Hierarchical Cross-Entropy Optimization for Fast On-Chip Decap Budgeting

Xueqian Zhao, Yonghe Guo, Xiaodao Chen, *Student Member, IEEE,* Zhuo Feng, *Member, IEEE,* and Shiyan Hu, *Senior Member, IEEE*

***Abstract*—Decoupling capacitor (decap) has been widely used to effectively reduce dynamic power supply noise. Traditional decap budgeting algorithms usually explore the sensitivity-based nonlinear optimizations or conjugate gradient (CG) methods, which can be prohibitively expensive for large-scale decap budgeting problems and cannot be easily parallelized. In this paper, we propose a hierarchical cross-entropy based optimization technique which is more efficient and parallel-friendly. Cross-entropy (CE) is an advanced optimization framework which explores the power of rare event probability theory and importance sampling. To achieve the high efficiency, a sensitivity-guided cross-entropy (SCE) algorithm is introduced which integrates CE with a partitioning-based sampling strategy to effectively reduce the solution space in solving the large-scale decap budgeting problems. Compared to improved CG method and conventional CE method, SCE with Latin hypercube sampling method (SCE-LHS) can provide 2× speedups, while achieving up to 25% improvement on power supply noise. To further improve decap optimization solution quality, SCE with sequential importance sampling (SCE-SIS) method is also studied and implemented. Compared to SCE-LHS, in similar runtime, SCE-SIS can lead to 16.8% further reduction on the total power supply noise.**

***Index Terms*—Adjoint sensitivity analysis, cross-entropy optimization, decoupling capacitor budgeting, power grid design, power supply noise.**

I. Introduction

With the dramatic frequency and supply voltage scaling, designing reliable on-chip power supply network becomes increasingly critical and challenging for nano-scale integrated circuit design. To mitigate the risks brought by excessive power supply noise, a variety of techniques has been proposed in the past [1]–[5], which typically falls into the following two categories: power grid wire sizing and decoupling capacitor (decap) optimization/budgeting. With the aggressive technology scaling, decap budgeting methods are more popular. Decap can utilize extra on-chip area to effectively suppress the dynamic power supply noise. Since on-chip area can be quite limited, achieving the optimal decap budgeting is challenging.

Manuscript received March 25, 2011; accepted June 10, 2011. Date of current version October 19, 2011. This paper was recommended by Associate Editor A. Elfadel.

The authors are with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA (e-mail: xueqianz@mtu.edu; yongheg@mtu.edu; cxiaodao@mtu.edu; zhuofeng@mtu.edu; shiyan@mtu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2011.2162068

Cross-entropy (CE) method [6] is an advanced parallel-friendly stochastic optimization technique exploiting the power of rare event probability theory and importance sampling. The CE method is suitable to tackle various combinatorial and continuous multi-modal optimizations. For instance, CE has been successfully applied to network reliability analysis and telecommunication system performance analysis. It has also been applied to solve combinatorial optimization problems [7]–[9]. However, to the best of our knowledge, none of the existing CE applications are related to integrated circuit design and optimizations.

Standard CE method is an iterative method. In each CE iteration, there are two key steps: 1) a random data sample generation step that is based on a specified mechanism, and 2) a parameter updating step that is used to generate "better" samples for the next CE iteration. The second step involves minimizing the cross-entropy or Kullback–Leibler divergence. Our experience indicates that a direct application of the standard CE method for decap budgeting optimization is prohibitively time consuming, which is due to the large number of required simulation samples.

In this paper, to gain high efficiency, we propose a hierarchical sensitivity-guided cross-entropy based (SCE) method to solve the decap budgeting optimization problem. Our SCE-based decap budgeting algorithm first partitions the large power grid circuit into a number of circuit blocks and identifies all the candidate blocks that may have various numbers of candidate nodes. Next, a block level decap budgeting can be accomplished by using the traditional CE optimization algorithm. In the following step, for each candidate block, fine-grained node level decap budgeting can be determined based on the information of relative decap sensitivities that can be obtained from one-time adjoint sensitivity analysis performed at the very beginning of our SCE algorithm. The proposed SCE algorithm can significantly improve the decap budgeting optimization efficiency by significantly reducing the number of simulation samples compared to prior approaches. More importantly, the SCE optimization procedure can be effectively and conveniently accelerated on nowadays multi-/many-core computing platforms. In the end, we show how to integrate a sequential importance sampling (SIS) method [10] in our SCE optimization framework to further improve the sampling effectiveness during each SCE iteration.

The outline of this paper is as follows. In Section II, we briefly review the background of decap budgeting problem

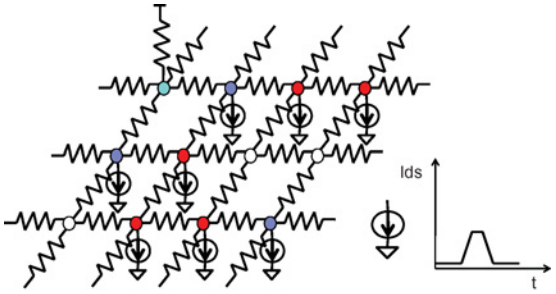


Fig. 1. Power supply network.

and power grid analysis methods. In Section III, the theoretical foundation of CE method is described in detail. In Section IV, a parallelizable hierarchical SCE method is proposed to tackle general on-chip decap budgeting problems. In Section V, the SIS method is deployed to further improve the block level decap budgeting during each SCE iteration. The saliency of the proposed decap budgeting method is validated and demonstrated through various experiments for a set of industrial power grid benchmarks in Section VI. The conclusion is provided in Section VII.

II. Background

A. Power Grid Analysis

Power supply network is a dynamic system delivering power to all on-chip functional components via interconnected wires which can be modeled as resistors and energy-storage circuit components such as capacitors and inductors. Circuit transient analysis can be adopted to solve the dynamic system at multiple time points by replacing all functional components with excitation current sources which are used to model components operations, as shown in Fig. 1. Node voltage solution can be obtained by solving [11]

$$C\frac{dx(t)}{dt} + Gx(t) = b(t) \tag{1}$$

where G denotes an $n \times n$ matrix containing the conductance values of all interconnected resistors in the power grid, C is a symmetric $n \times n$ matrix including the capacitance values of all on-chip capacitors, $x(t)$ represents an $n \times 1$ vector including all the instantaneous node voltages at time point t, $b(t)$ is an $n \times 1$ vector containing instantaneous values of all the time-varying current sources at time point t, and n denotes the number of the voltage supply nodes in power delivery network.

The Backward Euler's method can be used to transform the above continuous linear dynamic system to an equivalent linear system at time step t

$$\left(G + \frac{C}{h}\right)x(t) = b(t) + \frac{C}{h}x(t-h) \tag{2}$$

where h denotes the time step size. Direct methods such as LU or Cholesky matrix factorizations, or iterative methods [12], [13] can be used for power grid transient analysis according to problem scales.

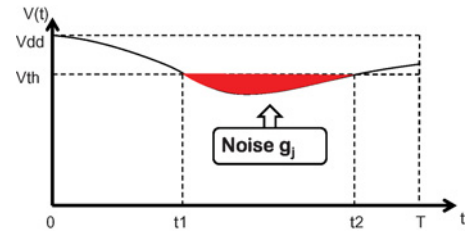


Fig. 2. Power supply noise.

