8

Stochastic Optimization Over Correlated Data Set: A Case Study on VLSI Decoupling Capacitance Budgeting

Yiyu Shi[1], Jinjun Xiong[2] and Lei He[3]
[1]*ECE Dept., Missouri University of Science and Technology, Rolla, MO, 65409*
[2]*IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 10598*
[3]*EE Dept., University of California, Los Angeles, LA, 90095*
U.S.A

1. Introduction

It is very common in engineering society to optimize certain objective functions under the worst scenario among a set of possible scenarios, i.e.,

$$\begin{aligned} &\min_{\mathbf{x}} \max_i \ f(\mathbf{x}) \\ &s.t. \qquad \mathbf{x} \in S(\mathbf{p_i}), \\ &\qquad\quad i = 1, 2, \ldots, \end{aligned} \tag{1}$$

where $\mathbf{p_i}$ are the parameters that control the feasible region S of x. For example, in the contingency analysis of power systems, $\mathbf{p_i}$ is a vector of 0-1 variables, indicating which of the branches are open. Each $\mathbf{p_i}$ corresponds to one contingency situation. Another example is the decoupling capacitance budgeting for very large scale integrated (VLSI) circuits and systems, where $\mathbf{p_i}$ can be different load current profile.

When the number of possible scenarios are small, we can use enumeration to find out the worst case. But when it is large or even infinite, enumeration becomes computationally expensive, sometimes even infeasible, and accordingly, we need some elegant algorithms that can efficiently solve the problem. In this chapter, we will use the decoupling capacitance budgeting problem in very large scale integrated (VLSI) circuits and systems to illustrate one recently developed algorithm when the Pi's are correlated.

The continuous semiconductor technology scaling leads to growing process variations (Agarwal & Nassif, 2007), and statistical optimization has been actively researched to cope with process variations. Recent examples include stochastic gate sizing for power reduction (Bhardwaj & Vrudhula, 2005; Mani et al., 2005) and for yield optimization (Davoodi & Srivastava, 2006; Sinha et al., 2005), stochastic buffer insertion to minimize clock delay (He et al., 2007), and adaptive body biasing with post-silicon tuning (Mani et al., 2006). However, all these papers ignore *operation variation* such as crosstalk difference over input vectors, power supply noise fluctuation over time, and processor temperature variation over workload. We

argue that a better design could be achieved by considering both operation and process variations.

The P/G network has to provide large currents within a short period of time but without causing considerable IR-drop and L*di/dt* noises. The noises on the P/G network can degrade signal integrity of the whole design, causing longer path delay, reduced noise margin, and even logic failures. In the presence of process variation, a fraction of chips after manufacturing may fail to meet the given power noise constraints, even though they were predicted to do so by the deterministic techniques, thus causing unnecessary yield loss. This observation has also been confirmed in recent studies on both statistical timing analysis (Chang & Sapatnekar, 2003; Visweswariah et al., 2004) statistical power network analysis (Ghanta et al., 2005; Kouroussis et al., 2005; Pant, Blaauw, Zolotov, Sundareswaran & Panda, 2004).

Decap budgeting is one of the most effective techniques to reduce the noise in P/G network. Assuming the netlist and the initial placement is given, decap budgeting assigns the right amount of decap to the right location. To solve the decap budgeting problem, most work employs a sensitivity-based optimization technique, such as those solved by either linear programming (Zhao et al., 2006), quadratic programming (Su et al., 2003), or conjugate gradient method (Fu et al., 2004; Li et al., 2005). At each iteration step during optimization, sensitivities of the objective function with respect to various decaps are obtained by running circuit simulations on the adjoint network followed by time-domain convolution (Li et al., 2005; Su et al., 2003). Because both simulation and convolution are time-consuming operations, the overall runtime is high and suffers from the scalability problem for large P/G networks. To mitigate this runtime issue, different techniques have been proposed. For example, (Su et al., 2003) employed piecewise-linear approximation for the time-domain waveforms so that convolution can be carried out faster with bounded accuracy loss. (Fu et al., 2004) exploited regular structures of P/G networks, and reduced circuit sizes by equivalent circuit transformation (such as Y-Δ transformation). Because of the reliance on special P/G structures, the applicability of this technique to large P/G networks is limited and the reduction ratio is not high in general. (Li et al., 2005) employed a divide-and-conquer approach that partitioned a P/G network into a number of sub-circuits so that decap budgeting can be solved efficiently for each sub-circuit. But to consider the inter-dependence between different sub-circuits, an artificial boundary condition has to be imposed, hence the accuracy of the solution cannot be guaranteed. Recently, (Zhao et al., 2006) used macromodeling and linear programming based approaches to solve the decap problem. However, same as the previous studies (Fu et al., 2004; Li et al., 2005; Su et al., 2003), it assumed a maximum current load at every port to guarantee the worst-case design scenario.

The maximum current model is over pessimistic as it ignores operation variation. Specifically, current loads at different ports are correlated and cannot reach the maximum at the same time due to the inherent logic dependency for a given design, hence exhibiting *logic-induced correlation*; and the current at a port also exhibits *temporal correlation*, i.e., the current cannot attain maximum all the time, and depending on the functionality being performed, the current variations for certain periods of clock cycles are correlated.

Unfortunately, few research has been conducted on how to extract these operation correlations. The stochastic modeling of IR drop with respect to given correlated current loads for a P/G network was studied in (Pant, D.Blaauw, Zolotov, S.Sundareswaran & Panda, 2004). However, the paper did not discuss how to extract the correlation of those current loads.

Moreover, it is still not clear how to use the correlation to guide the P/G network design and optimization such as decap budgeting.

In addition, the current loads are affected by process variations. (Ferzli & Najm, 2003) has considered process variation induced leakage variation for power grid analysis. While the leakage power is comparable to the dynamic power because not all components are active simultaneously in a large system-on-chip, we believe that the dynamic peak current is still dominant compared with the leakage current. However, how to design a reliable P/G network in the presence of process variation (particularly L_{eff} variation) has not been explicitly studied in existing work (Fu et al., 2004; Li et al., 2005; Su et al., 2003).

In this chapter, we develop a novel stochastic model for current loads, taking into account operation variation such as temporal and logic-induced correlations and process variations such as systematic and random L_{eff} variation. We propose a formal method to extract operation variation and formulate a new decap budgeting problem using the stochastic current model. We develop an effective yet efficient iterative alternative programming algorithm and conduct experiments using industrial designs. We show that under the same decap area and compared with the baseline model assuming maximum current peaks at all ports, the model considering temporal correlation reduces the noise by up to $5\times$, and the model considering both temporal and logic-induced correlations reduces the noise by up to $17\times$. Compared with using deterministic process parameters, considering L_{eff} variation reduces the mean noise by up to $4\times$ and the 3σ noise by up to $13\times$ when both applying the current model with temporal and logic-induced correlations. Therefore, we convincingly demonstrate the significance of considering both operation and process variations and open a new research direction for optimizing signal, power and thermal integrity with consideration of operation variation.

