On-Chip Decoupling Capacitor Budgeting By Sequence of Linear Programming

Zhenyu Qi[†], Hang Li[†], Jeffrey Fan[†], Sheldon X.-D. Tan[†], Yici Cai[§] and Xianlong Hong[§]
[†]Department of Electrical Engineering University of California, Riverside, CA 92521
[§]Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China *

Abstract

Excessive power supply noise increases propagation delay of switching gates and reduces noise margin of the circuit. Adding on-chip decoupling capacitors (decaps) is an effective way to reduce voltage noise in a on-chip power delivery system. In this paper, we propose an efficient and novel algorithm to allocate decaps in an area efficient way. The new algorithm applies the sequence of linear programming based approach to searching the minimum decap area to reduce voltage drop below user specified threshold. We show existing sensitivity based decap allocation algorithms tend to over estimate the decap areas due to nonlinear sensitivity dependence on decap values. Experimental results show that the proposed algorithm uses significantly less decap area than the existing conjugate gradient based approach but with similar CPU runtimes.

1 Introduction

With increasing integration density and roaring clock frequency, reliable on-chip power supply becomes a critical concern for VLSI chip design. To retain signal integrity in power/ground (P/G) network, which supplies power from the power/ground pads to all modules on a chip, extra careful design effort is required to reduce the voltage noise in the on-chip power supply networks. Excessive voltage variations, including voltage drop and ground bounce, would have an adverse impact on chip performance and reliability, since they not only degrade the already tight noise margin in today's VLSI circuits, but also increase gate delay, cause false logic switching, and sometimes even lead to logic failure. In practice, most designs now require voltage drop to be confined to certain percentage (like 10%) of the nominal supply voltage.

Adding decoupling capacitors is an effective way to reduce ΔI noise in package as shown before [6, 13]. For VLSI design, the inductive wire impedance can no longer be ignored, and the on-chip ΔI noise becomes more pronounced as inductance scales poorly with wire sizing. As a result, on-chip decaps are indispensable for robust on-chip power supply [15, 3, 17]. On the other hand, on-chip decaps are usually manufactured as gate capacitance of transistors. As the supply voltage continues scaling, leakage currents due to reduced threshold voltage and dielectric leakage prevent excessive use of decaps [2]. So economic use of on-chip decaps is equivalently important.

In this paper, we propose an efficient decap allocation algorithm, which explicitly minimizes the decap area subject to the voltage drops and other design rule constraints. We formulate the decap allocation problem as a linear programming problem and solve it by a sequence of linear programming (SLP) method. Although we only address decap optimization for voltage drop here, ground bounce can be dealt with in a similar fashion with little modification. The new decap allocation algorithm is especially suitable for P/G grids with a few troubling spots, which is typically the case for a properly designed P/G grid, where a priori decaps are added already by other tools based on some simple estimation. Experimental results show that the new algorithm uses significantly less decap area than the best existing decap allocation algorithm [8] with similar performance. Similar with [11], to achieve even higher efficiency with extremely large P/G networks, a hierarchical decap budgeting strategy can also be easily developed based on the algorithm presented here.

*This work is funded by NSF CAREER Award CCF-0448534, NSF grant OISE-0451688, UC MICRO #04-088 via Cadence Design System Inc.

The rest of this paper is organized as follows. The next section briefly reviews existing sensitivity-based decap budgeting algorithms. Section 3 forms decap budgeting into a sequence of linear programming problem. Section 4 presents the SLP based decap allocation strategy with discussion of some practical implementation considerations. Experimental results are presented in section 5. The last session concludes the paper and comments on future work.

2 Review of Sensitivity-Based Decap Allocation Algorithms

Existing on-chip decap budgeting algorithms basically fall into two categories. In [16, 12, 3, 15], the current pattern around 'hot spots' (where excessive voltage drop occurs) is first derived and electric charge needed to supply that current demand is estimated. To get an optimal decap budget, a critical step here is the precise estimation of voltage drop, which unfortunately, proves to be difficult for practical P/G networks. Usually only a lower or upper bound of maximum voltage drop can be obtained [2].

