Fast Decap Allocation Algorithm For Robust On-Chip Power Delivery

Zhenyu Qi, Hang Li, Sheldon X.-D. Tan,
Lifeng Wu†, Yici Cai‡and Xianlong Hong‡
Department of Electrical Engineering, University of California, Riverside
Riverside, CA, 92521, USA
†Cadence Design Systems Inc. San Jose, CA 95134, USA
‡Department of Computer Science and Technology, Tsinghua University,
Beijing, 100084, China.

Abstract

Adding on-chip decoupling capacitors (decaps) is an effective way to reduce voltage noise in power/ground networks and ensure robust power delivery. In this paper, we present a fast decap allocation algorithm, which is able to confine the voltage fluctuations below user specified threshold by adding decaps in an area efficient way. The new algorithm adopts the recently proposed time-domain adjoint network method for sensitivity calculation. To avoid the time consuming line search at each iteration in conjugate gradient method, we proposed a simple, yet efficient search step computation method to accelerate the optimization process. The experimental results show that the proposed algorithm is at least 10X faster than the fastest conjugate gradient method reported so far with similar optimization results.

1 Introduction

With increasing integration density and soaring clock frequency, reliable on-chip power supply becomes a critical concern for VLSI chip design. To retain signal integrity in power/ground (P/G) networks, which supplies power from the power/ground pads to all modules on a chip, extra design effort is required to reduce the voltage noise. 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 a certain percentage (like 10%) of the nominal supply voltage.

Adding decoupling capacitors is an effective way to reduce the ΔI noise in package as shown before [6, 11]. 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 have been introduced and widely adopted for robust on-chip power supply [13, 3, 15]. However, the problem of decap budgeting can be challenging. On one hand, since the value of an on-chip capacitor is linearly proportional to its on-chip area, decap placement is strictly limited by available on-chip white space. Minimization of decap area is equivalent to minimization of the total decap budget. Moreover, 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]. On the other hand, adding decap is essentially a post-processing step of P/G network design and is often expected to be fast even for large circuits. But since decap budget optimization involves repeated simulation of circuits, this step can be quite time-consuming when circuits grow very large [8].

In this paper, we propose an efficient decap allocation algorithm, which is based on the recently proposed time-domain merged adjoint network method [8] for sensitivity calculation and a binary search strategy to reduce the decap budget, subject to the voltage drop and other design rule constraints. Although we will only address decap optimization for voltage drop here, ground bounce can be dealt with in a very similar fashion with little change. Theoretical analysis indicates speed advantage of the new decap allocation algorithm compared to existing methods. Experimental results show that it is indeed significantly faster than existing methods like [8] with similar decap budgets, and can be applied to large circuits directly.

The rest of this paper is organized as follows. The following section briefly reviews existing sensitivity-based decap budgeting algorithms. The new algorithm is introduced in section 3. Aspects of practical implementation and the-

0-7695-2301-3/05 $20.00 (c) 2005 IEEE

oretical analysis regarding to time complexity are also discussed. Experimental results with different circuits are presented in section 4. Section 5 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 [14, 10, 3, 13], the current pattern around 'hot spots'(where excessive voltage drop occurs) is first derived and the amount of electric charge needed to supply that current demand is estimated. To obtain an optimal decap budget, a critical step for these methods is the precise estimation of voltage drops, which unfortunately proves to be difficult for practical P/G networks without simulation. Usually only a lower or upper bound of maximum voltage drop can be obtained [2].

Another category is the sensitivity based optimization [15, 1, 9], in which the adjoint network method [5, 4] can be applied to calculate sensitivity:

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

where c_i is the decap added at node i, n is the number of nodes where decaps can be added. For simplicity we assume all nodes are candidates for adding decap, thus n equals the circuit node number and this assumption applies to the rest of this paper.

$$g_j(c_1, ..., c_n) = \int_0^T \max(V_{min} - v_j(t), 0)dt \tag{2}$$

is defined as the *violation area* at node j. V_{min} is the tolerance of voltage drop, which can be specified by users. $v_j(t)$ is the transient voltage at node j. So the sensitivity defined in (1) measures how fast violation area at node j changes with respect to decap added at node i.

