A Fast On-Chip Decoupling Capacitance Budgeting Algorithm Using Macromodeling and Linear Programming

Min Zhao, Rajendran Panda, Savithri Sundareswaran, Shu Yan and Yuhong Fu
Freescale Semiconductor, Inc.

ABSTRACT

We propose a novel and efficient charge-based decoupling capacitance budgeting algorithm. Our method uses the macromodeling technique and effective radius of decoupling capacitance to reduce the size of the problem. We formulate the nonlinear optimization into a linear program (LP) by integrating the nodal equations across a time period of interest and through certain approximations. To reduce the error caused by linearization, we do multiple iterations of the linear program. Experimental results demonstrate that, with the proposed algorithm, even very large power networks (eg. 5 million nodes) can be optimized in a couple of hours with 1-2 transient analyses. Comparison of our algorithm with another heuristic method shows area efficiency and run time advantage of our method.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: INTEGRATED CIRCUITS—*layout*

General Terms

Algorithms, Performance, Reliability, Verification

Keywords

Decoupling capacitance, budgeting, macromodeling, sequence of linear programming

1. INTRODUCTION

With the increase in VLSI circuit frequency and supply voltage scaling, designing a robust power distribution network has become a challenging task. Static IR drop is usually addressed through increased metalization (reduction of resistance), pad placement, topology optimization and power density aware floorplanning. However, the chief technique for limiting dynamic voltage fluctuations is to place **decoupling capacitors** (known as **decap**, in an abbreviated manner) close to problem spot. As the fabrication process moves up the technology nodes ladder, the increased wire resistance is aggravating the supply noise problem. However, the high leakage current in these technology nodes discourages addition of abundant decoupling capacitance. Decoupling capacitance also uses up die area and affects die yield adversely. For these reasons, dynamic supply noise needs to be addressed with a minimum amount of decap which needs to be placed optimally. In this paper, we address this important problem. We formulate the problem to minimize the total decap subject to voltage constraints on the network nodes, and constraints on the decap amount that can be added at various places. The solution aims at realizing dynamic voltages better than a user-specified threshold level at all times.

Decap budgeting and placement is a non-linear optimization problem [1]. Recently, several sensitivity-based techniques were proposed. The authors of [1] proposed a sensitivity-based nonlinear program, which is solved by quadratic programming and using a compressed piecewise linear form to store adjoint sensitivity. [2] used the conjugate gradient(CG) method, with a merged adjoint sensitivity heuristic to speed up the sensitivity calculation. [3] reduced the problem size using the geometric multigrid concept and then used a sequential quadratic program on the reduced grid. Its application is thus limited to regular mesh structures. In [4], Li, et.al. used divide-and-conquer to reduce the size of the sensitivity-based optimization. [5] solved the nonlinear optimization through a sequence of linear programming method. Sensitivities to decaps were used as linear constraints in the optimization.

All these methods require calculation of sensitivities with respect to decoupling capacitance location. The adjoint method of sensitivity calculation [6–8] needs to store waveforms at every node in both the original and the adjoint networks, which may exhaust the memory resource for large networks. Moreover, the foresaid methods add decap and recompute sensitivity in an iterative procedure. As the complexity of one adjoint sensitivity computation is the same as one transient analysis, the iterative nonlinear optimization procedure becomes quite expensive.

Charge-based decap estimation techniques have also been proposed [9, 10] in the context of power supply noise-aware floorplanning. An approximate lumped decoupling capacitance is estimated for each floorplan module, assuming full VDD voltage as the initial voltage of decoupling capacitance.

Our approach differs from the previous works in many respects: (1) The charge basis of constraints, formed through integration of the nodal equations over a time period of in-

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

terest, simplifies computations substantially. Our approach first linearizes the charge-based constraints through certain approximations (to be explained further) and then performs a small number of iterations to account for the non-linearity. A small linear program is solved in each iteration. (2) The concepts of macromodeling [11] and effective radius of capacitance are used for problem size reduction. A small macromodel with few ports, which are the potential sites for decap connection, is created by reducing the network. (3) It handles the decap pre-existing in the network and the decap to be added in different ways. Intrinsic decap of devices and interconnect and pre-placed decap cells are modeled inside the macromodel through their companion models placed in the network before the macromodeling procedure. Additional decap that is optimized is external to the macromodel, and is connected at the model's ports.