The remaining of the chapter is organized as follows. We introduce the decap budgeting problem in Section 2, and develop the stochastic current model and parameterized MNA formulation in Section 3. We discuss the algorithms to solve the variation-aware decap budgeting problem in Section 4, and present experiments in Section 5. We conclude in Section 6. An extended abstract of this chapter with less details and no sequential quadratic programming (in Sections 4 and 5.3) was published by the 2007 International Conference on Computer-Aided Design (Shi et al., 2007).

2. Problem formulation

The P/G network can be modeled as a linear RLC network with each segment and pad modeled as a lumped RLC element from extraction. The behavior of any linear RLC network with p ports of interests is fully described by its state representation following the modified nodal analysis (MNA)

$$Gx + C\frac{dx}{dt} = Bu(t), \tag{2}$$

$$y = L_0^T x, \tag{3}$$

where x is a vector of nodal voltages and inductor currents, u is a vector of current sources at all ports, G is the conductance matrix, C is a matrix that includes both inductance and

capacitance elements, B and L_0 are port incident matrices, and y is the output voltages of interests at the p ports.

We model the P/G network noise based upon the response $y(t)$ from (3). Because of the duality between power and ground networks, in the following, we will focus our explanation on the power network design. But it is understood that the same formulation applies to the ground network design as well. Same as (Fu et al., 2004; Li et al., 2005; Su et al., 2003; Visweswariah et al., 2000), we model the power network induced noise at a node as the integral of the voltage drop below a user specified noise ceiling $\overline{U}$ over a certain period of time:

$$z_i = \int_{\Omega_i} (\overline{U} - y_i(t))dt, \tag{4}$$

where Ω_i is the time duration when voltage at port i, y_i, drops below the noise ceiling $\overline{U}$, i.e.,

$$\Omega_i = \{t | y_i(t) \leq \overline{U}\}. \tag{5}$$

The figure of merit that measures the qualify of the whole power network design is defined as the sum of noise at all ports of interest, i.e.,

$$f = \sum_{i=1}^{p} \int_{\Omega_i} (\overline{U} - y_i(t))dt. \tag{6}$$

We will call the noise measurement in (6) simply as noise in the rest of the chapter.

Based upon the noise modeling above, we can formulate the decap budgeting problem as the following optimization problem:

Formulation: **Decap Budgeting**: Given a power network modeled as an RLC network with specified power pads, time-varying current at different ports, and total available white space $\overline{W}$ for decoupling capacitance, the DecapOpt problem determines the places to insert decoupling capacitance and the sizes of each decoupling capacitance, such that the noise defined in (6) is minimized, considering the time-varying current $u(t)$ in (2) caused by logic-induced variation, temporal variation and process variation.

3. Stochastic modeling

In this section, we first propose our stochastic current model for the current loads of the P/G network in Section 3.1, where we extract the correlation from the extensive simulation of the circuit and then apply ICA to get the parameterized model of the load current. Then in Section 3.2, based on the load current model, we propose the parameterized MNA formulation and mathematically represent the variation-aware decap budgeting problem.

3.1 Stochastic current modeling

In this section, we propose our stochastic current modeling for current loads of the P/G network, i.e., $u(t)$ in (2). Similar to the vectorless P/G analysis in (Kouroussis et al., 2005), we assume that the circuit is partitioned into blocks such that different blocks are relatively independent. For each block, there are multiple ports connected to the power network, and each port is modeled as a time-varying current load for the power network. We apply extensive simulation to each block *independently* to get the current signatures. Because

we ignore the interdependence between blocks, the obtained current signatures are still conservative compared with the real current profiles.

For simplicity of presentation and similar to (Su et al., 2003) [1], we represent the current in one clock cycle as a triangular waveform with rising time, falling time, and peak value $\hat{I}$. The peak values vary in different clock cycles and over different ports. The correlation between currents for different ports is called *logic-induced correlation*. In addition, the currents of the same port in different clock cycles are also correlated. We call this type of correlation as *temporal correlation*. For example, it might take a block several clock cycles to execute certain functions and the current profile inside those clock cycles are dependent to each other. We denote L, the *correlation length*, as the maximum number of clock cycles in which the peak currents might be correlated and can be decided from the simulation results.

In the following, we devise a stochastic model which can efficiently capture the correlation from both the logic-induced variation and temporal variation, as well as from process variation.

3.1.1 Stochastic model to consider current interdependence

We record the peak currents at port k ($1 \leq k \leq p$ with p as the total port number) at different clock cycles, and put them into vectors, i.e.,

$$b_k^j = [\hat{I}_k^j, \hat{I}_k^{1+j}, \ldots], \quad 1 \leq k \leq p, 1 \leq j \leq L \tag{7}$$

where $\hat{I}_k^i$ is the peak currents at port k in clock cycle i, and b_k^j is the set of peak currents sampled every clock cycles starting from cycle j. Properly truncation from the end of b_k^j is necessary to make them of the same length for further processing. In other words, the corresponding samples in vectors $b_k^{j_1}$ and $b_k^{j_2}$ are $|j_1 - j_2|$ clock cycles apart. If the peak current at port k in the first clock cycle is selected from the r-th element of b_k^1, then the peak current in the second clock cycle should be the r-th element of b_k^2. As an example, if the peak values in each clock cycle for port 1 are $[0.1, 0.2, 0.3, 0.4]$, and for port 2 are $[0.01, 0.02, 0.03, 0.04]$, and we choose $L = 2$, then

$$\begin{aligned} b_1^1 &= [0.1, 0.2, 0.3], \quad b_2^1 = [0.01, 0.02, 0.03], \\ b_1^2 &= [0.2, 0.3, 0.4], \quad b_2^2 = [0.02, 0.03, 0.04]. \end{aligned} \tag{8}$$

We model the peak current at each port as a stochastic process. Then all the elements of b_k^j are the samples for the stochastic variable $\mathcal{B}_k^j$ with its mean $\mu(\mathcal{B}_k^j)$ and standard deviation $\sigma(\mathcal{B}_k^j)$. We call the correlation between $b_k^{j_1}$ and $b_k^{j_2}$ as temporal correlation, and the one between $b_{k_1}^j$ and $b_{k_2}^j$ as logic-induced correlation.