Violation area is shown to be a good metric for voltage noise [15]. Indeed, [15] chooses the total violation area in circuit as the objective function and computes sensitivity of violation area at each node to each decap. A Lagrangian function is formulated and then fed into a quadratic programming solver. The iteration goes on until violation is eliminated. A problem with this method is that decap area is not explicitly minimized but only taken as a constraint, which may lead to overestimated decaps. Moreover, sensitivity computation with the adjoint network method [5, 4] is quite expensive in this case. Two simulations are required s_{ij} for each node j, one for the normal circuit and one for the adjoint circuit with the same size. Ignoring overhead from the quadratic programming solver, the time complexity is approximately $O(2n^{1.5}lmh)$, where m is the number of violation nodes, l is number of sampling time points, and h number of optimization iterations. The term h depends on convergence rate of the optimization method, and $n^{1.5}$ is the typical time complexity for solving sparse matrices [17].

Recent work [8] improved the sensitivity based decap allocation algorithm by introducing the time domain merged adjoint network method and explicitly minimizing decap area by incorporating it into the objective function. In [8], the merged adjoint network method computes sensitivity directly for the objective function, instead of computing for individual nodes and summing them up. The conjugate gradient (CG) method is applied 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 extra decaps added at each iteration. Here violation and decaps are both incorporated into the objective function and will be optimized by CG at the same time. But such balance can be misleading for optimization. Since the direction of optimization is not decided by the sensitivity of violation with respect to decap, but the gradient of (3) in which α plays an important part. An improper α would inflate an actually unimportant part and renders optimization less efficient. In practice, selection of α is critical. If a chosen α happens to make the objective function (3) monotonously increasing, optimization stagnates and no extra decap will be added. Essentially this α is too small, favoring the first part of (3) too much. Unfortunately, how to select a α is not mentioned in [8].

3 Proposed Method

In this section, we show how we can derive a more robust and much faster method for decap budgeting, based on the improved objective function.

3.1 Problem Formulation

To avoid the inherent ambiguousness in (3), we still follow [15] and choose total violation area as the objective function.

Objective function

$$\min \sum_{j=1}^{m} g_j(c_1, ..., c_n) \tag{4}$$

Maximum decap constraint

$$(c_i) \leq d_i, \quad d_i \geq 0 \tag{5}$$

where d_i is the maximum decap allowed at node i, a parameter decided by the available white space around node i. There may be other power integrity constraints like current density for electro-migration. 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.

From the Weierstrass theorem in optimization theory [**?**], (4) is defined on a non-empty compact set and is guaranteed to have a maximum and a minimum. Intuitively they correspond to cases when no decap is added and when all decaps are added to their limit. However, our real objective is to eliminate violation with the user specified threshold. So in fact, it still remains to decide whether a solution exists.

As in [15], minimization of decap area is not incorporated in our problem formulation either, but it will be implicitly optimized with binary search as will be shown later.

3.2 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 is based on definition of sensitivity and involves construction of a sensitivity circuit, and the adjoint method [5, 4], which is deducted from Tellegen's Theorem and involves construction of an adjoint circuit [12, 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.

In particular, we consider sensitivities of violation area at a single node j with respect to all decaps $c_1, c_2, ..., c_n$. The performance function is

$$g_j(c_1, ..., c_n) = \int_0^T \max(V_{min} - v_j(t), 0)dt \quad (6)$$

Following adjoint method,

$$s_{ij} = \int_0^T v'_{i,j}(T-t) \times \dot{v}_i(t)dt, \quad (i = 1, 2, ..., n) \quad (7)$$

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 (6) is already in integration form, excitation of the adjoint network can be immediately derived:

$$I_j(t) = \begin{cases} \frac{\partial(V_{min} - v_j(t))}{\partial v_j(t)} = -1 & \text{if } v_j(t) < V_{min}; \\ 0 & \text{else.} \end{cases} \quad (8)$$

which is a unit step waveform between the start and end of violation at node j. Notice that there may be more than one violation intervals for a node.

Sensitivity of violation area at another node k with decaps can be derived in the same way.

$$s_{ik} = \int_0^T v'_{i,k}(T-t) \times \dot{v}_i(t)dt, \quad (i = 1, 2, ..., n) \quad (9)$$

where $v'_{i,k}(T-t)$ is calculated from an adjoint network with excitation at node k:

$$I_k(t) = \begin{cases} \frac{\partial(V_{min} - v_k(t))}{\partial v_k(t)} = -1 & \text{if } v_k(t) < V_{min}; \\ 0 & \text{else.} \end{cases} \quad (10)$$

Notice that from the principle of adjoint network construction, the adjoint networks for calculating s_{ik} and s_{ij} are exactly the same in topology and parameters except the excitation $I_k(t)$ and $I_j(t)$. We have

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

for all i. But from the superposition property of linear systems, $(v'_{i,j}(T-t) + v'_{i,k}(T-t))$ can be obtained directly by simulating an adjoint network with both excitation $I_j(t)$ and $I_k(t)$ present. More generally,

$$\sum_{j=1}^{m} s_{ij} = \int_0^T (v'_{i,all}(T-t)) \times \dot{v}_i(t)dt \quad (i = 1, 2, ..., n) \quad (12)$$

where $v'_{i,all}(T-t)$ is calculated from an adjoint network with excitations at all violation nodes present. Indeed, this is the basic idea of time domain merged adjoint network method in [8].