Another category is the sensitivity based optimization [17, 1, 10], where the adjoint method [5, 4] is applied to calculate sensitivity of *violation area* with respect to decap in time domain:

$$s_{ij} = \frac{\partial g_j(c_1, ..., c_m)}{\partial c_i} \tag{1}$$

where

$$\begin{aligned} g_j(c_1, ..., c_m) &= \int_0^T \max(V_{min} - v_j(t), 0)dt \\ &= \int_{t_s^{(j)}}^{t_e^{(j)}} (V_{min} - v_j(t))dt \end{aligned} \tag{2}$$

is defined as the *violation area* at node j, with $t_s^{(j)}$ the start of violation, and $t_e^{(j)}$ the end of violation. V_{min} is the tolerance of voltage drop. $v_j(t)$ is the transient voltage at node j and c_i is the added decap at node i. In [17], violation area at node j is explicitly reduced as the objective function based on the sensitivity information one at a time. But a problem with this method is that the decap area is not explicitly miniminized.

Recent work [8] improved the sensitivity based decap allocation algorithm by using time domain merged adjoint method for fast sensitivity calculation and explicitly considering the decap area optimization in the objective function. Merged adjoint method allows the sensitivity to be computed directly for the objective function instead of individual nodes. The conjugate gradient (CG) optimization method was used to minimize the following objective function iteratively, which considers decap area explicitly:

$$\sum_{i \in allnodes} (\Delta c_i) + \alpha \sum_{j \in allnodes} g_j \tag{3}$$

where the weighting factor α will keep changing in each CG iteration. Notice that in (3) we used Δc_i instead of c_i to denote added decaps at each iteration.

A drawback with CG method is the slow convergence of CG optimization. Theoretically it takes n steps before CG converges, where n is number of variables [9]. Moreover, in [8] α is estimated after each iteration by equating the two summation items in (3). The idea behind this is to let CG to reduce violation area and decap area at the same time. But such balance can be misleading for optimization – sometimes it tends to inflate an actually unimportant part and renders

optimization less efficient. For example, at later stages when there's very little violation left, a little bit more decap would suffice, but with a very large α, optimization is not very well oriented, because the direction of CG is decided by gradient of the objective function which is mainly determined by the value of α, instead of the sensitivity of violation with respect to decap itself.

More importantly, in practice, selection of α proves to be tricky and difficult. For CG to perform optimization efficiently, an objective function is supposed to be convex on its domain. If (3) is monotonously decreasing, line search for the minimum would end up with a full step and stop at the far end, adding decaps for all nodes with non-zero sensitivities and the maximum allowed decap for the node with the greatest sensitivity. This often leads to overestimation of the necessary decap. Even worse, when (3) is monotonously increasing, line search always ends at the starting point, and optimization is stuck at the current stage since no decap would be added. The first case corresponds to a too big α, which favors violation too much; the second case corresponds to a too small α, favoring decap area too much.

3 Sequence of Linear Programming Based Decaps Allocation

In this section, we show how the sequence of linear programming method can be applied to solve the decap allocation problem.

3.1 Sequence of Linear Programming

The basic idea of *sequence of linear programming* (SLP) method is to linearize the nonlinear parts of a nonlinear optimization problem and solve the linearized programming problem by a sequence of linear programming method in an iterative way. SLP method is similar to the *Newton-Raphson* method for solving nonlinear equations, where linearized circuit matrix (*Jacobian*) is solved iteratively to find the solution. So SLP will have a quadratic convergence rate and is usually better than conjugate gradient method in terms of convergence rate as demonstrated in [18].

3.2 Problem Formulation

In contrast to the inherent ambiguousness in (3), we choose decap area minimization as the sole objective and enforce violation elimination as a constraint. Since on-chip area for a capacitor is linearly proportional to the capacitance value, minimization of decap area is equivalent to minimization of total decap values.

Objective function

$$\min \sum_{i \in allnodes} (c_i) \tag{4}$$

Violation elimination constraint

$$\begin{aligned} g_j(c_1, ..., c_m) &= \int_0^T \max(V_{min} - v_j(t), 0) dt \\ &= \int_{t_s^{(j)}}^{t_e^{(j)}} (V_{min} - v_j(t)) dt \\ &= 0 \end{aligned} \tag{5}$$

Maximum decap constraint

$$(c_i) \leq d_i, \quad d_i \geq 0 \tag{6}$$

where d_i is the maximum decap allowed at node i, a parameter decided by the available white space (WS) around node i. Notice that there are other power integrity constraints like current density for eletromigration. Assuming those power integrity constraints are better addressed by other optimization options like wire-sizing and topology selection etc., we ignore them here for simplicity.