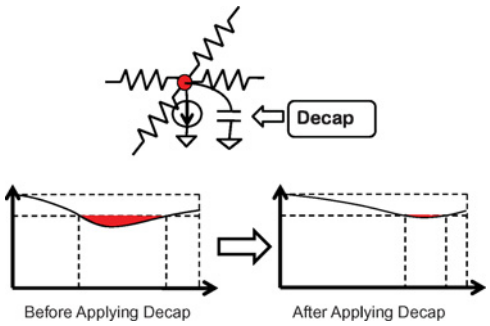


Fig. 3. Decap insertion for mitigating power supply noises.

B. Power Grid Noise and Sensitivity Analysis

When on-chip circuit components are drawing currents from the power grids, voltage drops along the interconnected wires can be observed which are mainly due to the resistive wires as well as di/dt effects. Excessive voltage variations that are greater than some threshold voltages (e.g., 10% of VDD voltage) may cause catastrophic circuit functionality failures. As shown in Fig. 2, power supply noise g_j at node j can be defined as the integration of the difference between transient voltage waveforms and voltage threshold V_{th} over the time period $[0, T]$ [2]

$$g_j(c_1, ..., c_m) = \int_0^T \max(V_{\text{th}} - v_j(t), 0)dt \tag{3}$$

where $v_j(t)$ denotes the instantaneous voltage response at node j, c_i represents the decap values applied at node i, and m is the number of candidate decap locations (nodes). In practice, the supply noise can be more properly defined as the average integration value over the time period T_{noise}

$$g_j(c_1, ..., c_m) = \frac{\int_0^T \max(V_{\text{th}} - v_j(t), 0)dt}{T} \tag{4}$$

which is a more useful metric for estimating the supply noises. For instance, we would like to distinguish the noise of 1 mV below threshold for 100 ns and the noise of 100 mV below threshold for 1 ns. It will be shown that our decap budgeting methods can easily handle the above noise metrics.

To reduce power supply noises efficiently using decaps, as shown in Fig. 3, we first need to obtain decap sensitivity defined as the effectiveness of a unit-size decap in mitigating the total supply noise

$$s_{i,j} = \frac{\partial g_j(c_1, ..., c_m)}{\partial c_i} \tag{5}$$

where s_{ij} denotes the decap sensitivity for supply noise g_j at node j considering the decap value c_i for node i.

However, (5) cannot be directly used to calculate the sensitivities of all decaps due to the extremely high simulation cost. Instead, the adjoint network sensitivity analysis method introduced in [2] and [14] can be adopted, in which the adjoint power delivery network can be constructed by removing all the previous excitation sources and attaching new current sources to the nodes of our interests. Then, the efficacy of decap at node i in removing noise at node j can be computed as follows:

$$s_{i,j} = \frac{\delta g_j}{\delta c_i} = \int_0^T V^*_{i,j}(T-t)\dot{v}_i(t)dt \tag{6}$$

where $V^*_{i,j}$ denotes the voltage waveform at node i under current excitation at node j in the adjoint power supply network, and $\dot{v}_i(t)$ denotes the time derivative of the voltage waveform at node i in the original power supply network [1]. Subsequently, the decap sensitivity for mitigating the total supply noise $s_{i,\text{all}}$ of all violated nodes with respect to the decap candidate c_i at node i can be computed as follows:

$$s_{i,\text{all}} = \sum_{j=1}^{n} s_{i,j} = \int_0^T V^*_{i,\text{all}}(T-t)\dot{v}_i(t)dt \tag{7}$$

where $V^*_{i,\text{all}}$ represents the voltage waveform at node i obtained from the adjoint power supply network with unit step current excitations applied at all violated nodes, and $\dot{v}_i(t)$ denotes the time derivative of the voltage waveform at node i in the original power supply network [1].

Furthermore, in this paper, graphic process units (GPUs) are adopted to accelerate both the original power grid transient analysis and the adjoint network transient analysis [13], [15], which can significantly reduce the simulation runtime by 20× compared to CPU-based simulations.

C. Objective for Decap Budgeting

In this paper, the decap budgeting optimization is defined as follows: given the limited decap budget, we try to obtain the optimal decap budgeting solution that will result in the minimum supply noises, which can be formulated as follows:

Objective function

$$\text{minimize} \quad G(c_1, c_2, \ldots c_m) = \sum_{j=1}^{n} g_j(c_1, c_2, \ldots c_m) \tag{8}$$

subject to

$$c_i \le C_{u_i}, \quad \text{and} \quad \sum_{i=1}^{m} c_i \le C_{\text{tot}} \tag{9}$$

where $G(c_1, c_2, \ldots c_m)$ denotes the total power supply noise that is the sum of noises at individual nodes g_j for $j = 1, ..., n$, n is the number of violated nodes, m represents the number of candidate decap locations, c_i denotes the decap value assigned to candidate node i, C_{u_i} is the local constraint for the decap that can be applied at node i, and C_{tot} is the total decap budget for the power grid design.

III. Cross-Entropy Method

In this section, we briefly introduce the theoretic background of the cross-entropy method which was originally proposed in [6], as well as key steps for the cross-entropy implementation for practical circuit design and decap budgeting optimization problems.

A. Theoretical Foundation

Cross-entropy method is a general Monte Carlo approach using the importance sampling technique that is proposed to solve rare event probability estimation problems [16]. In any optimization problem, optimum solution can be considered as a rare event. For completeness, the cross-entropy method is briefly introduced as follows. We refer the interested reader to [6] for further details.

Consider a minimization problem shown in

$$\gamma^* = \min_{x \in \chi} S(x) \tag{10}$$

where γ^* represents the minimal value of $S(x)$ and x is defined in the space χ. The cross-entropy method first formulates a family of probability density functions (PDF) distributed in χ, denoted by $f(x, v)$, parameterized by $v \in \nu$. For a minimization objective $S(x)$, we define

$$\ell(\gamma) = P_u(S(X) \le \gamma) = E_u(I_{\{S(X) \le \gamma\}}) \tag{11}$$

where γ is a threshold and $X = (X_1, X_2, ..., X_n)$ is a random vector (a set of n random samples) generated by PDF with parameter v set to u in $f(x, v)$. P_u denotes the probability, E_u denotes the expectation, and $I(\cdot)$ is the indicator function, i.e., $I_{S(x) \le \gamma} = 1$ if and only if $S(x) \le \gamma$.

The target of cross-entropy based optimization is to maximize γ such that $\ell(\gamma)$ approaches 0. At that moment, we conclude that γ gives the largest value such that $S(X)$ is (with high probability) greater than γ, i.e., $S(x) > \gamma$. That is, γ is the maximum lower bound on the minimization objective function $S(x)$, and thus achieves the minimum. There are two main issues in the above idea. First, how to compute $\ell(\gamma)$ with a known γ? Second, given current γ, how to obtain a better one? These questions will be answered in the following. For the first question, given a γ, one would perform an exhaustive search on the solution space χ to estimate $\ell(\gamma)$ that is certainly not practical. The practical way is to generate some samples from χ and run Monte Carlo simulations to estimate $\ell(\gamma)$. Precisely, generate n samples drawn from $f(x, v)$ and estimate $\ell(\gamma)$ as

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(X_i) \le \gamma\}}. \tag{12}$$

The problem with this simple Monte Carlo simulation idea is its low efficiency. One may require a large number of samples to accurately estimate $\ell(\gamma)$ when $S(X) \le \gamma$ is a rare event. Recall that if the probability $\ell(\gamma)$ is quite small, we call $S(X) \le \gamma$ a rare event. Note that, in our case, $S(X) \le \gamma$ will eventually be a rare event when $\ell(\gamma)$ approaches 0 as γ approaches the minimum. To tackle this problem, the cross-entropy method explores the power of importance sampling

technique. Basically, importance sampling uses a different probability density function $k(x, p)$ on χ and computes the estimation of $\ell(\gamma)$ as $\hat{\ell}(\gamma)$ using

$$\hat{\ell}(\gamma) = \frac{1}{N}\sum_{i=1}^{N} I_{\{S(X_i)\le\gamma\}}\frac{f(X_i, u)}{k(X_i, p)}. \tag{13}$$

At this moment, if we define

$$k^*(x, p) = \frac{I_{\{S(X)\le\gamma\}} f(x, u)}{\ell(\gamma)} \tag{14}$$

and choose k^* to be our k, we have

$$\hat{\ell}(\gamma) = \frac{1}{N}\sum_{i=1}^{N} I_{\{S(X_i)\le\gamma\}}\frac{f(X_i, u)}{k^*(X_i, p)} = \ell(\gamma). \tag{15}$$

In fact, in this estimation, only one sample suffices since the above is true for all i, that is a well-known fact in the importance sampling technique. Various sampling methods can be applied in CE algorithm, such as Latin hypercube sampling (LHS), and SIS techniques [10].