With those stochastic variables B_k^j's and their corresponding samples b_k^j's, we can compute the logic-induced correlation matrix $\rho(j; k_1, k_2)$ which describes the correlation between the peak

[1] Our noise verification in the experiment part does not depend on this assumption.

currents at any two ports k_1 and k_2 in clock cycle j as

$$\rho(j;k_1,k_2) = \frac{cov(\mathcal{B}^j_{k_i}, \mathcal{B}^j_{k_2})}{\sigma(\mathcal{B}^j_{k_1})\sigma(\mathcal{B}^j_{k_2})}, \quad (1 \le k_1, k_2 \le p), \tag{9}$$

where $cov(\mathcal{B}^j_{k_1}, \mathcal{B}^j_{k_2})$ are the covariance between $\mathcal{B}^j_{k_1}$ and $\mathcal{B}^j_{k_2}$, and $\sigma(\mathcal{B}^j_{k_1})$ and $\sigma(\mathcal{B}^j_{k_2})$ are their standard deviations, respectively. Similarly, the temporal correlation matrix $\rho(j_1, j_2; k)$ which describes the correlation between the peak currents between clock cycles j_1 and j_2 of a same port k can be computed as

$$\rho(j_1,j_2;k) = \frac{cov(\mathcal{B}^{j_1}_k, \mathcal{B}^{j_2}_k)}{\sigma(\mathcal{B}^{j_1}_k)\sigma(\mathcal{B}^{j_2}_k)}, \quad (1 \le j_1, j_2 \le L). \tag{10}$$

3.1.2 Extension to process variation with spatial correlation

(Orshansky et al., 2002) relates the current to the process parameters L_{eff}, t_{ox} and V_t as

$$\hat{I}^i_k \sim L^{-0.5}_{eff} t^{-0.8}_{ox} (V_{dd} - V_t). \tag{11}$$

As pointed out in (Cao & Clark, 2005), in 90nm regime the most significant variation source is the effective channel length (L_{eff}), and L_{eff} variation can be more than 30%. Furthermore, L_{eff} variation is mostly spatially correlated but not random (Orshansky et al., 2002). Therefore, we will use L_{eff} variation as an example to show how process variation can be embedded into our stochastic modeling. It is understood that the process variation of other parameters can be dealt with in a similar way.

We use the variation model for L_{eff} based on (Orshansky et al., 2002):

$$L_{eff} = L_0 + L^{prox} + L^{spat} + \epsilon, \tag{12}$$

where L_0 is the overall mean, L^{prox} is a discrete stochastic variable with a distribution determined by the frequency of each gate, L^{spat} corresponds to the spatial variation, and ϵ is the local random variation.

From (11), with L_{eff} variation, the sample $\hat{I}^j_k$ becomes a set of samples

$$\left[\hat{I}^j_k \sqrt{\frac{\bar{L}_{eff,k}}{L^1_{eff,k}}}, \hat{I}^j_k \sqrt{\frac{\bar{L}_{eff,k}}{L^2_{eff,k}}}, \dots \right], \tag{13}$$

where $L^i_{eff,k}$ with different i are the samples of $L_{eff,k}$ for the circuit block corresponding to port k with the nominal value $\bar{L}_{eff,k}$, and $\hat{I}^j_k$ are the peak current sample for $\mathcal{B}^j_k$ in the deterministic case without L_{eff} variation in (7). In other words, if we have n samples for $L_{eff,k}$ $(L^1_{eff,k}, L^2_{eff,k}, \dots, L^n_{eff,k})$, then every current sample I^j_k becomes n samples. Therefore, the sample vector b^j_k becomes n times longer in the presence of L_{eff} variation, and we denote this new vector as $\tilde{b}^j_k$. In addition, we denote the stochastic variable representing the set of $\tilde{b}^j_k$

as $\tilde{\mathcal{B}}_k^j$. In this case, the temporal correlation (9) becomes

$$\tilde{\rho}(j;k_1,k_2) = \frac{cov(\tilde{\mathcal{B}}_{k_i}^j, \tilde{\mathcal{B}}_{k_2}^j)}{\sigma(\tilde{\mathcal{B}}_{k_1}^j)\sigma(\tilde{\mathcal{B}}_{k_2}^j)}, \quad (1 \le k_1, k_2 \le p), \tag{14}$$

and the logic-induced correlation (10) becomes

$$\tilde{\rho}(j_1,j_2;k) = \frac{cov(\tilde{\mathcal{B}}_k^{j_1}, \tilde{\mathcal{B}}_k^{j_2})}{\sigma(\tilde{\mathcal{B}}_k^{j_1})\sigma(\tilde{\mathcal{B}}_k^{j_2})}, \quad (1 \le j_1, j_2 \le L). \tag{15}$$

3.2 Parameterized problem formulation

3.2.1 Parameterized current via ICA

Directly considering the temporal and logic-induced correlation including process variation as formulated in (14) and (15) is difficult for optimization. Therefore, we propose to remove the correlation between $\tilde{B}_k^j$'s and build a parameterized current model in the following.

If all those variable $\tilde{B}_k^j$'s are Gaussian, we can apply principal component analysis (PCA) to remove the interdependence between the stochastic variables $\tilde{\mathcal{B}}_k^j$'s. However, this is not the case for our stochastic current model. Therefore, we use independent component analysis (ICA) that is applicable to non-Gaussian distribution (Hyvarinen et al., 2001). The input to ICA is the samples $\tilde{b}_k^j$ as well as their correlation matrices (14) and (15), and the output are a set of independent stochastic variables r_i and their corresponding coefficients $a_i(j,k)$ to reconstruct each $\tilde{\mathcal{B}}_k^j$, i.e.

$$\tilde{\mathcal{B}}_k^j \approx \sum_{i=1}^{q} a_i(j,k) r_i. \tag{16}$$

The order q is determined for each design such that the relative error between the original currents and model predicted currents is less than 5%. The probability density function (PDF) for each r_i is also given in the output of ICA as a one-dimensional lookup table, based on which we can bound the range of r_i as

$$\underline{r_i} \le r_i \le \overline{r_i}, \tag{17}$$

where $\underline{r_i}$ and $\overline{r_i}$ can be related to r_i's mean (μ) and variance (σ^2). For example, we can take $\underline{r_i}$ as $\mu - 4\sigma$ and $\overline{r_i}$ as $\mu + 4\sigma$.

Therefore, assuming uniform rising and falling times across the chip for the triangular current waveform within a clock cycle [2], together with $a_i(j,k)$ which represents the i-th component of the peak current at port k in clock cycle j, we have all the necessary information to obtain the i-th time-varying current waveform component $u_i(t;j,k)$. If we denote T as the clock period,

[2] This uniform assumption does not affect the results in our experiments.

then $jT \leq t \leq (j+1)T$. Put those $u_i(t;j,k)$ at all ports in clock cycle j together as

$$u_i(t;j) = \begin{pmatrix} u_i(t;j,1) \\ u_i(t;j,2) \\ \vdots \\ u_i(t;j,p) \end{pmatrix}, \quad jT \leq t \leq (j+1)T, \tag{18}$$

and then combine all the $u_i(t;j)$ in different clock cycles, we can get $u_i(t)$ with $0 \leq t \leq LT$. Finally, according to superposition theorem, we have

$$u(t) = \sum_{i=1}^{q} u_i(t) r_i, \quad 0 \leq t \leq LT. \tag{19}$$

We call (19) as parameterized current load model.