In our problem, the only nonlinear portion is the constraint (5). Hence our first step is to linearize it with sensitivity s_{ij} defined in(1). Computation for s_{ij} will be discussed in the next section. With s_{ij}, to remove the IR drop violation at node j, we add decap at node i based on the first order approximation. This is true for other candidate nodes with tunable decaps. As a result, to meet with the nonlinear violation constraint (5) at violation node j, we have the following linearized constraint for added decaps Δc_i:

$$g_j \leq \sum_{i \in allnodes} s_{ij} \Delta c_i \tag{7}$$

Replacing constraint Eq. (5) with (7), we end up with a linear programming problem. Iteration goes on until all violation is eliminated. In practice, we may not always be able to remove all the IR drop constraints due to other constraints.

3.3 Sensitivity computation

The sensitivity defined in (1) is, more precisely, incremental sensitivity (also called small change sensitivity). Two major methods to calculate incremental sensitivity are the direct method which involves construction of a sensitivity circuit, and the adjoint method [5, 4], which involves construction of an adjoint circuit [14, 7]. As concluded in [7], the former has advantage in computation of sensitivities of many responses with respect to a few parameters, and the latter fits better in cases where sensitivities of a few responses with respect to many parameters are required, which is exactly our case. To save space here we only present some relevant conclusions from the adjoint method, which can be basically deducted from Tellegen's Theorem.

The performance function in our problem is

$$g_j(c_1, ..., c_m) = \int_0^T \max(V_{min} - v_j(t), 0) dt \tag{8}$$

Following adjoint method, we have

$$s_{ij} = \int_0^T v'_{i,j}(T - t) \times \dot{v}_i(t) dt \tag{9}$$

where $\dot{v}_i(t)$ is the derivative of voltage waveform at node i with respect to the normal forward time, $v'_{i,j}(T - t)$ is the waveform at node i in the adjoint circuit under excitation at node j with respect to the reverse time related to the adjoint circuit, and s_{ij} is defined in (1). Now that the performance function (8) is already in integration form, so

$$I_j(t) = \begin{cases} \frac{\partial(V_{min} - v_j(t))}{\partial v_j(t)} = -1 & \text{if } t_s^{(j)} \leq t \leq t_e^{(j)}; \\ 0 & \text{else.} \end{cases} \tag{10}$$

which is a unit step waveform between the start and end of violation at node j.

4 Proposed SLP-based Decap Allocation Framework

We first consider some important practical issues, which would affect speed, memory and optimization results, then present the entire flow.

4.1 Constraint Relaxation

One important observation is that decap sensitivity is a nonlinear function of decap values. Detailed study shows that the decap sensitivity (absolute value) of a node typically increases with added decap values at the same node as shown in Fig. 1. It reaches a peak at some point and then goes down to zero when the violation goes away. This implies that we would over-estimate decaps based on the sensitivity calculated when no decaps or small decaps are added at the beginning.

This causes two problems. First, SLP may fail to find a feasible solution given a very small sensitivity as small sensitivity may lead to large decaps to remove the same amount of violation area due to constraint (7). But the decap areas can not go too large as they are also bounded by constraint (6). Second, we may add more decaps than necessary due to smaller sensitivities. That actually explains why existing sensitivity based methods like [17], which does not optimize decap areas, tend to require more decaps. For SLP method, this also leads to over estimation as we do not further optimize decaps for a violating node when the violation is gone after decaps are added sufficiently.

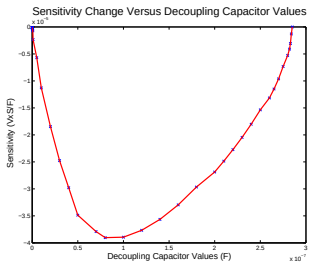

Figure 1. Typical pattern of sensitivity change with added decap value

To resolve this issue, we intentionally *relax* the violation area constraint by artificially increasing all sensitivity values calculated at earlier stages by multiplying them, or equivalently, dividing violation at all nodes, with a constant value. Notice that this relaxation won't change the relative magnitude among sensitivities, thus keeping optimization in the right direction until a feasible solution is found by a linear programming solver.