2. BACKGROUND

2.1 Overview of Power Grid Simulation

A chip's power distribution system is modeled as a linear RLC network with independent time-varying current sources modeling the switching currents of the transistors. Simulating the network requires solving the following system of differential equations, which is formed by a Modified Nodal Analysis (MNA) [12] approach:

$$\mathbf{G} \cdot \mathbf{x}(t) + \mathbf{C} \cdot \mathbf{x}'(t) = \mathbf{b}(t), \tag{1}$$

where $\mathbf{G}$ is a conductance matrix, $\mathbf{C}$ is an admittance matrix resulting from capacitive and inductive elements, $\mathbf{x}$(t) is the time-varying vector of voltages at the nodes, and currents through inductors and voltage sources, and $\mathbf{b}$(t) is the vector of independent time-varying currents and voltages from independent sources and companion models. This differential system is very efficiently solved by reducing it to a linear algebraic system

$$(\mathbf{G} + \mathbf{C}/h) \cdot \mathbf{x}(t) = \mathbf{b}(t) + \mathbf{C}/h \cdot \mathbf{x}(t-h), \tag{2}$$

using Backward Euler (BE) technique. A fixed time step, h, is used to make the LHS matrix stationary so that its factors can be reused in a transient simulation.

2.2 Macromodeling

A part of the power grid or the entire power distribution network can be modeled as a multi-port linear element with current transfer characteristics given by

$$\mathbf{I} = A \cdot \mathbf{V} + \mathbf{S}, \quad \mathbf{I}, \mathbf{V}, \mathbf{S} \in \mathbf{R}^m, A \in \mathbf{R}^{m \times m} \tag{3}$$

where m is the number of ports in the model, A is the port admittance matrix, $\mathbf{V}$ is the vector of port voltages, $\mathbf{I}$ is the vector of currents through the ports from an external network into the model, and $\mathbf{S}$ is a vector of current sources connected between each port and the reference node. $\mathbf{S}$ essentially brings the effect of moving all internal current sources to the ports.

Macromodeling is the procedure of deriving Equation (3) from the modified nodal equations (4) of a power grid, including the intrinsic and already placed decoupling capacitors modeled through their companion models using backward Euler technique.

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{J_1} \\ \mathbf{J_2} + \mathbf{I} \end{bmatrix} \tag{4}$$

where $\mathbf{I}$ is the vector of currents through the interface, and $\mathbf{U}$ and $\mathbf{V}$ are voltages at the internal nodes and ports respectively. G_{12}, G_{11}, G_{22} are parts of the admittance matrix which includes the companion conductance $\mathbf{C}/h$ of equation (2). $\mathbf{J_1}$ and $\mathbf{J_2}$ are current sources at the internal nodes and ports respectively, and include the companion currents $\mathbf{C}/h \cdot \mathbf{x}(t-h)$ of equation (2). The detailed macromodeling procedure is described in [11].

3. DECAP BUDGETING

3.1 Problem Formulation

We set out to minimize the total capacitance to be added at the network nodes, subject to the constraints that (i) the dynamic node voltages are better than a specified threshold, (ii) the capacitance that can be added at a node is bounded, and (iii) the node voltages satisfy the MNA equations.

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_i C_i \\ \text{subject to} \quad & (i)\ V_i^k \geq V_{thre}, \\ & (ii)\ C_i \leq C_{max,i}, \\ & (iii)\ V_i^k \text{ should satisfy MNA at time point } k, \\ & \text{for all power grid nodes i, and time points k} \end{aligned}$$

Here, V_{thre} is specified by the user and $C_{max,i}$ is set depending on local congestion and the decap amount realizable per unit area.

3.2 Flow Outline

Overall flow of the proposed technique is outlined below.

Step 1 Run transient analysis of the power network, with intrinsic capacitances and decoupling capacitances that have already been placed.

Step 2 Inspect the transient voltage waveforms and determine (i) the regions where the voltage constraints are violated (called violation regions), (ii) the time windows during which violations occur (called the violation time windows), and (iii) a set of nodes to be used for macromodeling and optimization (called the sampling nodes). This step is detailed out in Section 3.3. If no violation region is found, then optimization is completed. Otherwise, continue with step 3.