The point is how to obtain k^*. While it is difficult to directly compute k^* since $\ell(\gamma)$ is unknown (that is our target), one can use some probabilistic distributions close to k^* in computation. Formally, the cross-entropy technique defines the distance between two probability density functions $k(x, p)$ and $f(x, v)$ using Kullback–Leibler distance. In information theory literature, it is also referred to as cross-entropy that gives the name of the whole optimization approach. The Kullback–Leibler distance is defined as follows:

$$d(k, p) = E_f \ln\frac{k(x, p)}{f(x, v)} \tag{16}$$

$$= \int k(x, p)\ln k(x, v)dx - \int k(x, p)\ln f(x, v)dx. \tag{17}$$

Minimizing Kullback–Leibler distance is equivalent to finding p to maximize

$$\max_p \int k^*(x, p)\ln f(x, v)dx. \tag{18}$$

Plugging k^* in, one has

$$\max_p \int \frac{I_{\{S(X\_i)\le\gamma\}} f(x, v)}{\ell(\gamma)}\ln f(x, v)dx \tag{19}$$

that is

$$\max_p E_u I_{\{S(X)\le\gamma\}}\ln f(X, p). \tag{20}$$

By following [6], one can apply importance sampling again to rewrite the above to

$$\max_p E_u I_{\{S(X)\le\gamma\}}\frac{S(x, v)}{S(x, w)}\ln f(x, p) \tag{21}$$

for any reference parameter w. Its optimal p can be formulated as follows:

$$p^* = \arg\max_p E_u I_{\{S(X)\le\gamma\}}\frac{S(x, v)}{S(x, w)}\ln f(X, p). \tag{22}$$

One can estimate p^* as follows:

$$\hat{p}^* = \arg\max_p \frac{1}{N}\sum_{i=1}^{N} I_{\{S(X_i)\le\gamma\}}\frac{S(X_i, v)}{S(X_i, w)}\ln f(X_i, p) \tag{23}$$

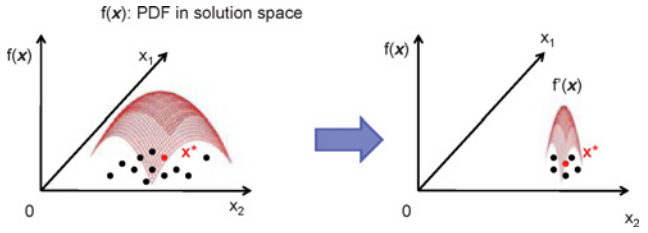


Fig. 4. PDF updating scheme in CE method.

where $X_1, X_2, ..., X_n$ are drawn from $f(x, w)$. Taking the derivative, p can be computed through solving

$$\frac{1}{N}\sum_{i=1}^{N} I_{\{S(X_i)\le\gamma\}}\frac{S(X_i, v)}{S(X_i, w)}\nabla\ln f(X_i, p) = 0. \tag{24}$$

B. Implementation

In the implementation of the cross-entropy method, by following [6], the algorithm proceeds iteratively. Multiple decap budgeting solutions are generated in each iteration. Each solution is treated as a sample. At the tth iteration, one generates a set of samples and selects the top (in terms of solution quality) ζ percent samples to update the parameters of the PDF $f(x, v)$ parameterized by v, as shown in Fig. 4.

Since top ζ percent samples are selected, in each iteration, γ is adaptively supplied. After each iteration, γ is set to the minimization objective value $f(x)$ corresponding to the $(\zeta\cdot n)$th best sample after ranking all of the n samples in terms of their solution quality.

C. Limitation of Traditional CE Method

In the traditional CE method, unavoidably, a great number of samples are required that may result in a prohibitively much longer runtime in finding the path to approach the optimal solution. Moreover, the objective function with more input parameters will generate a larger solution space. Meanwhile, in order to fully cover all kinds of input parameter combinations (decap distribution), more samples are inevitably required, that will lead to a great increase in optimization time cost.

In this work, a hierarchical CE-based optimization is proposed for decap assignment. To reduce the solution space, a partitioning technique is adopted to group multiple nearby candidate decap nodes into one candidate decap block, such that the total number of candidates can be greatly reduced. Consequently, fewer samples are required to approximately cover the whole solution space. Furthermore, to obtain a much faster convergence, compared to traditional CE method, sensitivity-guided decap budgeting is proposed to maximize the effectiveness of decaps in mitigating total noise.

IV. Sensitivity-Guided CE (SCE) Method

Sensitivity-guided CE method is a hierarchical decap budgeting optimization approach that applies different decap budgeting strategies for the block level and node level decap optimizations, as shown in Fig. 5.

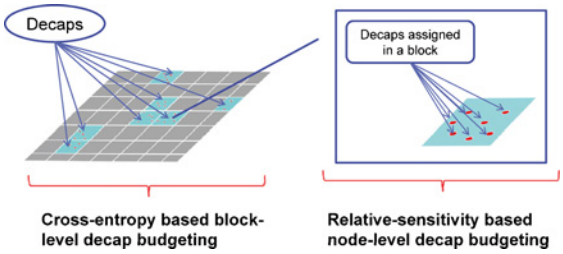


Fig. 5. Hierarchical optimization for the block level and node level decap budgeting.

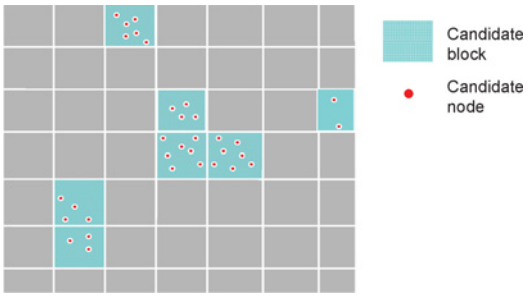


Fig. 6. Candidate decap blocks and decap nodes.

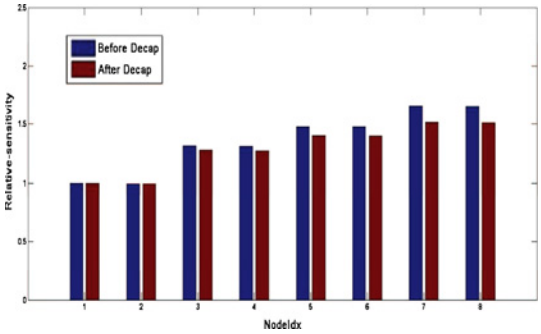


Fig. 7. Comparison of relative sensitivities before and after applying decap.