Another benefit of such constraint relaxation strategy is that for actually infeasible decap allocation problems (the violation can not be completely removed), the proposed SLP can detect the infeasible problem or give a solution which removes as much violation area as possible.

4.2 Fast Convolution Computation

If no controlled source exists in the original P/G network, which is true for P/G grids modeled as RLC linear networks, the adjoint network is a passive circuit and further improvement can be made.

From (10) we know that there's no excitation in the adjoint network and all node voltages remain zero until $t_e^{(j)}$, which is the time point when violation ends for node j (notice simulation for an adjoint network is performed in reverse time, from T to 0). As a result, convolution is only necessary in the interval $[0, t_e^{(j)}]$ for any node i and indeed, (9) can be modified with:

$$s_{ij} = \int_0^{t_e^{(j)}} v'_{i,j}(T-t) \times \dot{v}_i(t)dt \qquad (11)$$

4.3 Other Practical Issues

The most memory consuming part of the adjoint method is to store waveforms at every node in both original circuit and adjoint circuit. Basically we follow [17] and store waveforms with piecewise approximation. In this way the dimension of a waveform matrix to be stored is number of time steps by number of circuit nodes. This can be huge for large circuits, specially small time steps are required for precision. However, noticing that as the simulation for the adjoint network goes on from T to 0, convolution in (9) can be carried out at each time step on the fly. There's no need to store the adjoint waveforms, which leads to a great memory saving.

To avoid numerical problems, data scaling is needed, as sensitivities, decaps and voltage changes can be very small. Also, as indicated in [18], power networks should be converted to ground networks for better numerical stability during SLP. The transformation can be done by the following rules: (1) short-circuit all *VDD* pads to the ground; (2) inverse the directions of all independent current sources.

4.4 The SLP-based Decap Allocation Flow

With previous discussion, our decap budgeting algorithm can be summarized in Fig. 2.

Sensitivity relaxation discussed in section 4.1 is carried out in step 8, where all sensitivities are magnified by a constant. This constant can be selected according to optimization stages, or, in our implementation, the ratio of current violation with already added decap to original violation before decap placement. Usually the larger constant is, the smaller decap budget yielded, but the longer operation time.

```
DECAPBUDGETSLP(P/G network)
1   Solve input circuit, establish violation node set;
2   while (violation node set is not empty){
3     Store waveform for all nodes;
4     Construct adjoint network;
5     for node j in violation node set{
6       Apply excitation at node j and solve adjoint network;
7       Convolute waveforms and compute all s_ij;}
8     Magnify all s_ij according to current violation stage ;
9     Formulate LP problem and Solve the LP problem;
10    while (LP solver finds problem infeasible {
11      Increase s_ij solve the relaxed LP problem again;}
12    Update decap with output from LP solver;
13    Solve circuit with updated decap, update violation node set;}
```

Figure 2. SLP-BASED DECAP BUDGETING ALGORITHM .

The number of the outer iteration is dependent on the convergence of SLP, which has a quadratic convergence rate. Indeed we observe that SLP usually converges in several iterations for most of our experiment circuits. Within each of this iteration, a total number of (1+violation node number) simulations are to be performed. The first 1 denotes simulation for the normal circuit and the rest are simulation of adjoint network with different excitations.

The time complexity of the algorithm depends on the number of violation nodes. It is efficient given a small number of violaton nodes, which usually is the case for a well designed P/G grid. Also violation node number often decreases quickly as optimization goes on and it's not uncommon to find there's only several nodes at later iterations.

For extremely large P/G networks with many violation nodes, a hierarchical decap budgeting method can be developed. Basically, the circuit is partitioned into several blocks, each with a relatively small number of violation nodes. Note that in hierarchical decap allocation, boundary conditions need to be taken care of explicitly. The SLP algorithm presented here can be readily fit into hierarchical analysis like the work in [11]. Since the SLP-based decap algorithm usually yields smaller budgets than other methods, combination of hierarchical analysis and the SLP-based decap algorithm presented here can be expected to be both fast and efficient in decap allocation. In this paper we only focus on the decap budgeting algorithm itself and won't go to further details about hierarchical methods, but we note that for a given circuit, the hierarchical strategy usually yields similar or even smaller decap budgets than the flat method [11].