3.2.2 Parameterized MNA for decap budgeting

Considering the inherent parasitics, we model the decap similarly to (Zheng et al., 2003) as an equivalent series capacitor (ESC), and equivalent series resistor (ESR) and an equivalent series inductor (ESL). When a decap with size w_i is inserted into the power network at a given location, its impact can be considered by adjusting matrices G and C in (2) based on the location at the network and the size of the decap. Mathematically, it can be represented as

$$G = G_0 + \sum_{i=1}^{M} w_i \cdot G_{w,i}, \tag{20}$$

$$C = C_0 + \sum_{i=1}^{M} w_i \cdot C_{w,i}, \tag{21}$$

where G_0 and C_0 are the original matrices for the power network without decap, M is the total number of decaps, and $G_{w,i}$ and $C_{w,i}$ provide the stamping of a unit width decap at the i-th location. Due to the placement constraint, each w_i has an upper bound, i.e., we have the local constraints

$$0 \leq w_i \leq \overline{w_i}. \tag{22}$$

If only noise minimization is considered, then we can simply choose $w_i = \overline{w_i}$ $(\forall i)$, i.e., use up all the white space from the physical placement constraints. However, there are two other important issues we need to take into consideration: the leakage and the area overhead. With those two constraints, we cannot add too much decap, and therefore we have the global decap area constraint

$$\sum_{i=1}^{M} w_i \leq \overline{W}. \tag{23}$$

In practice we always have the following relationship between the local constraints (22) and the global constraint (23)

$$\sum_{i=1}^{M} \overline{w}_i \geq \overline{W}, \tag{24}$$

which implies that (23) is always tight for the optimization problem, and (22) is not tight for all i. In other words, we are given the total amount of decaps, and we want to allocate those decaps to the proper locations, so that the noise is minimized while there is no violation to (22).
The MNA equation of (2) with G given by (20), C given by (21), and u given by (19) can be written as follows

$$\begin{aligned}&(G_0 + \sum_{i=1}^{M} w_i \cdot G_{w,i})x + (C_0 + \sum_{i=1}^{M} w_i \cdot C_{w,i})\frac{dx}{dt}\\ &= B\sum_{i=1}^{q} u_i(t) r_i,\end{aligned} \tag{25}$$

where $0 \le t \le LT$ and r_i is a stochastic variable with $\underline{r_i} \le r_i \le \overline{r_i}$. We call this MNA equation as *parameterized MNA formulation* for decap budgeting. One of the major advantages in using this parameterized MNA formulation is that it enables us to implicitly compute sensitivities efficiently and accurately, which will become clearer in the later part of this chapter.
With the parameterized MNA, the variation-aware decap budgeting problem can be mathematically represented as follows:

$$(\mathbf{P1}) \quad \min_{w_i} \ \sup_{r_k} f = \sum_{i=1}^{p} \int_{\Omega_i} (\overline{U} - y_i(w_i, r_k; t))dt \tag{26}$$

$$s.t. \ \ \underline{r_k} \le r_k \le \overline{r_k} \quad 1 \le k \le q, \tag{27}$$

$$0 \le w_i \le \overline{w_i}, \quad 1 \le i \le M \tag{28}$$

$$\sum_{i=1}^{M} w_i \le \overline{W}, \tag{29}$$

where voltage y_i is a function of w_i, r_k, and time t and can be solved from (25) and (3).
Problem (**P1**) is a constrained min-max optimization problem. The *sup* operation over all random variables r_k is to find the worst-case noise violation measures for a given power network design. This operation guarantees that all P/G network designs satisfy the given design constrains while considering the temporal and logic-induced correlations as well as L_{eff} variation among ports. This is of particular use for ASIC-style designs, where the worst-case design performance has to be ensured for sign-off. The *min* operation over all decap sizes w_i is to find the optimal decap budgeting solution so that the worst-case noise violation is minimized.

4. Algorithms

In this section ,we present our iterative alternative programming approach to solve the problem (**P1**) stated in Section 3. In Section 4.1, we decompose the original min-max problem into two alternative optimization sub-problems, which are solved in Section 4.2 by an efficient sequential programming approach based. The detailed algorithm to compute sensitivities from parameterized MNA for such sequential programming is zoomed into detail in Section 4.3.

4.1 Iterative alternative programming with guaranteed convergence

Because there exists no general technique to solve the constrained min-max problem (**P1**) optimally, we resort to an effective iterative optimization strategy, which we call *iterative alternative programming* (IAP). That is, instead of solving the min-max problem (**P1**) directly, we solve it by iteratively solving the following two sub-problems alternatively.

The first sub-problem assumes that all decaps' sizes w_i are known, hence the worst-case noise can be obtained by solving the following optimization problem

$$(\mathbf{P2}) \quad \max_{r_k} \; f = \sum_{i=1}^{p} \int_{\Omega_i} (\overline{U} - y_i(w_i, r_k; t))dt \tag{30}$$

$$s.t. \;\; \underline{r_k} \leq r_k \leq \overline{r_k}, \quad 1 \leq k \leq q \tag{31}$$

The second sub-problem assumes that all random variables r_k have fixed values, hence the decap sizes to achieve the minimum noise can be obtained by solving the following optimization problem

$$(\mathbf{P3}) \quad \min_{w_i} \; f = \sum_{i=1}^{p} \int_{\Omega_i} (\overline{U} - y_i(w_i, r_k; t))dt \tag{32}$$

$$s.t. \;\; 0 \leq w_i \leq \overline{w_i}, \quad 1 \leq i \leq M \tag{33}$$

$$\sum_{i=1}^{M} w_i \leq W, \tag{34}$$

where W is the total white space available. Problem (**P3**) is exactly the deterministic version of the original problem formulation (**P1**).

We illustrate our idea in Fig. 1 and the overall algorithm can be described in Algorithm 1, where *iter* is the current iteration number and ϵ determines the stop criteria of the optimization procedure. For each iteration, we increase the total available white space by ΔW until $\bar{W}$.

The algorithm terminates when the change of objective function $|\Delta f|$ is sufficiently small indicating the convergence of the solution, or we have reached the global decap constraint (29). The first case corresponding to the situation where we have reduced noise below the bound before all the white space are used up, while the second case indicates that we have reached the global decap area constraint. In either case, the algorithm will terminates and the convergence of our algorithm is guaranteed as long as the algorithms for solving (**P2**) and (**P3**) converge, which will be discussed shortly. As shown in Fig. 2, the choice of ΔW reflects a tradeoff between the runtime and the solution quality. Smaller ΔW can result in smaller noise under the same decap area but the runtime is increased as well. Setting $\Delta W = 0.004W$ gives a good balance in our experiment.

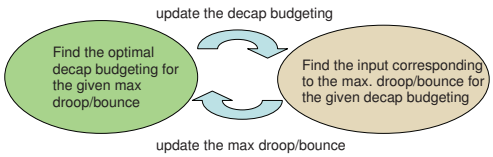

Fig. 1. Solve the min-max problem by iteratively solving two sub-problems.

Algorithm 1 Iterative alternative programming.