Step 3 For each violation region, determine the optimal amount of decap to be added at the sampling nodes. This is done by first macromodeling the network using the sampling nodes in the violation region as the ports, and then running few iterations of a linear program described in Section 4 and 5 on this model. For violation regions that overlap each other, the optimization for these regions is done simultaneously.

Step 4 Distribute the decap of the sampling nodes evenly into the respective sampling regions and go to step 1.

3.3 Violation Region, Sampling Nodes, and Violation Time Window

A **violation region** is illustrated in Figure 1. First, the die is partitioned into uniformly sized tiles using pre-defined x and y pitches. A set of contiguous tiles (the central circular region plus the darkly shaded region) wherein one or more nodes violate the voltage constraint defines the core violation region. The core violation region is then expanded in all directions by a pre-determined distance (known as the

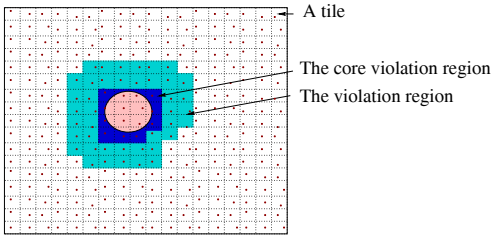

Figure 1: An illustration of the violation region

effective radius) to include additional tiles (lightly shaded tiles surrounding the darkly shaded ones). This expanded region is called the violation region. The expansion is based on the fact that decap added outside the expanded region has little or no impact on the voltages in the core region, and vice versa [1, 4]. Moreover, the expanded region provides more space for decap addition.

The large chunk of network outside the expanded region can be abstracted away by reducing those nodes in the macromodel. There can be more than one violation region, though only one is shown in the Figure 1. By running the optimization independently on different violation regions, we are thus able to reduce the problem size significantly. This also brings in an opportunity for parallelization. If two or more violation regions overlap each other, then these regions are optimized together. Optimizing them separately in such cases will result in sub-optimal results as we will then ignore the strong interaction of capacitances and voltages across these regions.

The effective radius for the decap depends on the effective resistance which is determined by the technology node, the current variation, and the metalization used in each layer. It is very easy to come up with a conservative estimation of how far the core region should be expanded based on one-time experiments with representative designs in a given technology, or through approximate calculations.

Sampling nodes are representative nodes sampled from each tile having a voltage violation. A few (less than 10) nodes per tile with worst voltage violation are sampled to represent the behavior of all nodes in those tiles.We keep the tile small enough for this assumption to be valid. The rest of the nodes are abstracted away by macromodeling. The LP formulation assumes that all the decap is added at the sampling nodes. After solving the LP, however, we distribute the decap at the sampling nodes evenly among the nodes in the tile regions that the sampling nodes belong to.

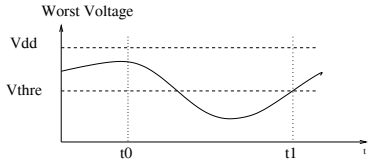

Figure 2: A violation time window $[t_0, t_1]$

The **violation time windows** for a sampling node are determined from the voltage waveform at that node obtained through the dynamic analysis of the power network. A violation time window is marked by 2 time instances: t_0, the time instance of maximum voltage that occurred before a violation, and t_1, the time instance when the voltage recovers back to V_{thre} after a violation. An example of a violation time window is shown in Figure 2. The voltage at t_0 is assumed to be the initial voltage of the decap before discharge. If a violation region has multiple violation time windows, the optimization is done satisfying the charge transfer constraints (discussed in Section4.1) for each of these windows.