A. Algorithm Overview

The conventional CE method can be considerably costly when applied to decap budgeting problem, since large number of simulation samples that involve power supply noise computations through full transient analysis are required. In order to reduce the number of required simulation samples, a novel hierarchical sensitivity-guided CE method is presented in this section.

For a realistic power grid circuit, the number of candidate nodes (variables) for decap budgeting optimization can be more than a few hundreds or thousands. To reduce the optimization complexity, we adopt a grid partitioning scheme (as shown in Fig. 6) to help identify candidate blocks that contain different numbers of candidate nodes, and set each block decap budget as an optimization variable during the CE iterations, which can effectively reduce the number of variables as well as the required simulation samples.

Moreover, a sensitivity-based decap budgeting scheme for each candidate block (obtained by partitioning) is developed in our new approach in order to accelerate the direction search of optimal solution. The hierarchical scheme is briefly described

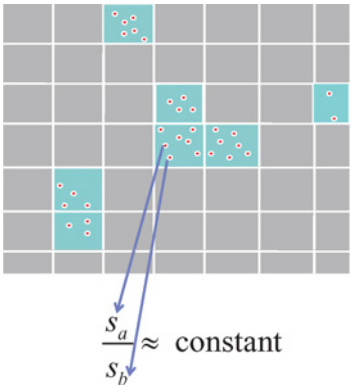


Fig. 8. Relative decap sensitivity values of the nodes within each block do not change significantly during the SCE iterations.

as follows. For each candidate block, the total block level decap budget is first determined using the CE method. Next, within each decap candidate block, the candidate nodes with larger noise sensitivities, as shown in (7), are assigned with greater decap values that do not exceed the local constraint C_u (Section II). By applying the block partitioning scheme and sensitivity-guided decap budgeting method, our SCE-based algorithm can lead to a significant reduction in computational cost during the decap budgeting optimization process.

It should be emphasized that, as shown in Figs. 7 and 8, the relative decap sensitivities within the same block would be similar even the absolute decap sensitivity values may have changed significantly after applying the block level decap budgeting step using the CE method. Therefore, the SCE decap algorithm only requires one time adjoint sensitivity analysis throughout the decap optimization procedure. In other words, instead of using the absolute sensitivity values for decap budgeting, we can use the relative sensitivity information to assign the decap values within each circuit block. In comparison, the previous sensitivity-based optimization approaches, such as the nonlinear optimization-based [2] or the conjugate gradient (CG)-based methods [1], need to perform the adjoint analysis for each optimization iteration.

B. SCE Decap Budgeting Optimization Algorithm

The basic idea of SCE algorithm flow is concluded in Algorithm 1 and described in detail as follows.

1) First, the power grid circuit is geometrically divided into many blocks (each block contains many candidate nodes). Next, the block level decap values are generated by the CE method using the LHS-based importance sampling technique. A screening process is introduced during the sample generation phase to ensure that each block decap value is within a valid range. For instance, a whole power grid circuit can be first partitioned into blocks with 30×30 dimensions, and then decap budgeting sample for each block can be generated by following the CE method subsequently.
2) Next, within each block, the relative sensitivity information of each decap candidate node (obtained through performing the adjoint sensitivity analysis in the initialization phase) is utilized to determine the fine-grained

Algorithm 1 The SCE-LHS decap budgeting algorithm

1: Perform partitioning scheme that divides the power grid circuit into n blocks and groups the candidate decap nodes into candidate decap blocks;
2: Compute sensitivities for all candidate decap nodes by adjoint sensitivity analysis and further calculate the relative sensitivity values;
3: Initialize the parameters with normal distributions for the following SCE iterations: $\vec{\mu}_0 = \{\mu_0^{(1)}, \mu_0^{(2)}, \ldots, \mu_0^{(i)}, \ldots \mu_0^{(n)}\}$ and $\vec{\sigma}_0 = \{\sigma_0^{(1)}, \sigma_0^{(2)}, \ldots, \sigma_0^{(i)}, \ldots \sigma_0^{(n)}\}$. For candidate block i, set the mean to be $\mu_0^{(i)} = \frac{N_i}{\sum_{j=1}^{n} N_j} C_{tot}$, and the standard deviation to be $\sigma_0^{(i)} = \frac{N_i C_u}{3}$, where N_i is the number of decap candidates in circuit block i, C_{tot} is the total decap budget and C_u is the upper bound of decap values for individual candidate node. Set $t = 1$.
4: Repeat steps 5 to 9 till the pre-defined stop criterion is satisfied or the total iteration number t reaches the limit;
5: Use Latin hypercube importance sampling method (LHS) to generate M samples $\{C_1, C_2, \ldots C_M\}$ from the $N(\vec{\mu}_{t-1}, \vec{\sigma}_{t-1})$ distribution, and use acceptance-rejection approach to select K feasible samples that satisfy the constraints in (9);
6: In each candidate block, distribute the decap values to each candidate node based on the relative sensitivity obtained from step 2;
7: Run transient power grid analysis for each sample and select top (best) N^{elite} samples (the samples that give the smallest noises). Define I the set that includes the indices of these top samples;
8: Update $\vec{\mu}_t$ and $\vec{\sigma}_t$: $\mu_t^{(i)} = \sum_{j \in I} C_j^{(i)} \Big/ N^{elite}$ and $\sigma_t^{(i)} = \sqrt{\sum_{j \in I} (C_j^{(i)} - \mu_t^{(i)})^2 \Big/ N^{elite}}$;
9: $t = t + 1$;
10: Return the decap solution.

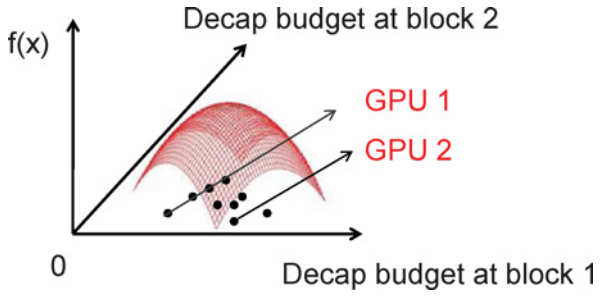


Fig. 9. SCE algorithms accelerated on multi-core-multi-GPU system.

decap solutions of all nodes. For instance, the decap value C_{ij} in block i can be assigned as $C_{ij} = \frac{s_{ij} C_{i,\text{tot}}}{\sum s_{ik}} \hat{C}_{ij}$, where s_{ij} is the sensitivity of decap candidate $\hat{C}_{ij}$ in block i.

C. Parallel Decap Budgeting Optimization

The optimization methods in [1]–[4] need to re-evaluate the sensitivity in each iteration, which may not be easily parallelized. On the other hand, the proposed SCE decap optimization algorithm can be efficiently parallelized on multi-core-multi-GPU platforms. For instance, power grid transient simulations for evaluating the supply noises of decap budgeting samples can be processed in parallel as shown in Fig. 9.

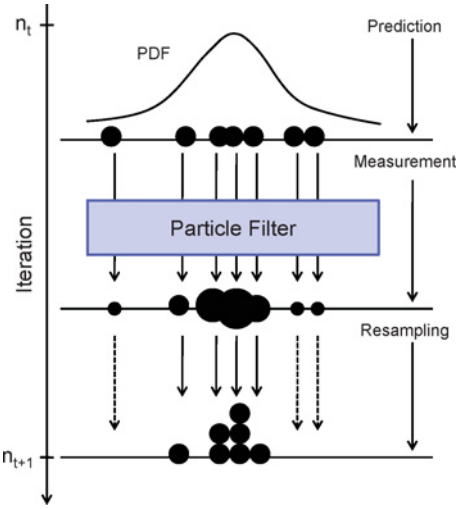


Fig. 10. Particle filter for sequential importance sampling.