INPUT: initial guess w_i, r_k, current white space $\bar{W}$;
OUTPUT: final solution w_i to problem (**P1**);
Initialize: The current white space available $W = 0$;
for $iter = 0$; $|\Delta f| \leq \epsilon$ and W $\leq \bar{W}$; $iter++$ **do**
  $W = W + \Delta W$;
  w_i = solve-P3($iter$, w_i, r_k, W);
  r_k = solve-P2($iter$, w_i, r_k, W);
  Compute objective function with new r_k and w_i;
end for

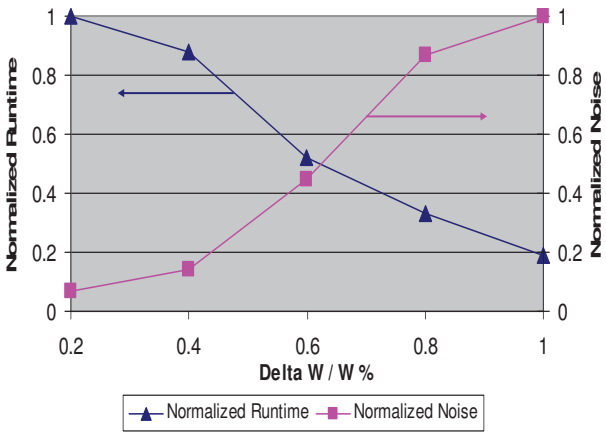

Fig. 2. The normalized runtime and noise w.r.t different $\frac{\Delta W}{\bar{W}}$.

4.2 Efficient sequential programming

Both problems (**P2**) or (**P3**) are constrained nonlinear optimization problems, and there exits no general technique to solve them efficiently. Because the constraints in both problems are linear, if we can approximate the objective function f by a first-order linear function, the original problems would become linear programming (LP). Or if we can approximate the objective function f by a second-order quadratic function, they would become a quadratic programming (QP) problem. Because efficient solvers exist for both LP and QP problems, we can solve the approximated problems more efficiently than solving the original problems directly. Therefore, we propose to solve the original (**P2**) or (**P3**) problem via sequential programming, either through LP or QP in the following.

For now, let us assume that we know how to compute the first- and second-order sensitivities of the objective function f with respect to changing variables, which will be discussed in Section 4.3. Therefore, we can easily obtain the linear and quadratic approximations of the objective function. For example, for the objective function in problem (**P3**), the changing variables are all Δw_i. Therefore, we have the following linear and quadratic approximations for the objective function

$$f_{lp} \approx f_0 + \sum_{i=1}^{M} \frac{\partial f}{\partial w_i} \Delta w_i, \tag{35}$$

$$f_{qp} \approx f_0 + \sum_{i=1}^{M} \frac{\partial f}{\partial w_i} \Delta w_i + \sum_{k=1}^{M} \sum_{j=1}^{M} \frac{\partial^2 f}{\partial w_i \partial w_j} \Delta w_i \Delta w_j, \tag{36}$$

where f_0 is the current value of the objective function, and $\frac{\partial f}{\partial w_i}$ and $\frac{\partial^2 f}{\partial w_i \partial w_j}$ are the first- and second-order sensitivities of f, respectively.

Apparently, (35) is a linear function of Δw_i, while (36) is a quadratic function of changing variables Δw_i. By replacing (30) with (35), we obtain an approximated LP formulation for (**P3**). Or by replacing (30) with (36), we obtain an approximated QP formulation for (**P3**). Both LP and QP can be solved efficiently.

A high-level description of the sequential programming algorithm to solve either problem (**P2**) or (**P3**) is shown in Algorithm 2, where *iter2* is the current iteration number, *ITER2* is the maximum iteration bound. The iterations stop when the change of objective function $|\Delta f|$ is smaller than ϵ_2, which is dynamically adjusted according to the iteration number *iter* in the outer-loop of Algorithm 1. We employ an exponential decreasing function to adjust ϵ_2 in this work. The idea is that when the out-loop iteration is small (or we are far from the optimal solution), we can have an early termination of the inner-loop optimization procedure as shown in Algorithm 2 early. But when the outer-loop iteration becomes large enough (or we are close to the optimal solution), we should spend more time in each inner-loop optimization to find a better global optimal solution. Parameter η is used to control the efforts that we should spend in the inner-loop's optimization.

The convergence for Algorithm 2 is guaranteed by noting that though the iterations the objective function is monotonically decreasing, and thus the loop must exit when a local or global minimum/maximum is obtained.

Algorithm 2 Sequential programming (sLP or sQP) for solving (**P2**) and (**P3**).

INPUT: *iter*, w_i, r_i, W;
OUTPUT: updated w_i for (**P3**) or r_i for (**P2**);
ϵ_2 = exp(-η·iter);
for *iter*2=0; $|\Delta f| \leq \epsilon_2$ or *iter*2 $\leq$ *ITER*2; *iter*2++ **do**
  Compute the first- (and second-order) sensitivities of f;
  Formulate (**P2**) or (**P3**) as an LP (or QP) problem;
  Call LP (or QP) solver to solve the above problem;
  Compute objective function with new w_i (**P2**) or r_i (**P3**);
end for

Even though we can solve problem (**P2**) and (**P3**) via either sequential LP or QP programming (sLP or sQP) as shown in Algorithm 2, there are several differences between these two approaches. If we approximate the problem as an sLP, at each optimization iteration we can find a guaranteed local optimal solution because of the convexity of LP formulation. But because of the relatively poor first-order approximation quality, we may not find a good final solution at the end. In contrast, if we approximate the problem as an sQP, the approximation quality is improved because of the use of higher-order sensitivity information. And each optimization iteration works more like a Newton step for solving convex optimization problems. Thus we may find a better final solution compared to the

first-order LP approximation. Our experimental results will show that, in practice, sQP solutions are indeed better than sLP's for large test cases. Of course we notice that the QP formulation at each iteration is not necessarily convex, as we cannot prove that the Hessian of (36) is always positive semidefinite. In practice, however, we find that the solution quality from sQP is high.

For practical use, the number of variables for the sLP or sQP can be huge. Luckily, promising research results have been presented which show that by fully utilizing partitioning, parallel computing and efficient data compression, problems with millions of variables and thousands of constraints can be solved within several hundred seconds (Andersen & Anderson, 1998; Karypis et al., 1994).

4.3 Sensitivity computation

To solve (**P2**) and (**P3**) via sLP or sQP, we need to compute the sensitivities of the objective function f with respect to the design variables, i.e., either w_i or r_i. Because this computation is similar for both (**P2**) and (**P3**), we will focus our discussion on (**P3**) in the following.

The first- and second-order sensitivities of the objective function f of problem (**P3**) are defined as

$$\frac{\partial f}{\partial w_i} = -\sum_{i=1}^{p} \int_{\Omega_i} \frac{\partial y_i}{\partial w_i} dt = -\sum_{i=1}^{p} \int_{\Omega_i} L_{0i}^T \frac{\partial x}{\partial w_i} dt, \tag{37}$$

$$\frac{\partial^2 f}{\partial w_i \partial w_j} = -\sum_{i=1}^{p} \int_{\Omega_i} \frac{\partial^2 y_i}{\partial w_i \partial w_j} dt = -\sum_{i=1}^{p} \int_{\Omega_i} L_{0i}^T \frac{\partial^2 x}{\partial w_i \partial w_j} dt. \tag{38}$$

For simplicity of presentation, we have loosely applied the derivative notation on a vector for component-wise derivative.