4. LP-BASED DECAP BUDGETING

4.1 Charge Transfer Equations of a Macromodel

The **charge** transfer equations corresponding to a violation time window are derived by integrating equation (4) over the violation time window, $[t_0, t_1]$:

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} \begin{bmatrix} \int_{t_0}^{t_1} \mathbf{U}dt \\ \int_{t_0}^{t_1} \mathbf{V}dt \end{bmatrix} = \begin{bmatrix} \int_{t_0}^{t_1} \mathbf{J_1}dt \\ \int_{t_0}^{t_1} \mathbf{J_2}dt + \int_{t_0}^{t_1} \mathbf{I}dt \end{bmatrix}$$

Now the macromodeling procedure as stated in Section 2.2 can be applied to obtain a charge-based macromodel:

$$\mathbf{Q} = A \cdot \mathbf{W} + \mathbf{B} \tag{5}$$

where $\mathbf{Q}$ is $\int_{t_0}^{t_1} \mathbf{I}dt$, A is $G_{22} - G_{12}^T G_{11}^{-1} G_{12}$, $\mathbf{W}$ is $\int_{t_0}^{t_1} \mathbf{V}dt$, and $\mathbf{B}$ is $(G_{12}^T G_{11}^{-1} \int_{t_0}^{t_1} \mathbf{J_1}dt - \int_{t_0}^{t_1} \mathbf{J_2}dt)$. Note that $\mathbf{Q}$ represents the total charge flowing through the ports from the decap to the network during the time period $[t_0, t_1]$ and $\mathbf{W}$ represents the average voltages of the ports multiplied by the time duration $[t_0, t_1]$.

Now our system consists of the network with pre-existing capacitors and inductors as a macromodel, and the decap to be added as external elements connected to the ports of the macromodel, as shown in Figure 3. The problem thus reduces to optimizing the decap added at the ports while providing sufficient charge to pull up the port voltages above V_{thre} during the violation time windows.

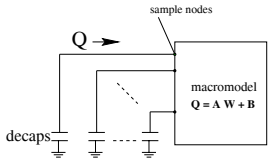

Figure 3: Charge macromodel with decaps

4.2 Voltage Based Constraints

Figure 4 depicts the voltage waveform at a node before and after adding decap. The objective of adding decap is to pull up the voltage above the V_{thre} level as shown in that figure.

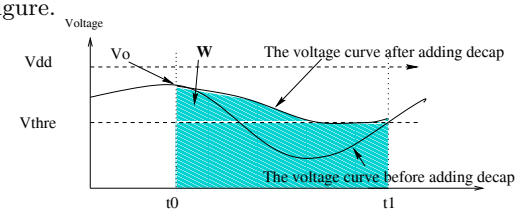

Figure 4: Voltage waveforms with & without decap

Applying the charge equations of the macromodel (5) to the case where decap has been added (i.e. the upper voltage waveform in Figure 4, W is $\int_{t_0}^{t_1} \mathbf{V}dt$, and represents the shaded area shown in that figure. Approximating the upper voltage waveform to a straight line between t_0 and t_1, we can write the voltage constraints as:

$$W_i \geq (V_{o,i} + V_{thre}) * (t_1 - t_0)/2, \ i \in SP, \tag{6}$$

where SP is the set of sampling nodes (ports), W_i is $\int_{t_0}^{t_1} V_i dt$ for the sampling node i, and $V_{o,i}$ is the voltage of the node i at the start of the window.

The above voltage based constraints make another approximation, viz. the V_o obtained from the transient analysis before addition of decap has not changed appreciably with the addition of decap.

In the charge equations of the macromodel (5), Q_i is the charge flow from the decap node i to the macromodel. In order to keep the supply network voltage above V_{thre}, enough charge should be released from the decap. To keep the decap node itself above V_{thre}, the maximum charge that can be released from the decap is $C \times (V_o - V_{thre})$. Therefore, we have the constraints:

$$\mathbf{M} \circ \mathbf{C} \geq A \cdot \mathbf{W} + \mathbf{B} \tag{7}$$

where

$$\mathbf{M} = \begin{bmatrix} V_{o,1} - V_{thre} \\ V_{o,2} - V_{thre} \\ \\ V_{o,m} - V_{thre} \end{bmatrix}, \mathbf{C} = [C_1, C_2, .., C_m]^T$$

where C_i is the decap value at the node i, $V_{o,i}$ is the original voltage of the node i. The $\circ$ operation represents Hadamard's product, i.e, the entry-wise product of vectors $\mathbf{M}$ and $\mathbf{C}$.