5 Experimental Results

Based on the proposed SLP-based optimization algorithm outlined in Fig. 2, we implemented our prototype decap budgeting tool in C++. All experiments are carried out on a Linux workstation with dual 1.6GHz AMD Althon CPUs and 2G memory.

All P/G circuits are generated by the authors with realistic parameters for R, C and current sources based on industry designs. Since our approach is general enough, those P/G circuits are enough for evaluation purpose too.

For fair and reasonable comparison between the conjugate gradient (CG) and sequence of linear programming (SLP) based optimization algorithms, we modified [8] by explicitly trying to bracket the minimum before each line search, thus avoiding the problem mentioned in section 2. If the bracket fails, α is adapted correspondingly and the bracket is tried again, and so on. Incorporating this step ensures the following line search effective and more importantly, makes the algorithm robust enough for all examples. Bracketing the minimum avoids useless line searches and would only improve efficiency of the CG algorithm.

We tested P/G networks of different sizes and results are summarized in Table 1. Column 1, 2, 3 represent circuit name, total node number, and violation node number respectively. Parameters including voltage drop tolerance, maximum decap at each node can be specified by users and are

Table 1. Comparison with CG method for P/G decap allocation

Circuit	#nodes	#vio nodes	CG			SLP			decap budget SLP/ CG
			decap	iter	time(s)	decap	iter	time(s)	
ckt1	185	4	4.08e-7	7	62	1.66e-7	2	4	40.7%
ckt2	553	15	2.40e-6	12	113	1.08e-6	7	11	45.0%
ckt3	1720	34	3.53e-7	10	158	8.50e-8	7	81	24.1%
ckt4	6105	260	3.70e-8	6	544	1.15e-8	1	198	31.8%
ckt5	14840	592	3.98e-8	7	1545	1.58e-8	5	1996	39.7%
ckt6	51360	126	4.83e-8	2	630	1.07e-8	1	754	22.2%

the same for CG and SLP. The last column compares the allocated decaps by SLP and CG.

For all these circuits, violation elimination requirement was achieved after decap placement. We compared decap area, iteration number and optimization time.

From Table 1, we found that to achieve the same requirement, less decap budget is found by SLP optimization. For some circuits like *ckt3*, *ckt6*, the SLP budget is only about one fourth of that from CG.

This is not surprising, since SLP is based on more information: adjoint method yields sensitivity of violation at every node with respect to every decap, while merged adjoint method only calculates the summed sensitivity of violation with respect to each decap.

It can be seen that for P/G networks up to 10^5 nodes with less than 5% violation nodes, SLP-based optimization can be faster than the merged time adjoint method based CG optimization. The speed advantage gained in sensitivity calculation by merged adjoint method is offset by the difficulty with finding a suitable α to satisfy the convex property of the objective function on its domain. The slow convergence inherent with CG optimization can also be observed by comparison of iteration times.

Both searching for α and more iterations involve extra line searches, which are expensive in decap optimization. Each line search step involves a complete transient simulation process of a circuit with different decap values, while simulation for an adjoint circuit with different excitation only needs forward substitution when LU decomposition of the original circuit is reused.

If method in [8] is directly implemented without the aforementioned α adjusting step, more line searches would be wasted for a probably monotonous objective function (3), and if unfortunately, (3) is a monotonously increasing one on its domain, CG would just refuse to make any step forward and optimization fails.

It can be observed from Table 1 that for large circuits with more than 10^5 nodes, the speed advantage of SLP diminishes because of the increasing number of violation nodes, each corresponding to an adjoint circuit to be simulated. Thus for truly large circuits, we will apply the *divide and conquer* strategy, i.e., partitioning the large circuit into a number of small circuits and optimize them individually. Notice that directly optimizing very large P/G grids by CG method is also prohibitive as transient simulations of the entire large P/G circuits, which already is very slow or even infeasible, are carried out at the internal optimization loop.