To compute the sensitivity of f w.r.t. w_i, all we need to know is the sensitivity of the state vector x with respect to w_i. We use Taylor expansion to express x as follows

$$x = x_0 + \sum_{i=1}^{M} \alpha_i \Delta w_i + \sum_{i=1}^{M} \sum_{j=i}^{M} \beta_{ij} \cdot \Delta w_i \Delta w_j + \ldots, \tag{39}$$

where α_i is the first-order sensitivity of x w.r.t. random variable w_i, and β_{ij} is the second-order sensitivity of x with respect to random variable w_i and w_j. In other words, we have

$$\frac{\partial x}{\partial w_i} = \alpha_i, \quad \frac{\partial^2 x}{\partial w_i \partial w_j} = \beta_{ij}. \tag{40}$$

To compute these sensitivities, we recognize that x also satisfies the differential equation given by the parameterized MNA formulation (25). By Laplace transformation, we re-write (2) as follows

$$(G + \sum_{i=1}^{M} \Delta w_i \cdot G_{w,i})x + s(C + \sum_{i=1}^{M} \Delta w_i \cdot C_{w,i})x = Bu. \tag{41}$$

By plugging (39) into (41), we obtain terms of Δw_i with different orders. By equating the zero-order terms of Δw_i from both left and right hand sides in (41), we obtain a set of equations

as follows

$$(G + sC)x_0 = Bu. \tag{42}$$

By equating the first-order terms of Δw_i, we obtain sets of equations as follows for all $1 \leq i \leq M$

$$(G + sC)\alpha_i = -(G_{w,i} + sC_{w,i})x_0. \tag{43}$$

Similarly, by equating the second-order terms of $\Delta w_i \Delta w_j$, we obtain another sets of equations as follows for all $1 \leq i \leq M$

$$(G + sC)\beta_{ij} = -(G_{w,i} + sC_{w,i})\alpha_j - (G_{w,j} + sC_{w,j})\alpha_i \tag{44}$$

By applying the Backward Euler integration formula and assuming the time step as h, we can re-write (42) and (43) as follows

$$(G + \frac{C}{h})x_0(t+h) = Bu(t+h) + x_0(t)\frac{C}{h}, \tag{45}$$

$$\begin{aligned}(G + \frac{C}{h})\alpha_i(t+h) = &-(G_{w,i} + \frac{C_{w,i}}{h})x_0(t+h) \\ &+ \frac{x_0(t)C_{w,i} + \alpha_i(t)C}{h},\end{aligned} \tag{46}$$

$$\begin{aligned}(G + \frac{C}{h})\beta_{ij}(t+h) = &-(G_{w,i} + \frac{C_{w,i}}{h})\alpha_j(t+h) \\ &-(G_{w,j} + \frac{C_{w,j}}{h})\alpha_i(t+h) \\ &+ \frac{\alpha_j(t)C_{w,i} + \alpha_i(t)C_{w,j} + \beta_{ij}(t)C}{h}.\end{aligned} \tag{47}$$

Because all equations in (45) and (46) share the same left-hand side matrix, $(G + C/h)$, we only need to perform LU-factorization once, and then reuse the same factorization to solve for x_0, α_i and β_{ij} sequentially at each time step. This computation is efficient because it only involves some matrix-vector multiplications, and backward and forward substitutions.
The integral interval Ω_i for port i is decided by x_0. Once x_0 is solved, we have $y = L_0^T x_0$, and then the corresponding interval can be decided from (5). By doing so we have assumed that the incremental δw_i is relatively small in each step and will not significantly influence the integral interval. In summary, we can compute the first and second-order sensitivities of the objective function f of problem (**P3**) by following the Algorithm 3.

5. Experimental results

In this section, we present experiments using four industrial P/G network designs. For each benchmark, we randomly select 20% of total nodes as candidate nodes for decap insertion, i.e., $M = 20\%N$. For fair comparison, when comparing the runtime and noise, the same white space is used up for different methods. We run experiments on a LINUX workstation with Pentium IV 2.66G CPU and 1G RAM. We partition the circuits according to the method in (Kouroussis et al., 2005). We use the package FASTICA (Hyvarinen & Oja,

Algorithm 3 Sensitivity computation for (**P3**).

INPUT: w_i, r_k, h, T;
OUTPUT: f, α_i and β_{ij};
factorization: LU factorize $G + C/h$;
for $t = 0; t + h \leq T; t = t + h$ **do**
  Solve (45) for $x_0(t + h)$;
end for
for $i = 1; i \leq p; i + +$ **do**
  Use (5) to compute Ω_i from $y(t) = L_0^T x_0(t)$;
end for
for $t = 0; t + h \leq T; t = t + h$ **do**
  Solve (46) for $\alpha_i(t + h)$;
  Solve $\frac{\partial f}{\partial w_i}$ from (37);
end for
for $t = 0; t + h \leq T; t = t + h$ **do**
  for $1 \leq i \leq K$ **do**
    for $1 \leq j \leq K$ **do**
      Solve (47) for $\beta_{ij}(t + h)$;
      Solve $\frac{\partial^2 f}{\partial w_i \partial w_j}$ from (38);
    end for
  end for
end for

1997) to perform ICA. Finally, we use MOSEK as the linear/quadratic programming solver (http://www.mosek.com, n.d.) and random walk based simulator (Qian et al., 2005) with detailed (not triangular) input current waveform to obtain the noise reported in this section.

5.1 Decap budgeting with operation variation

We compare three current models as shown in Table 1: maximum current peaks at all ports[3] (model 1), stochastic model (model 2) with logic-induced correlation only ($L = 1$), and stochastic model (model 3) with both logic-induced and temporal correlation. For temporal correlation, we always use $L = 4$ since all circuits tested take at most four clock cycles to complete any one instruction. Table 1 reports the noise and runtime for the four benchmarks with different number of nodes at the same decap area. Compared with the baseline model with maximum current peaks at all ports [4], the model considering temporal correlation reduces noise by up to 5×; and the model considering both temporal and logic-induced correlations reduces noise by up to 17× (see bold in Table 1). This is because the first two models cannot model the currents effectively and lead to inserting unnecessarily large decaps in some regions. As for the runtime, model 2 needs about 1.5× more time than model 1, while model 3 needs about 2.3× more. The runtime overhead is the price we have to pay in order to achieve better designs.

In Fig. 3, we plot the time-domain responses at one randomly selected port for two optimization iterations by alternatively solving the problem (**P3**) and (**P2**). The benchmark has 1284 nodes. The initial waveform is denoted by "A0:initial". After performing decap sizing once by solving problem (**P3**) with a fixed choice of random variables r_k, we obtain the

[3] We still use the detailed waveforms for the currents, except that the maximum values of those waveforms are always set to be the worst case values.
[4] We solve it by iteratively solving (**P3**) without altering to (**P2**).

