$, it is possible to have

$$Pr(A + C \leq 0, B + D \leq 0) < \eta.$$

We found the following case: the means of A and B are -0.68, the means of C and D are -0.32, and all of them have their standard deviations (σ) equal to 1. The covariance matrix is

$$\begin{pmatrix} 1.00 & -0.99 & 0.36 & -0.36 \\ -0.99 & 1.00 & -0.36 & 0.36 \\ 0.36 & -0.36 & 1.00 & 0.72 \\ -0.36 & 0.36 & 0.72 & 1.00 \end{pmatrix}$$

Then $Pr(A \leq 0, B \leq 0) = 0.5035$, and $Pr(C \leq 0, D \leq 0) = 0.5098$, while $Pr(A + C \leq 0, B + D \leq 0) = 0.4922$. □

Corollary 1.1 *Suppose $\mathbf{X}$ and $\mathbf{Y}$ are two vectors of random variables with a joint Gaussian distribution, and*

$$Pr(\mathbf{X} \leq 0) \geq \eta, \quad and \quad Pr(\mathbf{Y} \leq 0) \geq \eta,$$

where $\eta \in [0.5, 1)$. If both $\mathbf{X}$ and $\mathbf{Y}$ have more than one elements, $Pr(\mathbf{X} + \mathbf{Y} \leq 0) \geq \eta$ does not always hold.

This implies if A is inferior to B, and B is inferior to C, A may not be inferior to C. An important result based on this implication is that the order of the pruning of inferior solutions affects the final solutions. For example, suppose B is inferior to C, and there are k_1 solutions inferior to B, and k_2 solutions inferior to C, then it is possible to have $k_1 > k_2$, thus if B is pruned by C first, those solutions inferior to B but not inferior to C can not be pruned because B is no longer in the solution list. Therefore, in order to get a minimal set of non-inferior solutions, we have to compute the number of solutions dominated by each solution, and keep those solutions that dominate more solutions. Thus, the prune in a solution list with n solutions needs to compute the dominance relation between each pair of solutions, which leads to n^2 time complexity, and a merge procedure that merges two solution lists with m and n solutions needs $m^2 n^2$ computation of the dominance relation.

Another main difficulty in min-cost buffering is its big solution space. During the buffering, the mean value of the downstream capacitance of a statistical solution is always equal to the downstream capacitance of a deterministic solution since only the sum operation is involved in the capacitance computation. But for the delay, since the non-linear multiplication operation is involved, the mean value of the delay of a statistical solution may be different from the delay

of the deterministic solution. In addition, because the number of statistical solutions is always not less than the number of deterministic solutions, we can select a statistical solution that is prone to be better than the deterministic one. Suppose a set of deterministic solutions having the same cost at a node v are (P, D_1, C_1), (P, D_2, C_2), ..., (P, D_n, C_n), and they satisfy

$$C_1 < C_2 < \ldots < C_n \quad and \quad D_1 > D_2 > \ldots > D_n.$$

With process variations, the delay and capacitance of this set of solutions become random variables. Suppose their corresponding statistical representations are $\{(P, D_i', C_i') | 1 \leq i \leq n\}$, and then they must satisfy $\mu(C_i') = C_i$. Now if we can find another set of solutions $\{(P, D_i'', C_i'')\}(1 \leq i \leq n)$, such that for any (P, D_i', C_i'), there exist a (P, D_i'', C_i'') such that

$$(\mu(D_i'') \leq \mu(D_i')) \wedge (\mu(C_i'') \leq \mu(C_i') = C_i), \quad (1)$$

the yield is expected to be higher. If P is also random, we can pick those solutions satisfying

$$(\mu(P_i'') \leq \mu(P_i')) \wedge (\mu(D_i'') \leq \mu(D_i')) \wedge (\mu(C_i'') \leq \mu(C_i')). \quad (2)$$

So this is actually a greedy algorithm: always select the solution with higher probability to be better. As will be demonstrated in our experimental results, the deterministic buffering that fixes parameters at their worst case values gets much worse solutions on average. This is reasonable since the probability that a sample occurs in the region close to the nominal value is much bigger for Gaussian distributions. Therefore, here we select the solutions based on their mean value instead of their $\mu + 3\sigma$ value, and the computation of deterministic solutions fixes parameters at their nominal values.