6 Conclusion and Future Work

This paper proposed a novel and efficient algorithm for allocating on-chip decoupling capacitors. The new algorithm is based on sequence of linear programmings. Compared to existing conjugate gradient optimization and other nonlinear programming methods, sequence of linear programming demonstrates a faster convergence and much better optimization quality. For P/G circuits without too many violation nodes, which usually is the case for well designed P/G grids, the proposed algorithm is found faster than algorithms based on the time domain merged adjoint method. Since we explicitly optimize decap area, the proposed algorithm yields a smaller decap budget than existing sensitivity based decap optimization algorithm as demonstrated by experimental results.

For very larger P/G grids, partitioning based strategy can be employed to achieve higher efficiency. Since the SLP-based decap algorithm usually yields smaller budgets than other methods, combination of hierarchical analysis and the SLP-based decap algorithm presented here can be expected to be both fast and efficient in decap allocation.

References

[1] G. Bai, S. Bobba, and I. N. Hajj, "Simulation and optimization of the power distribution network in VLSI circuits," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, 2000, pp. 481–486.

[2] S. Bobba, T. Thorp, K. Aingaran, and D. Liu, "IC power distribution challeges," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, 2001, pp. 643–650.

[3] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI chip design," in *Proc. Design Automation Conf. (DAC)*, 1997, pp. 638–643.

[4] S. W. Director and R. A. Rohrer, "Automated network design – the frequency-domain case," *IEEE Trans. on Circuit Theory*, vol. 16, no. 3, pp. 330–337, Aug. 1969.

[5] ——, "The generalized adjoint network and network sensitivities," *IEEE Trans. on Circuit Theory*, vol. 16, no. 3, pp. 318–323, Aug. 1969.

[6] R. Downing, P. Gebler, , and G. Katopis, "Decoupling capacitor effects on switching noise," *IEEE Trans. on Components, Hybrids, and Manufacturing Technology*, vol. 16, no. 5, pp. 484–489, Aug. 1993.

[7] P. Feldmann, T. V. Nguyen, S. W. Director, and R. A. Rohrer, "Sensitivity computation in piecewise approximate circuit simulation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 2, pp. 171–183, Feb. 1991.

[8] J. Fu, Z. Luo, X. Hong, Y. Cai, S. X.-D. Tan, and Z. Pan, "A fast decoupling capacitor budgeting algorithm for robust on-chip power delivery," in *Proc. Asia South Pacifi c Design Automation Conf. (ASP-DAC)*, Jan. 2004, pp. 505–510.

[9] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins University Press, 1989.

[10] Y.-M. Lee, J.-L. Tsai, and C. C.-P. Chen, "Simultaneous area minimization and decaps insertion for power delivery network using adjoint senstivity analysis with ieks method," in *Proc. of VLSI Design/CAD Symposium*, 2003.

[11] H. Li, Z. Qi, S. X.-D. Tan, L. Wu, Y. Cai, and X. Hong, "Partitioning-based approach to fast on-chip decap budgeting and minimization," in *Proc. Design Automation Conf. (DAC)*, June 2005, pp. 170–175.

[12] M. Pant, P. Pant, and D. Wills, "On-chip decoupling capacitor optimization using architectural level current signature prediction," in *Proc. IEEE Midwest Symp. Circuits and Systems*, 2000, pp. 772–775.

[13] C. Paul, "Effectiveness of multiple decoupling capacitors," *IEEE Trans. on Electromagnetic Compatibility*, vol. 34, no. 2, pp. 130–133, May 1992.

[14] L. T. Pillage, R. A. Rohrer, and C. Visweswariah, *Electronic Circuit and System Simulation Methods*. New York: McGraw-Hill, 1994.

[15] C. K. S. Zhao, K. Roy, "Decoupling capacitance allocation and its application to power-supply noise-aware floorplanning," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 1, pp. 81–92, Jan. 2002.

[16] L. Smith, "Decoupling capacitor calculations for cmos circuits," in *Proc. IEEE Topical Meeting of Electrical Performance of Electronic Packaging*, 1994, pp. 101–105.

[17] H. Su, S. S. Sapatnekar, and S. R. Nassif, "Optimal decoupling capacitor sizing and placement for standard cell layout designs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 4, pp. 428–436, April 2003.

[18] X.-D. Tan, C.-J. Shi, D. Lungeanu, and J.-C. Lee, "Reliability-constrained area optimization of VLSI power/ground networks via sequence of linear programmings," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 12, pp. 1678–1684, Dec. 2003.