Model 1	maximum current peaks at all ports
Model 2	stochastic model with logic-induced correlation
Model 3	Model 2 + temporal correlation

Node #	Port #	noise (V*s)			runtime (s)		
		model 1	model 2	model 3	model 1	model 2	model 3
1284	**426**	**6.33e-7**	**1.28e-7**	**4.10e-8**	**104.2**	**161.2**	**282.3**
10490	3398	5.21e-5	1.09e-5	4.80e-6	973.2	1430	2199
42280	13327	7.92e-4	5.38e-4	9.13e-5	2732	3823	5238
166380	42146	1.34e-2	5.37e-3	2.28e-3	3625	5798	7821
avg		1	1/3×	1/9×	1	1.50×	2.26×

Table 1. Noise, runtime and area comparison between the three models.

new waveform as denoted by "A1:(**P3**)". We then switch to solve problem (**P2**) by varying the values of those random variables r_k, but with fixed decap sizes w_i. We see that the waveform of the final worst-case voltage drop becomes worse compared to the deterministic solution; hence we obtain a new voltage drop waveform as denoted by "A2:(**P2**)". We then switch back to solve the decap sizing problem (**P3**) with fixed but newly updated choice of random variables r_k. At the end of this optimization, we arrive at a new voltage waveform as denoted by "A3:(**P3**)". Apparently, compared to "A1:(**P3**)", the new solution has smaller voltage drop. If we continue the same procedures by following the IAP algorithm given in Fig. 1, similar sequences of time domain voltage drop waveforms would repeat as we have shown in Fig. (3) until we converge to an optimal solution. Also, The voltage drop is reduced mostly in the first optimization iteration denoted as "A1:(**P3**)". Afterward, the voltage drop reduction is relatively small. This observation is in agreement with the common knowledge about any sensitivity-based optimization techniques. In this particular example, we find that the first two iterations reduces the noise by 51.4%.

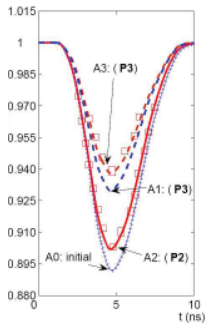

Fig. 3. Time domain waveforms at one port after sLP for different iterations.

5.2 L_{eff} variation aware decap budgeting

In the presence of process variation, we want to minimize the worst-case noise for L_{eff} variation. We solve this via the proposed IAP technique in Algorithm 1. We denote our L_{eff} variation aware approach as $sLP + L_{eff}$ and the counterpart as sLP. Before we quantitatively compare the two methods, we first use Fig. 4 to demonstrate the effectiveness of L_{eff} variation aware decap budgeting. We use the same circuit with 15% L_{eff} variation and perform Monte Carlo simulations with 14000 samples to obtain the noise histogram of the design from the sLP

Node #	Port #	sLP			$sLP + L_{eff}$		
		μ (V*s)	3σ (V*s)	RT (s)	μ (V*s)	3σ (V*s)	RT (s)
1284	426	9.28e-7	3.97e-7	184.2	6.14e-7	1.38e-7	332.8 (1.81×)
10490	3398	1.03e-4	4.79e-5	1121	7.22e-5	1.23e-5	3429 (3.06×)
42280	13327	2.29e-3	9.72e-4	2236	8.23e-4	1.01e-4	6924 (3.10×)
166380	**42146**	**2.06e-2**	**9.91e-3**	**3824**	**5.31e-3**	**8.32e-4**	**11224** (2.93×)
avg		1	1	1	0.50×	0.20×	2.73×

Table 2. The mean value μ, 3σ variance of the noise and runtime (RT) comparison between $sLP + L_{eff}$ and sLP with 10% intra-die L_{eff} variation.

and $sLP + L_{eff}$, respectively. From the figure we can see that the noise from $sLP + L_{eff}$ (mean value 8.4×10^{-9} V*s, 3σ value 0.4×10^{-9} V*s) is much smaller than that from sLP (mean value 9.7×10^{-9} V*s, 3σ value 1.9×10^{-9} V*s), although both have the same decap area constraints.

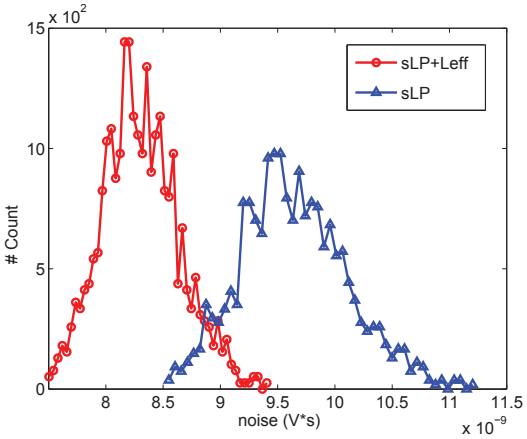

Fig. 4. The noise distribution for the an industry power mesh with decap budgeting using sLP and $sLP + Leff$.

Next we compare the mean value μ and 3σ value of the noise distribution with 10% L_{eff} variation based on Monte Carlo simulation with 10,000 runs, and the results are reported in Table 2. Compared with using deterministic L_{eff}, considering L_{eff} variation reduces the mean noise by up to 4× and 3σ noise by up to 13× (see bold in Table 2), when both applying the current model with temporal and logic-induced correlations. As for the runtime between sLP and $sLP + L_{eff}$, the latter needs about 2.7× more time than the former on average.

5.3 Comparison between sLP and sQP

We study the difference between our sLP and sQP approaches in terms of noise and runtime for five benchmarks with different number of nodes in Table 3 for deterministic case. The same white space are used up for both methods. An interesting observation from Table3 is that sQP almost always obtain smaller noise than sLP, particularly for those large test cases, with longer runtime. This is expected, as higher-order sensitivities are used in sQP to guide the optimization. In terms of noise, sQP is much better than sLP for large test cases and slightly worse for the small test case. In terms of runtime, however, sLP is on average 3.25× faster than sQP. Similar experimental results are presented in Table 4 in the presence of Leff variation. We can see that not only the mean noise is reduced by 19%, the 3σ valude is also

Node #	Port #	sLP		sQP	
		noise (V*s)	time (s)	noise (V*s)	time (s)
128	41	1.83e-9	2.4	1.85e-9 (1.01×)	8.3 (3.46×)
512	174	1.83e-9	23.8	1.81e-9 (0.99×)	66.0 (2.77×)
1280	477	1.85e-9	151	1.79e-9 (0.97×)	497 (3.29×)
5120	1731	1.91e-8	982	1.30e-8 (0.68×)	3779 (3.85×)
12800	3324	1.94e-4	1960	0.81e-4 (0.42×)	5658 (2.89×)
Avg		1	1	0.81×	3.25×

Table 3. Noise and runtime comparison between sLP and sQP.