The merit of this method can be observed immediately if we notice

$$\sum_{j=1}^{m} s_{ij} = \frac{\partial \sum_{j=1}^{m} g_j(c_1, ..., c_n)}{\partial c_i} \quad (13)$$

So with the time-domain merged adjoint network method, sensitivities of the objective function with respect to all the decaps can be calculated in two simulations, one for the original network and one for the merged adjoint network with all applicable excitations present.

In fact, just as the adjoint method can be adapted for different forms of performance functions, the merged adjoint method can also be adapted for different objective functions, like(4) used here and(3) used in [8]. We only need to reformulate the excitations.

3.3 New Efficient Search Step Computation

Although the merged adjoint method can significantly reduce the number of transient simulations, but a number of

transient simulation of the whole P/G grids are still required at each step in the conjugate gradient based method. The reason is that direct application of conjugate gradient optimization like [8] requires a number of line searches to compute the best search step at each step. In each line search, a number of transient simulations are performed to evaluate the objective function at different points along the current gradient direction, which is very expensive.

To avoid costly line search at each step, we develop a simple, yet efficient, search step computation method. The method is based on the observation that the step size in each search direction can be simply determined by computing the maximum decap value allowed on one or some nodes under this search direction (i.e. those nodes are added to their maximum decap values under the given search direction). If the violation are still present after this iteration, we continue the process until all the violation are gone.

If all the violations are removed after we add the maximum decaps at some nodes afer one or more iteration, we may add more decaps than necessary. To alleviate this problem, a simple binary search will be employed to find the best step where violations are just removed and decap areas are minimized. Notice that we only need to do one binary search in the entire optimization process, which is in constrast to the previous CG based method, where linear search is performed at *every* step. Experimental shows that such simple decap stretegy leads to X10 speedup over the previous method at mild increase of decap areas.

3.4 Some Practical Considerations

If no controlled source exists in the original network, which is true for P/G grids modeled as RLC linear networks, the adjoint network is a passive circuit. For some period in the beginning there's usually no excitation in the adjoint network, thus all node voltages remain zero and no simulation or convolution is needed in (12). From (8), we know this period is $[T, t_e]$ where t_e is the end of the *latest* violation interval among *all* violation nodes. So (12) can be modified into a more efficient version:

$$s_{ij} = \int_0^{t_e} v'_{i,j}(T-t) \times \dot{v}_i(t) dt \qquad (14)$$

At a first glance we need to store waveforms both for original and adjoint network for the above convolution. Basically we follow [15] and store waveforms with piecewise approximation. In this way the dimension of each waveform matrix to be stored is $n \times l$. Recall that n is circuit node number and l is the number of sampling time points, $n \times l$ grows very quickly with circuit size or decreased time step needed for fast changing waveforms. However, if the convolution is carried out reversely after each time step as simulation for the adjoint network goes from t_e to 0, no waveform needs to be stored for the adjoint network, resulting in a great memory cut.

Moreover, for numerical consideration, data scaling is needed, for sensitivities, decaps and voltage changes can be very small. Also as indicated in [16], power networks should be converted to ground networks by the following rules:

1. short-circuit all *VDD* pads to the ground,
2. inverse the directions of all independent current sources.

3.5 Flow of The Proposed Algorithm

With previous discussion, our decap budgeting algorithm can be summarized in the following:

```
DECAPBUDGETCG(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 merged adjoint network;
5     Solve merged adjoint network, convolve for sensitivities;
6     Compute the next conjugate gradient direction;
7     Obtain the maximum step along the current direction;
8     Update all decaps with this step};
9   while (violation requirement satisfied) {
10    Halve the step to see if violation requirement is satisfied;}
11  Finalize decap budget with the smallest possible step obtained;
```

Figure 1. PROPOSED DECAP BUDGETING ALGORITHM .

3.6 Time Complexity Analysis

The whole algorithm can be separated into two stages. First, existence of solution is checked in the current direction computed by CG. Then a solution is located by binary searches. Within each CG iteration, only two simulations are needed for gradient computation. So the time complexity of the algorithm is about $O(2n^{1.5}l(h+r))$. Again, n is circuit node count, l is the number of time points and h is optimization iteration number. r denotes the number of steps which binary search attempted.