4.3 Constraints on Capacitance

The decoupling capacitance has to satisfy the capacitance constraints, which are specified for the tile regions. After we determine the decap at the sampling nodes, we evenly distribute them within the tile regions that the sampling nodes belong to. Therefore, if $C_{max,i}$ is the maximum amount of decap allowed in the tile region i, we will have

$$\sum_{k \in \text{tile } i} C_k \leq C_{max,i} \tag{8}$$

$C_{max,i}$ is decided by the capacitance density possible for a given technology and the amount of white space available in the tile. When a block consists of multiple tiles and the decap placement is used for floorplan purpose, a more complex set of constraints on capacitance can be derived from the floorplan constraints.

4.4 The Complete LP Formulation

Combining the charge transfer constraints (7), voltage-based constraints (6), constraints on capacitance (8), the decap budgeting problem for a violation region can be formulated into a linear programming formulation

$$\begin{array}{ll} \text{minimize} & \sum_{i \in SP} C_i \\ \text{subject to} & \mathbf{M'} \circ \mathbf{C} \geq A \cdot \mathbf{W} + \mathbf{B} \\ & \mathbf{W} \geq \mathbf{L} \\ & \sum_{k \in \text{tile } i} C_k \leq C_{max,i}, \forall \text{tile } i \in \text{violation region}, \end{array} \tag{9}$$

where $m = |SP|$, $\mathbf{C} = [C_1, C_2, .., C_m]^T$,

$$\mathbf{M'} = \begin{bmatrix} V_{o,1} - V_1 \\ V_{o,2} - V_2 \\ \\ V_{o,m} - V_m \end{bmatrix}, \mathbf{L} = \begin{bmatrix} (V_{thre} + V_{o,1}) \times (t_1 - t_0)/2 \\ (V_{thre} + V_{o,2}) \times (t_1 - t_0)/2 \\ \\ (V_{thre} + V_{o,m}) \times (t_1 - t_0)/2 \end{bmatrix},$$

where SP is the set of sampling nodes. If we assume that V_i reaches the threshold voltage at t_1, then V_i is V_{thre} and M' is the same as M given in (7). Note that, in the above formulation, the elements of vector $\mathbf{C}$ are the only variables of optimization.

5. ENHANCEMENTS

5.1 Sequence of Linear Programming

In Section 4.2, we assumed that the maximum charge that could be discharged from a decap i is $C \times (V_{o,i} - V_{thre})$. This is true only if the node's worst voltage at t_1, denoted as V_i, is V_{thre}. Actually, V_i itself is dependent on C. Therefore, constraints (7) should actually be rewritten as

$$(V_{o,i} - V_i) \cdot C_i \geq \sum_{1 \leq k \leq m} A_{ik} W_k + B_i, \forall i \in SP \tag{10}$$

Since both V_i and C_i are variables, constraint (10) is not linear anymore. To handle this nonlinearity, we use the following iterative procedure, wherein the linear programming optimization is solved in each iteration.

1. Set $p = 0$ and the initial value V_i^p based on the transient analysis results. If $V_i^p < V_{thre}$, set $V_i^p = V_{thre}$.

2. Set $V_i^p \rightarrow V_i$, and determine the decap budget C_i^p by solving the LP problem (9).

3. Set $C_i^p \rightarrow C_i$, and get the new node voltage V_i' by substituting W_i by $(V_i' + V_{o,1}) \times (t_1 - t_0)/2$ and then solving the linear equations

$$\begin{bmatrix} (V_{o,1} - V_1') \times C_1 \\ (V_{o,2} - V_2') \times C_2 \\ \\ (V_{o,n} - V_m') \times C_m \end{bmatrix} = A \cdot \begin{bmatrix} (V_1' + V_{o,1}) \times T \\ (V_2' + V_{o,2}) \times T \\ \\ (V_m' + V_{o,m}) \times T \end{bmatrix} + \mathbf{B}$$

where $(t_1 - t_0)/2$ is T. Note that the left hand side of the above equations imply Backward Euler formulation of the decap, C_i. Here, $V_i' \times C_i$ is equivalent to $\mathbf{C}/h \cdot \mathbf{x}(t)$ of equation 2 and $V_{o,i} \times C_i$ is equivalent to the $\mathbf{C}/h \cdot \mathbf{x}(t-h)$ of Equation 2.