V. SCE with Sequential Importance Sampling (SIS)

In LHS method, the range of each variable is divided into M equally probable intervals, and then M sample points are placed to satisfy the LHS requirements. In this case, all intervals are equally weighted without distinguishing the intervals that better samples may fall into. As a result, the LHS-based sampling method may create some less useful simulation samples in the SCE iterations. To further improve the effectiveness of simulation samples, SIS technique has been adopted in our SCE algorithm for pruning the block level decap budgeting samples. In SIS, a particle filter is constructed to explore the solution region, and to facilitate the computation of target probability, which has been successfully applied for SRAM yield analysis in [10].

A. Particle Filter

SIS is a commonly used particle filtering algorithm for estimating the sequence of hidden parameters, x_k for $k = 0, 1, 2, \ldots$, based on the observed data y_k. It also aims to approximate the filtering distribution by a weighted set of M particles (also called samples) [17]

$$w_k{}^{(n)}, x_k{}^{(n)} : n \in \{1, \ldots, M\} \tag{25}$$

where x_k represents the kth interval, and w_k represents the weight of the kth interval that are approximations to the relative posterior probabilities of the samples satisfying $\sum_{n=1}^{M} w_k{}^{(n)} = 1$. As in importance sampling, the mean of a PDF function $f(\cdot)$ can be approximated by a weighted average

$$\int f(x_k) p(x_k | y_0, \ldots, y_k) dx_k \approx \sum_{n=1}^{M} w_k{}^{(n)} f(x_k{}^{(n)}). \tag{26}$$

By drawing samples from the proposal PDF $f(\cdot)$, the importance weights are updated up to a normalizing constant

$$\hat{w}_k^{(n)} = w_{k-1}^{(n)} p(y_k | x_k^{(n)}). \tag{27}$$

In this paper, the particle filter is used to explore the solution region. It can adaptively track the region containing better

solutions regardless of the probability distribution of the solution region [10]. The procedure of the particle filtering phase consists of three key steps: *prediction*, *measurement*, and *resampling*, as shown in Fig. 10. The features of the particle filter are summarized as follows.

1) At the prediction step, new samples are generated based on predicted solution region and updated PDF.
2) At the measurement step, the solution quality of each sample is evaluated, and the weight of each sample is updated.
3) At the resampling step, samples with better solution quality are replicated, while the PDF and new solution region is predicted and updated based on replicated samples, respectively.

By applying these steps iteratively, samples converge to a local optimal solution region. Since samples with higher quality are replicated, the variance of the PDF reduces quickly.

B. Implementation of SCE with SIS

In our implementation of sequential importance sampling technique applied in SCE (SCE-SIS) method, the initial solution region is predicted by following the decap sensitivity. The central point of the region which is also known as the mean $\vec{\mu}_t$ of the PDF is initialized along the sensitivity direction, and a large radius which is also called variance $\vec{\sigma}_t$ of the PDF is assigned to the region as well. In this paper, Gaussian distribution is adopted, and any other distributions can be used for initialization. In the following steps, a set of samples are randomly generated within the predicted solution region and evaluated. Subsequently, the solution region with a new mean and a smaller radius is updated based on the top best samples. The above procedures will be applied until the convergence condition is satisfied.

At the prediction step, new samples are randomly generated based on the Gaussian distribution with its mean and variance updated based on the top best samples in the previous iteration. While in the first iteration, the mean is initialized based on the decap sensitivity, and the variance is initialized with a large value. At the measurement step, the power supply noise of all samples is evaluated by applying power grid transient analysis. The weight of each sample is updated based on the filter. The top best samples with smaller power supply noise will be kept and replicated, while other samples will be eliminated due to higher power supply noise. Samples with greater weights will be replicated into more copies. Thus the mean of the PDF is gradually moving closer to the interval that holds the best sample, and the variance has been reduced quickly which can lead to faster convergence rate.

Fig. 11 illustrates how the SIS works with the particle filter. In this example, for the simplicity, a 2-D decap optimization is used as an example. $\vec{x} = \{x_1, x_2\}$ is a 2-D variable which represents two decap candidates and the local and global constraints are assumed as

$$x_1 \leq 1 \quad \text{and} \quad x_2 \leq 1 \quad \text{and} \quad x_1 + x_2 \leq 1.5. \tag{28}$$

As shown in Fig. 11(a), arrow $\vec{S}$ denotes the decap sensitivity, c_1 denotes the mean of solution region R_1, and r_1 denotes the variance of solution region R_1. After eliminated and replicated samples are identified as illustrated in Fig. 11(b), in the next iteration, the new solution region R_2 is updated with a variance r_2 and mean c_2, as shown in Fig. 11(c). A new set of eliminated and replicated samples are also obtained in Fig. 11(d). By repeating the above processes, samples will converge to the final optimal solution.

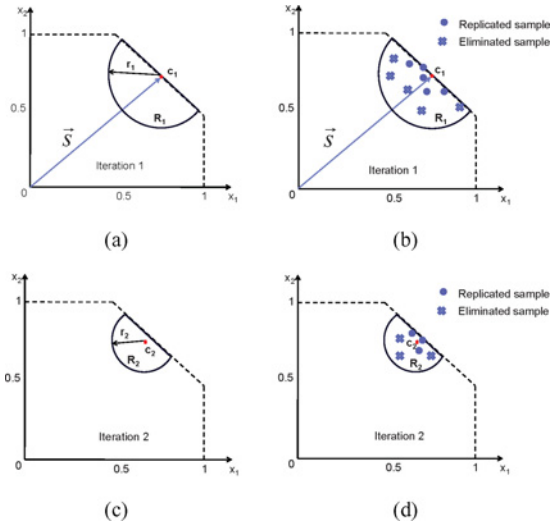


Fig. 11. Particle filter applied in decap optimization. (a) Initial solution region R_1. (b) Replicated and eliminated samples in R_1. (c) Updated solution region R_2. (d) Replicated and eliminated samples in R_2.

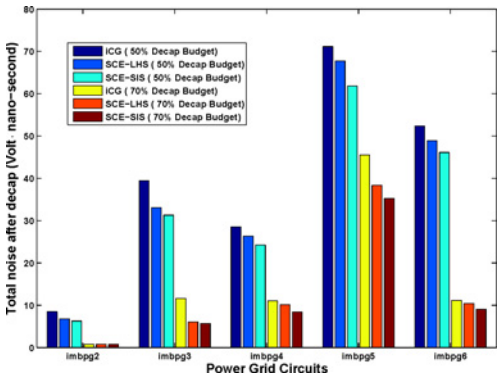


Fig. 12. Comparison of iCG, SCE-LHS, and SCE-SIS optimization results with different decap budgets.

VI. Experiment Results

A. Experimental Setup

Extensive experiments have been performed to evaluate our proposed hierarchical SCE-based optimization algorithms for decap budgeting. LHS and SIS sampling methods are adopted in SCE algorithm. A set of IBM-benchmark power grid circuits [18] with various supply noise levels are tested in our experiments. Our proposed algorithms and the iCG algorithm [1] are implemented in C++, while the power grid analysis and adjoint sensitivity analysis programs are implemented in C++ and the GPU programming language CUDA [19]. All experiments are executed on a 2.66 GHz quad-core machine with Ubuntu 8.04 64 bit, 6 GB DRAM memory, and two Nvidia GeForce GTX285 GPU cards.