Time complexity of [8] is derived in the original paper and can be rewritten as $O(n^{1.5}lh(2+r'))$, where all variables are as defined above except r', which is the number of line searches in each iteration. Usually h, r' and r are all within the same order of magnitude. So the proposed algorithm can be expected to be much faster as we only do the binary (line) search once and we avoid the hr term in the time complexity.

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

Circuit	#nodes	#vio nodes	CG1 (existing)			CG2 (proposed)			speedup ratio
			decap	iter	time(s)	decap	iter	time(s)	
ckt1	88	29	1.82e-6	8	2.3	2.12e-6	1	0.1	23
ckt2	336	63	3.15e-6	10	15.2	4.16e-6	2	0.8	19
ckt3	1233	143	2.43e-5	10	132	2.62e-5	1	2.4	55
ckt4	12673	1083	3.09e-7	8	1995	3.67e-7	1	54	37
ckt5	89496	592	1.94e-7	5	7241	3.10e-7	1	394	18
ckt6	242600	36501	N/A	N/A	$>$ 15hour	3.08e-7	1	400	N/A

Compared with decap algorithm in [15], the proposed method also lead to a great run time reduction, whose time complexity has been derived in Section 2. The violation node number m multiplied there is orders of magnitude larger than the r added here. Also, the fast convergence of conjugate gradient method has been observed in many other applications.

4 Experimental Results

Based on the proposed algorithm in Fig. 1, 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 1G 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.

We compared our method with [8]. To make comparison possible, we implemented [8] such that before each line search, an explicit attempt to bracket the minimum is made, and if the minimum is found to lie at the start of the line, α is augmented. In this way we avoid the problem mentioned in section 2 and make the algorithm robust enough for all our tests.

We tested P/G networks of different sizes and results are summarized in Table 1, where CG1 denotes the method in [8] and CG2 denotes the proposed method. 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 the same for both methods. The last column compares total optimization CPU times for the two algorithms. For all these circuits, violation elimination requirement was successfully achieved after decap allocation.

Table 1 demonstrate the CPU efficiency of the proposed method over the existing approach. To eliminate violation for the same circuit, the proposed algorithm are usually more than 10X times faster than the method in [8]. The fact that a circuit with 240k nodes can be optimized within 10 minutes is quite impressive. CG1 fails to optimize this circuit within 15 hours. One thing needs to be addressed is that we simulated all those circuits without any simplification. Our transient simulation is based on LU decomposition while a structure level reduction technique was introduced and applied first before optimization in [8].

An interesting observation is that CG1 usually takes more iteration than the proposed method, which often fulfills the task within one or two iterations. Three reasons may account for this difference. First, the α problem mentioned in section 2 often delays CG convergence, since the objective function is always being updated and optimization may be ill guided. In contrast we focus on the violation directly in our objective function and optimization is based on sensitivities directly. Second, [8] as well as most other sensitivity based approaches strictly follow existing optimization techniques trying to find the minimum, while we relax the decap optimization problem (its objective function) to simplify the optimization problem. Finally, in our algorithm exactly one simulation is performed along all optimization directions before the last one, whereas a series of line searches is required in [8], which further explains the difference in speed.

We also notice that the decap area obtained by the proposed method is slightly larger than the CG1 method. But the increase in the decap areas are not significant.

The proposed optimization results could be further improved by refining binary searches. A user defined threshold may be introduced to decide whether more binary searches should be made. It's not uncommon in optimization problems that the 90% refining efforts lead to only 10% improvement. Especially in the problem of decap budgeting where objective function evaluation is very expensive, such marginal profits at the expense of time may not be wise. Notice that even with refined binary searches, the time complexity of the proposed algorithm remains unchanged, and the speed advantage can still be expected.

5 Conclusions and Future Works

This paper proposed an extremely fast algorithm for allocating on-chip decoupling capacitors. The new algorithm is based on the time domain adjoint method for sensitivity calculation, and computes gradient of the objective function directly. Compared with the existing approaches, our con-

tribution is a simple, yet efficient search step computation method to avoid the time consuming line search at each iteration in conjugate gradient method. Experimental results demonstrated that the proposed method can lead to at least one order of magnitude speedup over the fastest sensitivity based decap allocation algorithm report so far with similar optimization results.

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 Pacific Design Automation Conf. (ASPDAC)*, Jan. 2004, pp. 505–510.

[9] 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.

[10] 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.

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

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

[13] 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.

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

[15] 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, April 2003.

[16] 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.

[17] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York, NY: Van Nostrand Reinhold, 1995.