The general framework of our statistical optimization approach is shown in Fig. 1. Let us see a specific application of the framework on buffering. The deterministic buffering that assumes parameters are at their nominal values is done first to get a set of non-inferior solutions at each node. Then we do the statistical buffering bottom-up from the sinks to the root. The merge of two solution lists with m and n solutions may generate mn new solutions, so we are focusing on the merge procedure. When a merge is encountered, for each deterministic solution K in each branch solution list, we pick a statistical solution with its cost less than or equal to the cost of K, its mean downstream capacitance less than or equal to the downstream capacitance of K, and its mean delay is the minimal[1]. So the number of picked statistical solutions to be merged is close to the number of deterministic non-inferior solutions, which can greatly improve the efficiency of the algorithm. This algorithm is denoted as DL-FSBI. Note that the framework in Fig. 1 has many different implementations. For example, the deterministic buffering may fix parameters at other corners, and also the statistical solutions can be selected based on the comparison on the $\mu + 3\sigma$ values.

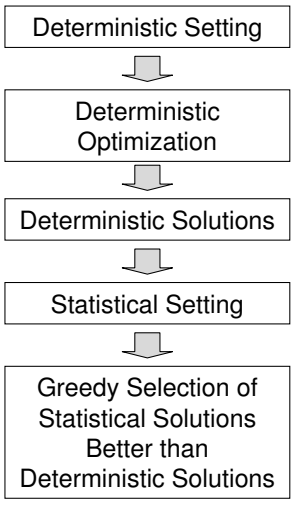

Figure 1: General flow of our statistical optimization framework.

Note that the pruning is an expensive step since it is at least needed to compute the dominance relations between each pair of non-inferior solutions. One variate of DL-FSBI is to do the solution picking without the pruning. This can greatly improve the efficiency. This variate is denoted as DL-PK-FSBI. Another big difference between DL-PK-FSBI and DL-FSBI is that DL-PK-FSBI picks solutions in the merged solution list, and those solutions that cannot be picked according to Eq.(2) are directly discarded, while DL-FSBI picks solutions in the solution lists waiting for merge, and all those newly generated non-inferior solutions are stored in a sorted list. Thus, the number of solutions in DL-PK-FSBI is always much smaller. Our experimental results will demonstrate that the quality of solutions does not degrade much in the efficient DL-PK-FSBI. DL-PK-FSBI-V2 is a variate of DL-PK-FSBI, and it picks solutions based on both their mean values and their $\mu + 3\sigma$ values.

3.2 Slew constrained buffering

The rising or falling time of a signal transition is its slew rate. We use the slew model in [16]. Let S_u represent the slew rate at node u. If a buffer b is attached to the node u, the slew rate at u's downstream node v is $S_v = \sqrt{S_{u,out}^2 + S_p^2}$, where $S_{u,out}$ is the output slew rate of the buffer, and S_p is the slew degradation along the path p from u to v. According to Bakoglu's metric [4], $S_p = (\ln 9) D_p$, where D_p is the Elmore delay of path p.

Hu *et al.* [14] proposes a deterministic slew constrained buffering approach that assumes the input slews of buffers are fixed at conservative values, and gets $S_{u,out} = P_b C_u + Q_b$, where P_b and Q_b are empirical fitting parameters. This assumption is also used in our approach for simplicity, and our approach can be easily extended to handle the cases without this assumption. Let (P_v, S_v, C_v) represent a solution at node v. We prove the following theorem.

Theorem 2 *If the input slews of buffers are fixed at conservative values, there is only one type of buffer, and the input capacitances of buffers are no greater than sink capacitances, the costs of the non-inferior solutions at a potential buffering node have at most two distinct values for the buffer area minimization problem.*

With process variations, the delays and the downstream capacitance become random variables, so do the slew rates. Assuming that delays and downstream capacitance have Gaussian distributions, we know that path slew degradations S_p's also have Gaussian distributions. Let $D_p = D_0 + \sum_{i=1}^{n} D_i \epsilon_i + D_{n+1} R_{D_p}$, and $C_v = C_0 + \sum_{i=1}^{n} C_i \epsilon_i + C_{n+1} R_{C_v}$, then $S_p = \ln(9) D_0 + \sum_{i=1}^{n} \ln(9) D_i \epsilon_i + \ln(9) D_{n+1} R_{D_p}$, and $S_{u,out} = (P_b C_0 + Q_b) + \sum_{i=1}^{n} P_b C_i \epsilon_i + P_b C_{n+1} R_{C_v}$. Here, we assume that P_b and Q_b are deterministic values for simplicity. If they also have process variations, the moment-matching approach in [8] can be used here to get a canonical form for $S_{u,out}$.