TABLE I
EXPERIMENTAL SETUP INFORMATION

CKT	N	I	L	r_{slew} (A/second)	Power Switching (V·mA)	#Vio.N.	Total Noise (V· nano-second)	Full Budget (nF)
*ibmpg*2	127 238	37 926	5	2.76×10^{11}	9.55	481	43.0	6
*ibmpg*3	851 584	201 054	5	9.39×10^{10}	3.17	1829	163.0	18
*ibmpg*4	953 583	276 976	6	1.76×10^{10}	0.594	912	81.0	6
*ibmpg*5	1 079 310	540 800	3	1.80×10^{10}	0.608	1809	161.0	16
*ibmpg*6	1 670 494	761 484	3	1.39×10^{10}	0.472	1926	74.0	20

"N" denotes the total number of nodes in the power grid, "I" denotes the total number of current sources, "L" denotes the number of metal layers of each power grid design, "r_{slew}" denotes the slew rate of current sources with the unit of A/second, "# Vio.N." denotes the number of violated nodes, and "Total Noise" denotes the original total noise without decaps. "Power Switching" denotes the average power in one clock period for one current source. The power supply for all benchmarks is 1.8 V. The duration time of transient analysis is 0.5 nano-second.

TABLE II
COMPARISON OF ICG, CE, AND PARTITION-BASED SCE METHODS WITH DIFFERENT BUDGET LEVELS

Budget 50.00%	iCG			CE-LHS			Partition-Based SCE-LHS					
							Block Size 10 × 10			Block Size 25 × 25		
CKT	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*
*ibmpg*2	8.47 (19.7%)	15	62.4	15.09 (35.1%)	20	315.8	6.36 (14.8%)	3	38.2	6.79 (15.8%)	2	25.4
*ibmpg*3	39.45 (24.2%)	15	638.3	44.50 (27.3%)	30	3603.9	31.30 (19.2%)	4	400.5	33.09 (20.3%)	3	300.4
*ibmpg*4	28.51 (35.2%)	13	375.9	32.48 (40.1%)	25	2462.3	24.54 (30.3%)	3	209.9	26.33 (32.5%)	2	145.6
*ibmpg*5	71.16 (44.2%)	15	1265.8	72.61 (45.1%)	30	9234.1	61.34 (38.1%)	4	1026.1	67.78 (42.1%)	3	729.3
*ibmpg*6	52.37 (30.1%)	15	1409.1	62.12 (35.7%)	40	12072.5	48.20 (27.7%)	5	1257.5	48.89 (28.1%)	3	770.5
Budget 70.00%	iCG			CE-LHS			Partition-Based SCE-LHS					
							Block Size 10 × 10			Block Size 25 × 25		
CKT	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*
*ibmpg*2	0.77 (1.8%)	13	54.8	5.33 (12.4%)	20	312.4	0.77 (1.8%)	3	38.2	0.77 (1.8%)	2	25.4
*ibmpg*3	11.57 (7.1%)	14	591.7	16.46 (10.1%)	31	3724.1	2.77 (1.7%)	3	307.3	6.03 (3.7%)	2	203.5
*ibmpg*4	11.02 (13.6%)	13	375.0	11.66 (14.4%)	25	2464.3	8.34 (10.3%)	3	210.1	10.13 (12.5%)	2	142.9
*ibmpg*5	45.56 (28.3%)	15	1285.5	50.22 (31.2%)	30	9236.6	34.45 (21.4%)	4	1028.4	38.32 (23.8%)	3	734.7
*ibmpg*6	11.14 (6.4%)	15	1429.8	16.88 (9.7%)	39	11782.4	8.87 (5.1%)	5	1219.1	10.44 (6.0%)	3	768.6
Budget 90.00%	iCG			CE-LHS			Partition-Based SCE-LHS					
							Block Size 10 × 10			Block Size 25 × 25		
CKT	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*	*Noise*	*Iter.*	*Time (s)*
*ibmpg*2	0.01 (0.02%)	16	65.1	0.01 (0.02%)	19	294.0	0.00 (0.0%)	5	62.5	0.00 (0.0%)	3	37.4
*ibmpg*3	0.00 (0.0%)	16	617.3	0.00 (0.0%)	29	3481.4	0.00 (0.0%)	3	298.7	0.00 (0.0%)	4	398.1
*ibmpg*4	4.21 (5.2%)	13	376.8	4.94 (6.1%)	26	2542.1	3.24 (4.0%)	4	286.1	3.65 (4.5%)	3	218.5
*ibmpg*5	14.81 (9.2%)	17	1459.2	17.87 (11.1%)	32	7893.8	11.43 (7.1%)	5	1251.4	13.52 (8.4%)	3	1119.3
*ibmpg*6	0.00 (0.0%)	1	150.6	0.00 (0.0%)	10	4240.1	0.00 (0.0%)	1	353.6	0.00 (0.0%)	1	356.1

Both CE and SCE methods use LHS sampling methods. For *CKT ibmpg*6, the block sizes are 30 × 30 and 50 × 50, respectively. "Noise" represents the optimized supply noise in V·nano-second, and the percentage noise of the original total noise without Decap, and "Iter." denotes the number of optimization iterations.

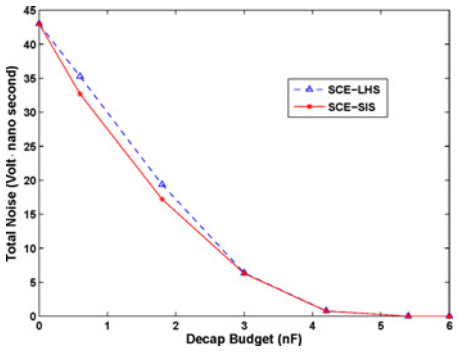


Fig. 13. Tradeoff analysis of total noise and the decap budget for test case *ibmpg*2 using SCE-LHS and SCE-SIS methods.

In our experiments, we define the 100% full decap budget as the minimally required budget which can eliminate all the power supply noises after uniformly assigning decaps to all candidate decap nodes. Our decap budgeting objective is to minimize the total supply noise subject to local and global constraints in (9). A few budget levels, such as 50%, 70%, and 90% of the full budget are considered in the decap budgeting optimizations. Table I summarizes the experimental setup information of our decap budgeting optimizations. The noise here is defined as the product of voltage and time interval: $V \cdot nano\text{-}second$. The full decap budget set for different benchmarks may vary a lot: for test cases with larger chip sizes and supply noises, greater decap budgets are applied, while smaller test cases are assigned with smaller full budgets. To measure the final decap solution quality, we use the final power supply noise level metric: the decap solution resulting in smaller noises indicates higher decap solution quality.