Node #	Port #	$sLP + L_{eff}$			$sQP + L_{eff}$		
		μ (V*s)	3σ (V*s)	RT (s)	μ (V*s)	3σ (V*s)	RT (s)
1284	426	6.14e-7	1.38e-7	332.8)	4.98e-7	7.70e-8	985.0 (2.96×)
10490	3398	7.22e-5	1.23e-5	3429)	5.91e-5	5.28e-5	11932.9 (3.48×)
42280	13327	8.23e-4	1.01e-4	6924)	6.77e-4	5.93e-5	18348.6 (2.65×)
166380	**42146**	**5.31e-3**	**8.32e-4**	**11224**	**4.11e-3**	**4.71e-4**	**36365.8** (3.24×)
avg		1	1	1	0.81×	0.54×	3.08×

Table 4. The mean value μ, 3σ variance of the noise and runtime (RT) comparison between $sLP + L_{eff}$ and $sQP + L_{eff}$ with 10% intra-die L_{eff} variation.

reduced by 46%. We believe both sLP and sQP are of practical value, and they provide good trade-off between runtime efficiency and design quality. Note that no existing approach in the literature leverages them for decap budgeting. Our sLP/sQP solution is the first of the kind.

6. Conclusions and discussions

This chapter studied a variation-aware decoupling capacitance (decap) budgeting problem for reliable power network design. The major contributions of this work are two-fold: (1) a novel method to solve the the deterministic decap budgeting problem efficiently; and (2) a new variation-aware decap budgeting problem that takes into account process variation effects. Experimental results show that compared to existing industrial quality decap budgeting techniques as proposed in the literature, we achieve 13× speed-up while achieving similar design quality. It also serves as an example for general stochastic optimization.

7. References

Agarwal, K. & Nassif, S. (2007). Characterizing Process Variation in Nanometer CMOS, *IEEE/ACM DAC*.

Andersen, E. D. & Anderson, K. D. (1998). A parallel interior-point algorithm for linear programming on a shared memory machine, *CORE Discussion Paper 9808*.

Bhardwaj, S. & Vrudhula, S. B. K. (2005). Leakage Minimization of Nano-scale Circuits in the Presence of Systematic and Random Variations, *IEEE/ACM DAC*.

Cao, Y. & Clark, L. T. (2005). Mapping statisitical process variations toward circuit performance variability: An analytical modeling approach, *IEEE/ACM DAC*.

Chang, H. & Sapatnekar, S. S. (2003). Statistical timing analysis considering spatial correlations using a single PERT-like traversal, *IEEE/ACM ICCAD*, pp. 621 – 625.

Davoodi, A. & Srivastava, A. (2006). ąśVariability-Driven Gate Sizin for Binning Yield Optimization, *IEEE/ACM DAC*.

Ferzli, I. A. & Najm, F. N. (2003). Statistical verification of power grids considering process-induced leakage current variations, *IEEE/ACM ICCAD*.

Fu, J., Luo, Z., Hong, X., Cai, Y., Tan, S.-D. & Pan, Z. (2004). A fast decoupling capacitor budgeting algorithm for robust on-chip power delivery, *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pp. 505–510.

Ghanta, P., Vrudhula, S., Panda, R. & Wang, J. (2005). Stochastic power grid analysis considering process variations, *Proc. European Design and Test Conf. (DATE)*, Vol. 2, pp. 964–969.

He, L., Kahng, A., Tam, K. H. & Xiong, J. (2007). Simultaneous Buffer Insertion and Wire Sizing Considering Systematic CMP Variation and Random Leff Variation, *IEEE Trans. on CAD* .

http://www.mosek.com (n.d.).

Hyvarinen, A., Karhunen, J. & Oja, E. (2001). *Independent Component Analysis*, John Wiley & Sons.

Hyvarinen, A. & Oja, E. (1997). A Fast Fixed-Point Algorithm for Independent Component Analysis, *Neural Computation* .

Karypis, G., Gupta, A. & Kumar, V. (1994). A Parallel Formulation of Interior Point Algorithms, *ACM/IEEE Conference on High Performance Networking and Computing*.

Kouroussis, D., Ferzli, I. A. & Najm, F. N. (2005). Incremental partitioning-based vectorless power grid verification, *IEEE/ACM ICCAD*.

Li, H., Qi, Z., Tan, S. X.-D., Wu, L., Cai, Y. & Hong, X. (2005). Partitioning-based approach to fast on-chip decap budgeting and minimization, *IEEE/ACM DAC*, pp. 170–175.

Mani, M., Devgan, A. & Orshansky, M. (2005). An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints, *IEEE/ACM DAC*.

Mani, M., Singh, A. & Orshansky, M. (2006). Joint Design-Time and Post-Silicon Minimization of Parametric Yield Loss using Adjustable Robust Optimization, *IEEE/ACM ICCAD*.

Orshansky, M., Milor, L., Chen, P., Keutzer, K. & Hu, C. (2002). Impact of Spatial Intrachip Gate Length Variability on the Performance of High-speed Digital Circuits, *IEEE Trans. on CAD* .

Pant, S., Blaauw, D., Zolotov, V., Sundareswaran, S. & Panda, R. (2004). A stochastic approach to power grid analysis, *IEEE/ACM DAC*, pp. 171–176.

Pant, S., D.Blaauw, Zolotov, V., S.Sundareswaran & Panda, R. (2004). A stochastic approach to power grid analysis, *IEEE/ACM DAC*.

Qian, H., Nassif, S. R. & Sapatnekar, S. S. (2005). Power Grid Analysis Using Random Walks, *IEEE Trans. on CAD* .

Shi, Y., Xiong, J., Liu, C. C. & He, L. (2007). Efficient Decoupling Capacitance Budgeting Considering Operation and Process Variations, *IEEE/ACM ICCAD*.

Sinha, D., Shenoy, N. V. & Zhou, H. (2005). Statistical Gate Sizing for Timing Yield Optimization, *IEEE/ACM ICCAD*.

Su, H., Sapatnekar, S. S. & Nassif, S. R. (2003). Optimal decoupling capacitor sizing and placement for standard-cell layout designs, *IEEE Trans. on CAD* **22**: 428–436.

Visweswariah, C., Haring, R. A. & Conn, A. R. (2000). Noise Consierations in Circuit Optimization, *IEEE Trans. on CAD* .

Visweswariah, C., Ravindran, K., Kalafala, K., Walker, S. & Narayan, S. (2004). First-order incremental block-based statistical timing analysis, *IEEE/ACM DAC*.

Zhao, M., Panda, R., Sundareswaran, S., Yan, S. & Fu, Y. (2006). A fast on-chip decoupling capacitance budgeting algorithm using macromodeling and linear programming, *IEEE/ACM DAC*.

Zheng, H., Krauter, B. & Pileggi, L. (2003). On-Package Decoupling Optimization with Package Macromodels, *Proc. IEEE Custom Integrated Circuits Conference (CICC)*.