We use the same dynamic programming framework as in [14]. We traverse the tree bottom-up from sinks to the root, and at each node, we prune those inferior solutions. Since we are considering the slew instead of the delay, the inferior condition becomes:

Definition 1 (C_{v1}, S_{v1}, P_{v1}) *is* **inferior** *iff there exists another solution* (C_{v2}, S_{v2}, P_{v2}) *satisfying*

$$Pr(C_{v2} \leq C_{v1}, S_{v2} \leq S_{v1}, P_{v2} \leq P_{v1}) \geq \eta,$$

[1] if costs are also random, we select the solution that has the biggest probability that its delay and cost are less than the delay and cost of K, respectively.

at the same node, where η is a given constant real number in $[0.5, 1]$.

When a buffer is attached to a node, we need to compute the probability that the output slew rate satisfies the slew constraints. If it does not satisfy the constraint, the solution needs to be discarded. The square root computation of random variables is involved in the computation S_v, which is not convenient for computation. So we compute S_v^2 instead of S_v.

Let Sq_v represent the S_v^2. Then according to $E(A^2) = E(A)^2 + \sigma(A)^2$, we have

$$\begin{aligned} E(Sq_v) &= ((P_b C_0 + Q_b)^2 + \sum_{i=1}^{n+1} P_b^2 C_i^2) + (\ln(9) D_0)^2 \\ &\quad + \sum_{i=1}^{n+1} (\ln(9) D_i)^2 \end{aligned}$$

Also

$$E(Sq_v^2) = E(S_{u,out}^4) + E(S_p^4) + 2E(S_{u_o ut}^2 S_p^2).$$

Suppose X has a Gaussian distribution with mean μ and sigma σ, then the 4th raw moment of X is $E(X^4) = \mu^4 + 6\mu^2\sigma^2 + 3\sigma^4$. Suppose $X = x_0 + \sum_{i=1}^{n} x_i \epsilon_i + x_{n+1} R_x$, and $Y = y_0 + \sum_{i=1}^{n} y_i \epsilon_i + y_{n+1} R_y$. Then

$$\begin{aligned} E(X^2 Y^2) &= x_0^2 y_0^2 + x_0^2 \sigma_Y^2 + y_0^2 \sigma_X^2 + 2 \sum_i x_i^2 y_i^2 \\ &\quad + \sigma_X^2 \sigma_Y^2 + 4 x_0 y_0 cov(X, Y) \\ &\quad + 4 \sum_{i=1, j=i+1, i \neq j}^{i=n-1, j=n} x_i y_i x_j y_j \end{aligned}$$

Now we get $E(Sq_v^2)$ and $E(Sq_v)$, and then $\sigma^2(Sq_v) = E(Sq_v^2) - E(Sq_v)^2$.

It is not necessary for the Sq_v to be represented in a canonical form since it is needed only when a buffer is inserted, and is never propagated. Once we know the mean and the variance of the Sq_v, assuming that Sq_v has a Gaussian distribution, we compute the probability that the slew constraint is satisfied and discard those solutions that violate the slew constraint. Our experiments demonstrate that this approximation is accurate enough for the estimation of the yield. This slew constrained min-cost buffering algorithm is called SW-FSBI.

Our approach can be easily extended to handle the cases where the output slew of a buffer is specified by a 2D look-up table (LUT): change the deterministic comparison in the approach of [14] to statistical comparison. For example, if the original comparison is $A \leq B$, we change it to $Pr(A \leq B) \geq \eta$, where η is a given value between 0.5 and 1.