B. Comparisons of Decap Optimization Methods

Detailed decap budgeting results are summarized in Tables II and III. In Table II, "iCG" denotes the improved conjugate gradient method [1], "CE-LHS" denotes the traditional

TABLE III
RESULTS OF SCE-SIS WITH THE TRADEOFFS BETWEEN RUNTIME AND SOLUTION QUALITY

Budget 50.00%	SCE-SIS A				SCE-SIS B			
CKT	*Noise*	*Iter.*	*Time (s)*	*Q.I.*	*Noise*	*Iter.*	*Time (s)*	*T.I.*
ibmpg2	6.28 (14.6%)	2	25.3	7.5%	6.71 (15.6%)	3	26.7	−5.0%
ibmpg3	31.30 (19.2%)	3	300.7	5.4%	32.60 (20.0%)	3	235.4	21.6%
ibmpg4	24.22 (29.9%)	2	145.8	8.0%	26.24 (32.4%)	3	153.1	−5.5%
ibmpg5	61.82 (38.4%)	3	730.1	8.7%	65.69 (40.8%)	3	560.9	23.2%
ibmpg6	46.11 (26.5%)	3	769.8	5.6%	46.46 (26.7%)	4	718.3	6.7%
Budget 70.00%	SCE-SIS A				SCE-SIS B			
CKT	*Noise*	*Iter.*	*Time (s)*	*Q.I.*	*Noise*	*Iter.*	*Time (s)*	*T.I.*
ibmpg2	0.73 (1.7%)	2	25.4	5.6%	0.77 (1.8%)	3	26.7	−5.0%
ibmpg3	5.71 (3.5%)	2	203.2	5.4%	5.71 (3.5%)	3	213.4	−4.9%
ibmpg4	8.42 (10.4%)	2	142.9	16.8%	10.13 (12.5%)	3	150.3	−5.6%
ibmpg5	35.26 (21.9%)	3	734.7	7.9%	37.03 (23.0%)	3	570.7	22.3%
ibmpg6	9.05 (5.2%)	3	768.9	13.3%	9.92 (5.7%)	4	725.4	5.6%
Budget 90.00%	SCE-SIS A				SCE-SIS B			
CKT	*Noise*	*Iter.*	*Time (s)*	*Q.I.*	*Noise*	*Iter.*	*Time (s)*	*T.I.*
ibmpg2	0.00 (0.0%)	3	37.3	0.0%	0.00 (0.0%)	3	26.1	−2.7%
ibmpg3	0.00 (0.0%)	3	298.8	0.0%	0.00 (0.0%)	4	278.8	30.1%
ibmpg4	3.32 (4.1%)	3	211.5	8.8%	3.65 (4.5%)	3	150.8	27.5%
ibmpg5	11.43 (7.1%)	3	1120.5	15.4%	12.88 (8.0%)	4	1045.5	5.8%
ibmpg6	0.00 (0.0%)	1	354.4	0.0%	0.00 (0.0%)	1	283.5	19.6%

"SCE-SIS A" denotes SCE-SIS with the same number of samples as SCE-LHS, and "SCE-SIS B" denotes SCE-SIS with the same solution quality as SCE-LHS. "Noise" represents the optimized noise in V·nano-second, and the optimized percentage noise of the original total noise without Decap, "Iter." denotes the number of iterations, "Q.I." denotes the solution improvement, and "T.I." denotes the runtime improvement.

cross-entropy method (as described in Section III) using LHS sampling method, and "partitioning-based SCE-LHS" method denotes our proposed SCE method with partitioning and LHS sampling technique. In our experiments, more than 20 samples are generated for each CE iteration and around 10 samples are generated for each SCE iteration. In Tables II and III, "Noise" denotes the optimized supply noise in the unit of $V \cdot nano\text{-}second$ and the percentage noise ratio of the original supply noise without inserting any decaps.

As observed in Table II, iCG method usually takes much more iterations and longer runtime than the SCE-LHS method. The final decap optimization results (solution quality) of the iCG method are also significantly worse than the results obtained by SCE algorithm, since the standard CG method would require many CG iterations to converge considering a large number of the decap variables. Although an iCG method is proposed to improve the convergence [1], it may not converge to a fairly good decap solution. In our experiments, although iCG method requires only a few iterations (13–15 iterations) to converge, it produces worse decap solution quality compared to the original CG method. On the other hand, our SCE-LHS method provides the similar solution quality as CG method, while taking much less optimization time.

The partition-based SCE-LHS method with different block sizes can result in different decap budgeting solutions. From Table II, it can be observed that when the SCE-LHS method uses a coarser block partitioning strategy, although the final decap solution quality can be slightly worse, the overall optimization can be significantly faster than using finer block partitioning schemes. This is mainly due to the drastically reduced number of optimization variables and thus the required simulation samples during the SCE optimization iterations. Although the original CE method can converge to the final solution using a sufficiently large number of simulation samples, we observe in Table II that when using the original CE method with LHS sampling but no sensitivity information the runtime for decap optimization can be much longer than using iCG and SCE-based methods, since CE considers each of candidate decaps as a variable while SCE only treats the block level decap budget as a single optimization variable.

In addition, Table III shows the comparison between sequential importance sampling technique (SCE-SIS) and SCE-LHS in terms of solution quality and runtime. As illustrated in Table III, "SCE-SIS A" includes the experimental results on the final decap solution quality of SCE-SIS when using the same number of simulation samples as SCE-LHS method. In the meantime, "SCE-SIS B" compares the runtime between SCE-SIS and SCE-LHS for obtaining the similar decap solution quality or supply noises. As observed, with the same number of samples in each SCE iteration, "SCE-SIS" can obtain up to 16.8% improvement on decap solution quality compared to "SCE-LHS" method, while for obtaining the similar decap solution qualities, "SCE-SIS" can obtain up to 30% improvement in runtime.

As shown in Fig. 12, for all the test cases, our SCE-LHS method produces better solution quality than the iCG method. Moreover, compared to SCE-LHS, SCE-SIS brings further improvement in solution quality. In Fig. 13, we also show the tradeoff between decap budget and the total power supply noise levels (solution quality) for test case *ibmpg2* using SCE-LHS and SCE-SIS methods. As observed, more than 95% noise can be eliminated by using only 70% of full decap budget.

Next, we demonstrate the effectiveness of our decap budgeting methods. Fig. 14 illustrates the current waveform, voltage waveform without applying any decaps, and voltage waveform

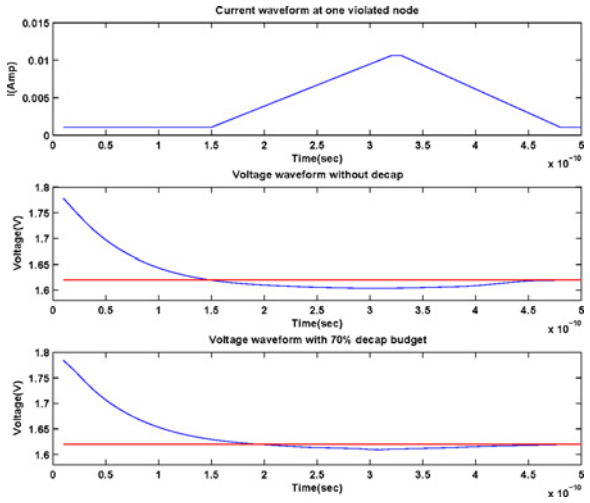


Fig. 14. Current and voltage waveforms of one violated node in transient analysis of ibmpg2, where the red curve denotes the voltage threshold.

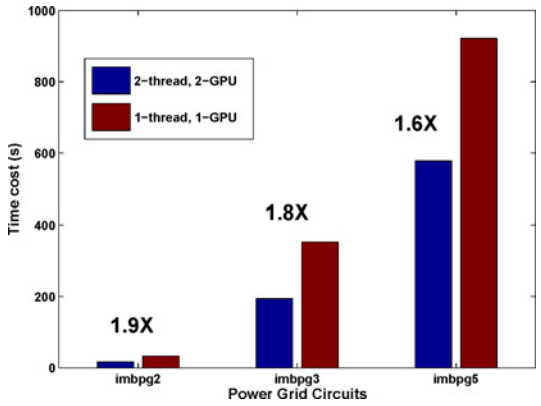


Fig. 15. Runtime results of the SCE-LHS decap budgeting algorithm executed on parallel CPU-GPU platforms.

with 70% decap budget for one violated node in *ibmpg*2, where the red solid curve represents the voltage threshold that is 90% of the power supply voltage *VDD*.