4. Update V_i^{p+1}. Here, we set a step length σ to limit the solution space to the neighborhood of V_i^p. If V_i' is below V_{thre}, we set it to be V_{thre}. The procedure can be described as follows.

   if $V_i' - V_i^p$ is satisfactorily small, stop;
   else
   if $V_i' < V_{thre}$, $V_i^{p+1} = V_{thre}$
   else if $V_i' \leq V_i^p - \sigma$, $V_i^{p+1} = V_i^p - \sigma$
   else if $V_i' \geq V_i^p + \sigma$, $V_i^{p+1} = V_i^p + \sigma$
   else $V_i^{p+1} = V_i'$
   $p = p + 1$

Here, we could also use the first order Tayler series approximations of the nonlinear constraints (10) to replace the formulation (9), as done in the traditional successive linear programming (SLP) [13]. Our comparison of the two methods showed the results to be similar. Therefore, we use the above described intuitive, iterative, procedure.

5.2 Relaxation of Capacitance Constraints

During the iterative linear programming procedure, the voltage of the sampling nodes may bounce up and down before settling down. But, if the constraints on capacitance are tight, this may lead the optimization into the infeasible

region without a chance for recovery. Therefore, we added a relaxation factor to relax the capacitance constraints. With this, the objective of LP formulation (9) becomes

$$\text{minimize} \sum_{i \in SP} C_i + \beta \cdot C_{relax}$$

and the constraints on capacitance become

$$\sum_{k \in \text{tile } i} C_k \leq C_{max,i} + C_{relax}, \forall \text{tile } i \in \text{violation region}$$

where C_{relax} is the relaxation factor and β is a weight factor. Generally, β is set to a very large number to force the capacitance constraints to be satisfied. By setting the β, the relaxation factor could also provide a trade-off between the capacitance constraints and the total capacitance required.

5.3 Nonuniform Tiling

When there are multiple violation regions in a network, uniform size tiling may cause problem. Typically, some violation regions occupy a large area with a slow voltage gradient across the region and other violation regions occupy very little area with a large voltage gradient across the region. Given two nodes $n1$ and $n2$, **the voltage gradient** is defined as $V(n1) - V(n2)$ divided by the distance of two nodes. When gradient is large, the voltage and current drawn at two closely located nodes could be quite different. The two nodes then can not be represented well by the sampling node anymore. Therefore, a very fine grain tiling may become necessary. However, the fine grain tiling will cause large violation regions to generate a very large LP problem. In view of this, we use non-uniformed tiling. Fine grain tiling is applied to the small violation regions with high voltage gradient, while coarse grain tiling is used for large violation regions with small voltage gradient.

6. COMPLEXITY ANALYSIS AND EXPERIMENTAL RESULTS

The proposed techniques were implemented as part of an existing in-house power grid analysis tool [14]. An efficient direct linear solver based on Cholesky factorization was used for macromodeling and a public domain linear programming solver, lp_solve [15], was utilized to solve the linear programming problem. All the experiments were performed on Linux machines with 3.2GHz CPU and 1GB-2GB memory.

Table 1: Benchmark Details

Chip	#nodes	Supply vol(v)	Worst Vol(v)	#time points	Analysis CPU(s)
Chip-1	198,396	2.50	2.14	42	7.0
Chip-2	4,868,838	1.65	1.42	62	344.9

The proposed method was benchmarked using global power networks from 2 real designs. The number of nodes, supply voltage, worst voltage, number of time steps simulated and CPU time (in Seconds) for the dynamic analysis are listed in Table 1. Ideally, we would like to compare our results with the existing sensitivity-based methods that could handle very large power networks with non-uniform meshes. However, for example, [4] requires about $2000s$ on a $3GHz$ Linux machine to solve a non-uniform power network of about 1 million nodes. Due to the absence of common benchmarks and the effort involved in implementing the complicated sensitivity-based algorithms within the framework of our tool, we will compare the experimental results with a simpler heuristic method, and also present a complexity analysis of our method.

6.1 Complexity Analysis

The various approximations discussed before usually cause the decap to be under/over-estimated slightly. So, as stated in Section 3.2, we iterate between the optimization and transient analysis verification (steps 2-4) until the voltage constraints are satisfied. From column 11 of Table 2, we can see that constraints are satisfied typically in 1-2 iterations.