3.3 Analysis

We observed that the delay constrained buffering is much less sensitive to process variations than the slew constrained buffering: it often occurs that the statistical delay constrained buffering gets the exactly same solution as the deterministic delay constrained nominal buffering, while the slew constrained buffering gets much better results. The merge of two solution lists with m solutions and n solutions respectively may generate mn new solutions in statistical situation. But if the solutions in a solution list always have bigger delays than those in solution lists of other branches, the merged solutions always have the same (or similar) delays with those solutions in the dominating solution list, and thus, the number of merged solutions is much less than mn, which is not good for the statistical buffering. For whole circuit buffering, this situation often occurs since the criticalities of wires in different branches are often much different. In addition, most nets in a circuit are small, so the merge operation seldom occurs, and thus the number of newly generated solutions is not big, which makes the statistical delay-constrained buffering behave not much better either. Therefore, we can use the fast deterministic buffering as a pre-processing step to check if it is necessary to do the statistical delay-constrained buffering. Slew constrained buffering is a local optimization problem, and some works just used the wire length to do slew constrained buffering at the early stage. So slew constrained buffering is more sensitive to variations.

4 Min-cost buffering on a combinational circuit

The assignment of various timing budgets on less-critical paths in a combinational circuit would give multiple buffering solutions, among which we need to select the one with the minimal costs. Sze *et al.* [26] proposed a deterministic path-based buffering technique, where the circuit is partitioned into trees, and Lillis' min-cost buffering algorithm is used to buffer each tree. It is straight forward to extend that algorithm to consider process variations: our statistical min-cost delay constrained buffering algorithm is used to buffer each tree. But as shown in [29], the solutions from [26] have much more buffers than those from [29].

Our idea is that the deterministic optimization techniques are used to get a decent solution, and a statistical refinement is done to get a better solution. This algorithm is denoted as CC-FSBI. The first step is the deterministic buffering of the circuit. Here as mentioned before, parameters use their nominal values. The whole circuit buffering techniques in [7, 29] can be used here. The second step is the statistical refinement: we use DL-FSBI to buffer each net. Here the required times and the load capacitances of sinks use those deterministic values computed by the first step.

The second step of CC-FSBI buffers each net, and does not re-budget the timing between nets. If a gate has only one input, we can buffer the combination of its fanin net and its fanout net as a whole tree by viewing the gate as an existing buffer. If a gate has more than one inputs, we do not combine them, so the path-reconvergence problem as mentioned in [26] is avoided. A variate of CC-FSBI is to use this technique to re-budget the timing between neighbor nets.

5 Experimental results

DL-FSBI, SW-FSBI and CC-FSBI are implemented in C++ and tested on a set of big nets and combinational circuits. In the statistical buffering, process parameters are set to have at most 30% deviation ($3\sigma/\mu$) from their nominal values. We also implemented a deterministic delay-constrained buffering approach, denoted as DL-DETBI, that uses Lillis's algorithm and Shi's speed-up techniques [24] for delay-constrained min-cost buffering, and the deterministic slew-constrained min-cost approach [14], denoted as SW-DETBI, for comparison. In DL-DETBI and SW-DETBI, all the process parameters are fixed at their nominal values or their $\mu + 3\sigma$ values. All the experiments were run on a Linux Redhat machine with a 2.4 GHz Xeon CPU and 2.0 GB memory.

5.1 Delay constrained buffering

We also implemented a statistical buffering technique that does the pruning only according to the optimality condition, and does not do the greedy selection of statistical solutions. This is denoted as SBI. Therefore, we know if the quality of

solutions degrades in DL-FSBI when we compare DL-FSBI and SBI. We tested DL-FSBI and SBI on those nets in the circuits from [26]. There are four types of buffers. These nets are generally very small (most of them have less than 4 sinks), so SBI can finish them in a short time. But we did not see any degradation on the yield for DL-FSBI on this set of nets.