In the end, we show the runtime results of the proposed SCE-based decap budgeting method running on parallel CPU-GPU platforms in Fig. 15. For the three test cases with various problem sizes, 1.6× to 1.9× speedups have been achieved on our dual-CPU-dual-GPU machine. Since the most computational costly kernels (e.g., the transient circuit simulations and sensitivity computations) in SCE-based algorithms can be easily parallelized, the proposed SCE decap budgeting algorithm is expected to scale well on more powerful parallel computing platforms.

VII. Conclusion

In this paper, we presented a hierarchical CE-based optimization approach that is more efficient and parallel-friendly than prior decap optimization algorithms, by exploring the power of rare event probability theory and importance sampling. To gain higher efficiency, an SCE framework is introduced that integrates traditional CE method with a partitioning-based sampling strategy to effectively reduce the number of optimization variables in large-scale decap budgeting problems. Compared to previous iCG method and conventional CE method, SCE algorithm with Latin hypercube sampling method (SCE-LHS) can provide 2× speedups, while achieving up to 25% improvement on power supply noise. To further improve decap optimization efficiency and solution quality, SCE-SIS is also developed, which leads to up to 16.8% further reduction on the total power supply noise in similar runtime compared to the original SCE-LHS method. We also observed up to 1.8× speedups when running the SCE-LHS algorithm on a dual-CPU-dual-GPU machine.

References

[1] H. Li, J. Fan, Z. Qi, S. Tan, L. Wu, Y. Cai, and X. Hong, "Partitioning-based approach to fast on-chip decoupling capacitor budgeting and minimization," *IEEE Trans. Comput.-Aided Des.*, vol. 25, no. 11, pp. 2402–2412, Nov. 2006.

[2] H. Su, S. Sapatnekar, and S. Nassif, "Optimal decoupling capacitor sizing and placement for standard-cell layout designs," *IEEE Trans. Comput.-Aided Des.*, vol. 22, no. 4, pp. 428–436, Apr. 2002.

[3] K. Wang and M. Marek-Sadowska, "On-chip power-supply network optimization using multigrid-based technique," *IEEE Trans. Comput.-Aided Des.*, vol. 24, no. 3, pp. 407–417, Mar. 2005.

[4] M. Zhao, R. Panda, S. Sundareswaran, S. Yan, and Y. Fu, "A fast on-chip decoupling capacitance budgeting algorithm using macromodeling and linear programming," in *Proc. IEEE/ACM DAC*, Sep. 2006, pp. 217–222.

[5] X. Zhao, Y. Guo, Z. Feng, and S. Hu, "Parallel hierarchical cross entropy optimization for on-chip decap budgeting," in *Proc. IEEE/ACM DAC*, Jun. 2010, pp. 843–848.

[6] R. Rubinstein and D. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning* (Annals of Operations Research). New York: Springer-Verlag, 2004.

[7] K. Chepuri and T. Mello, "Solving the vehicle routing problem with stochastic demands using the cross-entropy method," *Ann. Oper. Res.*, vol. 134, no. 1, pp. 153–181, 2005.

[8] D. Kroese, R. Rubinstein, and T. Taimre, "Application of the cross-entropy method to clustering and vector quantization," *J. Global Optimization*, vol. 37, no. 1, pp. 137–157, Jan. 2007.

[9] Z. Botev and D. Kroese, "An efficient algorithm for rare-event probability estimation, combinatorial optimization, and counting," *Methodol. Comput. Appl. Probab.*, vol. 10, no. 4, pp. 471–505, May 2008.

[10] K. Katayama, S. Hagiwara, H. Tsutsui, H. Ochi, and T. Sato, "Sequential importance sampling for low-probability and high-dimensional SRAM yield analysis," in *Proc. IEEE/ACM ICCAD*, Nov. 2010, pp. 703–708.

[11] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. Comput.-Aided Des.*, vol. 24, no. 8, pp. 1204–1224, Aug. 2005.

[12] T. H. Chen and C. C.-P. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Proc. IEEE/ACM DAC*, Jun. 2001, pp. 559–562.

[13] Z. Feng and P. Li, "Multigrid on GPU: Tackling power grid analysis on parallel SIMT platforms," in *Proc. IEEE/ACM ICCAD*, Nov. 2008, pp. 647–654.

[14] L. Pillage, R. Rohrer, and C. Visweswariah, *Electronic Circuit and System Simulation Methods*. New York: McGraw-Hill, 1995.

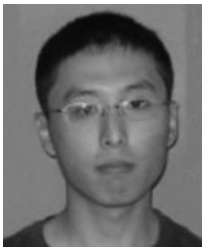
[15] Z. Feng and Z. Zeng, "Parallel multigrid preconditioning on graphics processing units (GPUs) for robust power grid analysis," in *Proc. IEEE/ACM DAC*, Jun. 2010, pp. 661–666.

[16] P. Boer, D. Kroese, S. Mannor, and R. Rubinstein, "A tutorial on the cross-entropy method," *Ann. Oper. Res.*, vol. 134, no. 1, pp. 19–67, Jan. 2005.

[17] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

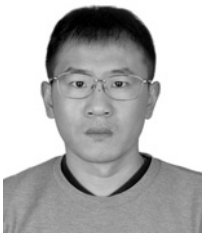
[18] S. R. Nassif. (2008). *IBM Power Grid Benchmarks* [Online]. Available: http://dropzone.tamu.edu/pli/PGBench

[19] NVIDIA Corporation. (2007). *NVIDIA CUDA Programming Guide* [Online]. Available: http://www.nvidia.com/object/cuda.html



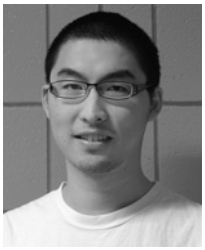
Xueqian Zhao received the B.S. degree in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2008. Currently, he is pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton.

His current research interests include parallel very large scale integration simulation and modeling on GPU, on-chip interconnect parasitics extraction, and runtime performance modeling and optimization for GPU computing.



Yonghe Guo received the B.S. and M.S. degrees in electronic engineering from the Beijing Institute of Technology, Beijing, China, in 2002 and 2006, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton.

His current research interests include very large scale integration computer-aided design and smart grid cybersecurity.



Xiaodao Chen (S'11) received the B.E. degree in telecommunication from the Wuhan University of Technology, Wuhan, China, and the M.S. degree in electronics engineering from Michigan Technological University, Houghton. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Michigan Technological University.

His current research interests include the area of computer-aided design of very large scale integration design and optimization.

Zhuo Feng (S'03–M'10) received the B.Eng. degree in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2003, the M.Eng. degree in electrical engineering from the National University of Singapore, Singapore, in 2005, and the Ph.D. degree in electrical and computer engineering from Texas A&M University, College Station, in 2009.

Since July 2009, he has been an Assistant Professor with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton. His current research interests include very large scale integration, and computer-aided design on energy efficient parallel computing platforms.

Dr. Feng served on the Technical Program Committee of the International Symposium on Quality Electronic Design in 2009 and 2010. He received two IEEE/ACM William J. McCalla ICCAD Best Paper Award Nominations in 2006 and 2008.

Shiyan Hu (SM'10) received the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2008.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, where he serves as the Director of the Michigan Tech VLSI CAD Research Laboratory. He was a Visiting Professor with the IBM Austin Research Laboratory, Austin, TX, in 2010. He has over 50 journal and conference publications. His current research interests include computer-aided design for very large-scale integrated circuits on nanoscale interconnect optimization, low power optimization, and design for manufacturability.

Dr. Hu has served as a technical program committee member for a few conferences such as ICCAD, ISPD, ISQED, ISVLSI, and ISCAS. He received the Best Paper Award Nomination from ICCAD 2009.