For each such (outer) iteration, one sequence of linear programming optimization and one transient analysis are executed. Suppose n, m, l, N and h_l are the number of nodes, sampling nodes, time points, violation regions and linear program iterations respectively. Let $O_{lp}(m)$ be the time complexity of LP. Let $n^{1.5}l$ be the time complexity of the transient analysis, as assumed in [4]. Then the time spent per outer iteration is $n^{1.5}l + n^{1.5}N + N \cdot O_{lp}(m) \cdot h_1$, where $n^{1.5}N$ is time spent on macromodeling of violation regions. Generally the size of samples m is several orders smaller than the original network size, n. Most important of all, by adjusting the tiling granularity and using non-uniform tiling, the size of m can be kept within control, independent of the size of the power grid and the number of the violation nodes. In our implementation, we adjust the tiling such that $m \leq 2000$. Also, we set $h_l \leq 10$. If the voltages of the sampling nodes do not converge within 10 LP runs, we continue with the next step of flow. Overall, the optimization time is limited compared to that of one transient analysis of the entire network, especially for large networks. This can be seen from columns 10 and 11 of Table 2. For example, in case of Chip-2 with 1.485V specification, the time spent on 2 transient analysis runs (one for step 1 and one for step 4 of Section 3.2) is 690s, while the time spent on optimization is only 68s.

6.2 Comparison with a Binary Search based Greedy Method

To benchmark the quality of our LP based decap optimization, we compared the results with another intuitive method which is a greedy method based on binary search. The binary search method places decap uniformly across the core violation region. For each core violation region, the amount of decap placed is decided through binary search. The search begins by placing $0.1ff/um \times area$ capacitance in the violation regions. The capacitance is increased by 10 times in each step until the first successful solution is found. From then on, the binary search is used to determine a minimum capacitance that satisfies the voltage constraints.

Table 2 compares the results of our method with that of the binary search method. The voltage specification and number of violation nodes are shown in columns 2 and 3 respectively. The worst voltage after optimization, amount of decap added, the CPU time, and the number of iterations are given in columns 4-7 for the Binary Search method and in columns 8-11 for our method. The CPU time includes the entire decap budgeting flow as given in Section 3.2, including the pre-optimization transient analysis, sequence of LP, and the post-optimization transient analysis verification. The number of iterations refers to the number of optimization-verification iterations. The total number of transient analysis runs is thus $\#iter + 1$.

Table 2: Comparsion with the Binary Search Method

Chip	Spec Vol(v)	#Vio Nodes	the Binary Search Method				Our Method			
			Vol(v)	Decap	CPU(s)	#iter	Vol(v)	Decap	CPU(s)	#iter
Chip-1	2.25	3137	2.2525	1.18 nF	80.5	12	2.2574	0.84 nF	18.6	1
	2.30	5451	2.3023	3.04 nF	80.5	12	2.3084	2.17 nF	36.8	1
	2.35	10447	2.3500	11.46 nF	80.5	12	2.3522	7.32 nF	28.2	2
	2.40	18936	2.4001	36.96 nF	92.5	14	2.4002	21.01 nF	48.5	2
	2.43	22411	2.4305	105.40 nF	128.1	20	2.4306	64.54 nF	49.9	2
Chip-2	1.485	218	1.4850	4.71 pF	5280.2	13	1.4855	4.05 pF	758.3	1
	1.50	548	1.5000	12.42 pF	5280.2	13	1.5005	12.16 pF	791.3	1
	1.51	14173	1.5100	34.82 pF	5280.2	13	1.5113	21.61 pF	1247.3	2
	1.52	184141	1.5133	242.17 pF	11314.3	30	1.5202	70.48 pF	1771.6	2

From Table 2, we can see that our method always uses less decap than the binary search based method. The larger the violation regions, the more gains our method shows. This is because the proposed optimization places the decap at the most needed location. In addition, our method always consumes much less CPU time. Note that, in order to speed up the greedy method, we did the binary search of all the violation regions simultaneously. Otherwise, the CPU time per iteration will be *#vioRegion* times the CPU time per iteration listed in Table 2. The drawback with the simulatanous binary search is that a capacitance value that originally pulled the voltages up successfully above the threshold level may fail because of the capacitance reduction of the neighboring violation regions. In this case, the binary search has to be reset. Moreover, it is difficult to apply the greedy technique to floorplan or placement purpose due to the inflexibility in placing different decap at different locations of one violation region based on floorplan/placement constraints.