For delay constrained buffering, we also compare the yields from DL-DETBI and DL-FSBI for each cost value. We use the testcases from [8]. There is one type of buffer. Elmore delay model is used for the delay computation. DL-DETBI has two variates: DL-NOMBI has all the parameters at their nominal values, and DL-WSTBI has all the parameters at their worst case ($\mu + 3\sigma$) values. For each cost value, we set the delay constraint to the sum of the minimal delay achieved by DL-NOMBI and its standard deviation. So the yield from DL-NOMBI is always close to 84.13%. Table 1 shows the comparison results. Note that SBI cannot finish any of these testcases in one day, so DL-FSBI is much more efficient than SBI. A good property of our approach is that the efficiency of DL-FSBI also depends on the number of non-inferior solutions in the deterministic buffering step, so with the help of existing speed-up techniques on deterministic buffering (e.g., sampling), DL-FSBI can achieve much better efficiency. The "# cost" column shows the total number of distinct costs; the "# diff" column shows the number of cost values on which the DL-FSBI/DL-PK-FSBI has more than 4% improvement on the yield compared with DL-NOMBI; the "Avg (%)" columns show the average improvement on the yields among those cost values where the DL-FSBI/DL-WSTBI has more than 4% improvement/degradation; the "Max (%)" column show the maximal improvement on the yields among all the cost values. Actually, we do not see any case where the DL-FSBI has yield degradation. Note that all the yields data here have been verified by Monte-Carlo simulation. The results indicate that the DL-FSBI can achieve big yield improvement (e.g., p1 has 15.87% maximal improvement) on a set of cost values. The DL-WSTBI always gets much worse solutions than DL-NOMBI. DL-PK-FSBI is much more efficient than DL-FSBI, and has only 8.99% runtime overhead on average compared with DL-NOMBI. So the statistical buffering step in DL-PK-FSBI is extremely fast. DL-PK-FSBI has only 6% degradation on the "# diff" values on average compared with DL-FSBI. Those statistical solutions better than deterministic solutions with respect to $\mu + 3\sigma$ are also picked in DL-PK-FSBI-V2, and the degradation becomes even less. For example, the "# diff"s in "r2" and "r3" are both improved to 23.

On the other hand, comparing those values in the "# diff" column and the "# cost" column, we see that DL-FSBI can only improve the yield on a small set (max 15/131=11%) of cost values. For "p2", the statistical buffering does not have any big improvement. Deng *et al.* [11] analyzed the buffering on a two-pin net and showed that it is not necessary to consider process variations based on some ideal conditions (e.g., no blockage). Alpert *et al.* [2] also showed that the density of wire segments does not change the solutions greatly. From our experiments, we see similar things.

We also tested our approach on the much more accurate delay model D2M. The comparison results are shown in Table 2. The results indicate that DL-PK-FSBI-V2 achieves 8.51% improvement on timing yield on average.

5.2 Slew constrained buffering

For the slew constrained min-cost buffering, we compare the yields from SW-FSBI and SW-DETBI [14]. The comparison results are shown in Table 3. The "Max" and "Min" columns show the maximal and the minimal number of inserted buffers, respectively. The objective yield in SW-FSBI is set to 97%. Here, we also used Monte-Carlo simulation to compute the yield for each solution, and observed that the computation of the yield using the proposed approximation technique is very accurate (difference is less than 1% on average). The results indicate that the SW-FSBI achieves the objective yield on all the cases except the last one, and achieves 56% yield improvement with 3.3% cost overhead on average compared with the deterministic buffering that uses nominal parameter values. The deterministic buffering that uses the worst case values achieves 100% yield on most cases, but it cannot get feasible solutions for two cases where the slew constraints are tight. In addition, it has 11.8% cost overhead on average compared with the SW-FSBI. The SW-FSBI is also efficient since the slew constrained buffering does not have many non-inferior solutions. In summary, the consideration of process variations on slew constrained min-cost buffering can greatly improve the yield.

Table 2: Delay constrained min-cost statistical buffering vs. deterministic buffering on a net for D2M delay model

Nets	DL-NOMBI		DL-WSTBI	DL-PK-FSBI-V2		
	#cost	time(s)	avg(%)	#diff	avg(%)	time(s)
p1	44	11.07	-8.49	9	11.06	19.04
p2	87	770.01	-12.89	24	11.75	877.12
r1	127	375.62	-32.59	35	3.59	413.21
r2	168	530.02	-25.36	52	7.51	588.35
r3	142	736.38	0.00	32	8.63	813.16

5.3 Circuit buffering

We test CC-FSBI on a set of testcases from [26]. The nets in these testcases are very small, and most of those nets have only one sink. There are four types of buffers. The deterministic buffering approach called CC-DETBI works as follows: the whole circuit buffering approach in [7] is done; then in order to improve the yield, the net buffering approach in [24] is used to buffer each net such that the maximal delay from the source to sinks is minimized while the total buffer area is also within a specified range. The comparison results on a testcase "a1" are shown in Table 4. Column "Cost" shows the cost from CC-FSBI over the cost from CC-DETBI. CC-FSBI does not have big improvement on the yield, and its reduced cost is also not very prominent. The other cases have similar results, which are omitted because of the space limit. Therefore, in general, a deterministic buffering approach is accurate enough to handle the delay constrained buffering of those circuits where most nets are small.

Table 4: Delay constrained min-cost statistical buffering vs. deterministic buffering on a combinational circuit

Circuits		CC-DETBI		CC-FSBI		
	Delay	Yield	Time (s)	Cost	Yield	Time (s)
a1	291 ps	83.52%	162.93	99.86%	85.44%	289.27

6 Conclusion

With the consideration of process variations, we proposed effective approaches to handle the delay constrained min-cost buffer insertion and the slew constrained min-cost buffer insertion on a net, and also proposed an approach to handle the combinational circuit buffering problem. We observed that process variations do not have great impact on the delay constrained buffering, especially for small nets, but they do have great impact on the slew constrained buffering that is mostly used in the current industry buffering practice [20].

Table 1: Delay constrained min-cost statistical buffering vs. deterministic buffering on a net for Elmore delay model

Nets		DL-NOMBI		DL-WSTBI	DL-FSBI				DL-PK-FSBI		DL-PK-FSBI-V2
	# sinks	# cost	time (s)	avg (%)	# diff	max (%)	avg (%)	time (s)	#diff	time (s)	#diff
p1	269	131	35.31	-52.53	15	15.87	13.93	384.30	15	38.81	15
p2	603	164	251.81	-6.43	0	0	0	3861.59	0	272.17	0
r1	267	212	168.76	-20.79	9	9.19	5.54	2167.96	9	185.60	9
r2	598	374	1711.26	-26.63	24	14.50	5.65	23367.98	21	1801.77	23
r3	862	359	1607.01	-16.65	24	14.00	5.99	33149.76	20	1794.67	23

Table 3: Slew constrained min-cost statistical buffering vs. deterministic buffering on a net

Nets		Nominal				Worst			SW-FSBI			
Name	Slew (ps)	Max	Min	Yield (%)	Time (s)	Max	Min	Yield (%)	Max	Min	Yield (%)	Time (s)
p1	600	57	55	54.8	0.01	57	55	100	57	55	100	0.03
	300	113	111	39.9	0.01	127	125	100	122	120	100	0.02
	200	191	189	38.7	0.01	219	217	100	211	209	100	0.02
p2	600	127	125	43.85	0.03	134	132	100	133	131	97.75	0.05
	300	264	262	48.31	0.03	283	281	100	270	268	99.04	0.05
	200	445	443	46.64	0.03	494	492	100	457	455	99.87	0.05
r1	1000	63	61	60.63	0.02	78	76	100	64	62	96.69	0.03
	800	75	73	58.00	0.02	87	85	100	78	76	97.26	0.03
	600	88	86	40.59	0.02	N/A	N/A	N/A	89	87	97.64	0.03
r2	1300	96	94	28.53	0.05	118	116	100	98	96	98.57	0.08
	1100	109	107	38.16	0.04	135	133	100	111	109	99.70	0.07
	1000	117	115	43.28	0.04	N/A	N/A	N/A	119	117	99.82	0.06
r3	1000	145	143	24.66	0.05	178	176	100	151	149	98.45	0.08
	800	170	168	32.45	0.05	211	209	100	176	174	98.95	0.08
	500	242	240	35.26	0.05	293	291	100	246	244	95.93	0.08

References

[1] K. Agarwal, D. Sylvester, D. Blaauw, F. Liu, S. Nassif, and S. Vrudhula. Variational delay metrics for interconnect timing analysis. In *DAC*, pages 381–384, 2004.

[2] C. J. Alpert and A. Devgan. Wire segmenting for improved buffer insertion. In *DAC*, pages 588–593, 1997.

[3] C. J. Alpert, M. Hrkic, and S. T. Quay. A fast algorithm for identifying good buffer insertion candidate locations. In *ISPD*, pages 47–52, 2004.

[4] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.

[5] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single pert-like traversal. In *ICCAD*, pages 621–625, 2003.