To illustrate how the iterative LP procedure improves the quality of optimization, Table 3 shows the results of the first 4 iterations of 5 violation regions of Chip-1 at 2.40V specification. Column 2 is the ratio of the violation region area to the total area, and column 3 is the number of sampling nodes. The decap added in each of the 4 iterations are reported under columns 4-7. We can see that only one iteration of LP may significantly overestimate the decap value. After a couple of iterations, the decap value becomes stable. In our implementation, we set change in voltage $dv \leq 0.002$ as the condition of convergence. With only 2 LP iterations, the violation regions 3, 4, and 5 converged.

Table 3: The first four LP iterations

Vio region	Area ratio	#Sample nodes	1st (nF)	2nd (nF)	3rd (nF)	4th (nF)
Vio-1	0.0140	185	2.06	1.63	1.54	1.52
Vio-2	0.1611	1534	38.53	20.08	19.14	19.13
Vio-3	0.0021	25	0.14	0.12	-	-
Vio-4	0.0021	25	0.16	0.15	-	-
Vio-5	0.0083	106	0.28	0.28	-	-

7. CONCLUSION

In this paper, we have proposed a novel and efficient charge-based decoupling capacitor budgeting algorithm. We proposed several techniques to reduce the problem size and also to have the problem size scale well w.r.to the network size and the number of violation nodes. We used macromodeling, and the locality effects of the decoupling capacitance, along with non-uniform tiling. By integrating the nodal equations over the violation time period, we replaced the time consuming sensitivity calculations and nonlinear programming optimization with a much cheaper sequence of linear programming procedure. Actually, each individual technique proposed here can be combined with other existing approaches. Experimental results were given to demonstrate that the proposed algorithm is capable of optimizing power networks with 5 million nodes using 1-2 transient analysis runs, and in a couple of hours. Comparison of our algorithm with a greedy heursitic shows the advantage of area efficiency and run time efficiency of our method. By modifying the capacitance constraints, the method can easily be incorporated into physical design flows.

8. REFERENCES

[1] H. Su, S. S. Sapatnekar, and S. R. Nassif, "An algorithm for optimal decoupling capacitor sizing and placement for standard cell layouts," in *ISPD*, pp. 68–73, 2002.

[2] 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 *ASP-DAC*, pp. 505–510, 2004.

[3] K. Wang and M. M. Sadowska, "On-chip power supply network optimization using multigrid-based technique," in *DAC*, pp. 113–118, 2003.

[4] H. Li, Z. Qi, S. X. D. Tan, L. Wu, Y. Cai, and X. Hong, "Partitioning-based approach to fast on-chip decoupling capacitor budgeting and minimization," in *DAC*, pp. 170–175, 2005.

[5] Z. Qi, H. Li, J. Fan, S. X. D. Tan, Y. Cai, and X. Hong, "On-chip decoupling capacitor budgeting by sequence of linear programming," in *Proceedings of 6th International Conference on ASIC*, 2005.

[6] S. W. Director and R. A. Rohrer, "The generalized adjoint network and network sensitivites," *IEEE Transactions on Circuit Theory*, vol. 16, pp. 318–323, Aug. 1969.

[7] P. Feldmann, T. V. Nguyen, S. W. Director, and R. A. Rohrer, "Sensitivity computation in piecewise approximate circuit simulation," *TCAD*, vol. 10, pp. 171–183, Feb. 1991.

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

[9] S. Zhao, K. Roy, and C. K. Koh, "Decoupling capacitance allocation for power supply noise suppression," in *ISPD*, 2001.

[10] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI chip design," in *DAC*, pp. 638–643, 1997.

[11] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. T. Blaauw, "Hierarchical analaysis of power distribution networks," *TCAD*, vol. 21, pp. 159–168, Feb. 2002.

[12] C. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, vol. CAS-22, no. 6, pp. 504–509, 1975.

[13] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming*. John Wiley and Sons, Inc., 1993.

[14] A. Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuianu, and D. Bearden, "Design and analysis of power distribution networks in PowerPC microprocessors," in *DAC*, pp. 738–743, 1998.

[15] M. R. C. M. Berkelaar and et. al., "LP SOLVE 5.5 Users' Manual," 2005.