[6] R. Chen and H. Zhou. A flexible data structure for efficient buffer insertion. In *ICCD*, pages 216–221, 2004.

[7] R. Chen and H. Zhou. Efficient algorithms for buffer insertion in general circuits based on network flow. In *ICCAD*, pages 322–326, 2005.

[8] R. Chen and H. Zhou. Fast buffer insertion for yield optimization under process variations. In *ASP-DAC*, 2007.

[9] K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester. Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation. In *ICCAD*, pages 1023–1028, 2005.

[10] A. Davoodi and A. Srivastava. Variability-driven buffer insertion considering correlations. In *ICCD*, 2005.

[11] L. Deng and M. D. Wong. Buffer insertion under process variations for delay minimization. In *ICCAD*, pages 317–321, 2005.

[12] M. Guthaus, N. Venkateswaran, C. Visweswariah, and Z. Zolotov. Gate sizing using incremental parameterized statistical timing analysis. In *ICCAD*, pages 1029–1036, 2005.

[13] L. He, A. Kahng, K. H. Tam, and J. Xiong. Simultaneous buffer insertion and wire sizing considering systematic CMP variation and random Leff variation. In *ISPD*, pages 78–85, 2005.

[14] S. Hu, C. J. Alpert, J. Hu, S. Karandikar, Z. Li, W. Shi, and C. N. Sze. Fast algorithms for slew constrained minimum cost buffering. In *DAC*, pages 308–313, 2006.

[15] F. Huebbers, A. Dasdan, and Y. Ismail. Computation of accurate interconnect process parameter values for performance corners under process variations. In *DAC*, pages 797–800, 2006.

[16] C. Kashyap, C. Alpert, F. Liu, and A. Devgan. Closed form expressions for extending step delay and slew metrics to ramp inputs. In *ISPD*, pages 24–31, 2003.

[17] V. Khandelwal, A. Davoodi, A. Nanavati, and A. Srivastava. A probabilistic approach to buffer insertion. In *ICCAD*, pages 560–567, 2003.

[18] W. J. Krzanowski. *Principles of Multivariate Analysis*. Oxford University Press, 2000.

[19] J. Lillis, C. K. Cheng, and T. T. Y. Lin. Optimal wire sizing and buffer insertion for low power and a generalized delay model. *IEEE Journal of Solid-State Circuits*, 31:437–447, 1996.

[20] P. J. Osler. Placement driven synthesis case studies on two sets of two chips: Hierarchical and flat. In *ISPD*, pages 190–197, 2004.

[21] Y. Peng and X. Liu. Low-power repeater insertion with both delay and slew rate constraints. In *DAC*, pages 303–307, 2006.

[22] P. Saxena, N. Menezes, P. Cocchini, and Desmond A. Kirkpatrick. The scaling challenge: Can correct-by-construction design help? In *ISPD*, pages 51–58, 2003.

[23] W. Shi and Z. Li. A fast algorithm for optimal buffer insertion. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 24(6):879–891, 2005.

[24] W. Shi, Z. Li, and C. J. Alpert. Complexity analysis and speedup techniques for optimal buffer insertion with minimum cost. In *Proc. Asia and South Pacific Design Automation Conference*, pages 609–614, 2004.

[25] D. Sinha, N. Shenoy, and H. Zhou. Statistical gate sizing for timing yield optimization. In *ICCAD*, pages 1037–1041, 2005.

[26] C. N. Sze, C. J. Alpert, J. Hu, and W. Shi. Path based buffer insertion. In *DAC*, pages 509–514, 2005.

[27] L. P. P. P. van Ginneken. Buffer placement in distributed RC-tree networks for minimal Elmore delay. In *ISCAS*, pages 865–868, 1990.

[28] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. In *DAC*, pages 331–336, 2004.

[29] M. Waghmode, Z. Li, and W. Shi. Buffer insertion in large circuits with constructive solution search techniques. In *DAC*, pages 296–301, 2006.

[30] J. Xiong and L. He. Fast buffer insertion considering process variations. In *ISPD*, 2006.

[31] J. Xiong, K. Tam, and L. He. Buffer insertion considering process variation. In *DATE*, 2005.

[32] H. Zhou, D. F. Wong, I-Min Liu, and A. Aziz. Simultaneous routing and buffer insertion with restrictions on buffer locations. DAC, 1999.