Partitioning-Based Approach to Fast On-Chip Decap Budgeting and Minimization *

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

ABSTRACT

This paper proposes a fast decoupling capacitance (decap) allocation and budgeting algorithm for both early stage decap estimation and later stage decap minimization in today's VLSI physical design. The new method is based on a sensitivity-based conjugate gradient (CG) approach. But it adopts several new techniques, which significantly improve the efficiency of the optimization process. First, the new approach applies the time-domain merged adjoint network method for fast sensitivity calculation. Second, an efficient search step scheme is proposed to replace the time-consuming line search phase in conventional conjugate gradient method for decap budget optimization. Third, instead of optimizing an entire large circuit, we partition the circuit into a number of smaller sub-circuits and optimize them separately by exploiting the locality of adding decaps. Experimental results show that the proposed algorithm achieves at least 10X speed-up over the fastest decap allocation method reported so far with similar or even better budget quality and a power grid circuit with about one million nodes can be optimized using the new method in half an hour on the latest Linux workstations.

Categories and Subject Descriptors

T5.2 [**Signal Integrity and Reliability Analysis**]

General Terms

Simulation, Algorithm, Optimization

Keywords

Decoupling Capacitor, IR drop, On-Chip Power/Grid Networks

1. INTRODUCTION

Power integrity has become the most insidious issue in nowadays deep sub-micron and nanometer VLSI regime. IR drop is caused by device switching current flowing through the parasitic power/ground network. A supply voltage (VDD/GND) variation throughout the entire chip will occur, which will lead to adverse impact on chip performance, longer path delay, or even logic failure. With the reduced noise margin and increased switching frequency as technology scales, it is required to confine the supply voltage fluctuation within a certain range of the nominal VDD value to guarantee the reliable power delivery. For removing dynamic IR drop which arises from resistive and inductive effect, intentionally adding decoupling capacitance (decap) between power and ground buses or between power/ground buses and substrate is the most efficient way. As shown in Fig 1, decaps provide a reservoir of current that is instantly available for the near switching components to remove spikes and glitches in the power rail. Since on-chip decaps are typically manufactured using MOS transistors, and excessive on-chip decaps could cause more leakage power, low yield and lower resonant frequency [1], the total decap area should be added in an area-efficient way.

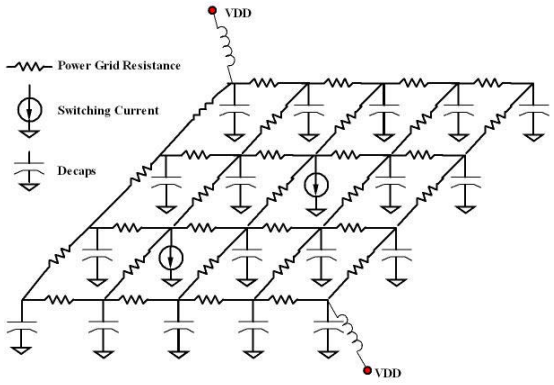

Figure 1: Model of Power Grid Network

Budgeting decap in an area-efficient way, however, is a difficult task due to prohibitive analysis costs of P/G networks with millions of nodes and extracted on-chip and off-chip RLC components in modern VLSI design. Mathematically, optimal decap allocation is a nonlinear optimization problem and many existing approaches [11, 5] use sensitivity-based optimization methods to solve the problem. To compute the sensitivity, transient simulations of the whole P/G networks have to be carried out at every optimization step. Given the fact that the transient simulation of P/G networks with millions of nodes is already an extremely time-consuming task, the CPU and memory cost of the optimization method that uses transient simulations as the internal loops will be prohibitive. Recent study [5] shows that allocating decaps for a P/G grid with about one million nodes will take about 10 hours in modern workstations with improved simulation techniques. Given the

*This work is funded by NSF CAREER Award CCF-0448534, UC Micro #04-088 and Cadence Design System Inc. The work of Y. Cai and X. Hong is funded by Hi-Tech Research & Development (863) Program of China 2004AA1Z1460 and the National Natural Science Foundation of China (NSFC) 60476014.

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

increasing sizes of P/G networks, existing decap budgeting techniques do not scale well for future VLSI on-chip power distribution network design and verification.

In this paper, we propose a fast decap allocation and budgeting algorithm in solving large power grid circuit, which is modeled as RLC networks considering both on-chip and off-chip parasitics. The new method is based on the sensitivity-based conjugate gradient methods. Our contributions include: (1) The proposed algorithm applies the time-domain merged adjoint network method for sensitivity calculation, in which the computations of each individual node's sensitivity can be combined together and the sensitivity of the objective function with respect to every decap value can be obtained by two transient simulations; (2) Instead of doing line search at every step in conjugate gradient optimization, we develop a new search step scheme to speed up the optimization process; (3) Based on the local effect of adding decap to reduce IR drops, we partition the whole circuit into a number of sub-circuits and optimize them individually. A noise-aware partition scheme is proposed to perform the required partitioning. The combination of our new optimization algorithm and partition scheme significantly improves the analysis speed for extremely large circuits even on a single CPU workstation.

The rest of the paper is organized as follows: Section 2 briefly reviews existing sensitivity-based decap budgeting algorithms. In Section 3, we describe our new algorithm, with theoretical analysis regarding to time efficiency. Section 4 gives description of a graph-based partition algorithm in dealing with special decap situation. The experimental results are summarized in section 5 to verify our method, with conclusions in section 6.

2. REVIEW OF PREVIOUS DECAP OPTIMIZATION ALGORITHMS

Existing on-chip decap budgeting algorithms basically fall into two categories. In [2, 7, 10, 9], the current pattern around *hot spots* (where violation of IR drop occurs) is 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.

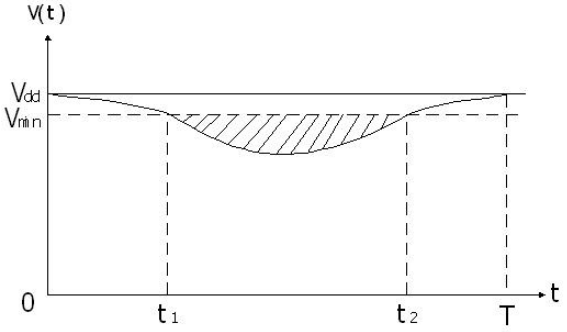

Figure 2: Illustration of IR drop violation

Another category is sensitivity-based approaches depended on actual circuit simulation, which is the more accurate and well-accepted method. Fig 2 gives an illustration of VDD fluctuation of a node within one clock cycle [11]. The *violation area* at node j is defined as:

$$g_j(c_1,...,c_n) = \int_0^T \max(V_{min} - v_j(t), 0)dt \tag{1}$$

which equals to the shaded area below a certain VDD threshold in the graph. The *sensitivity* of decap added at node i in contributing to remove this violation area at node j is:

$$s_{ij} = \frac{\partial g_j(c_1,...,c_n)}{\partial c_i} \tag{2}$$

where c_i is the decap value added at node i, and n the number of nodes where decaps can be added. If we allow all circuit nodes to be candidates for adding decaps, n equals to the number of circuit nodes. The adjoint network method is the conventional way to calculate this sensitivity, which requires two full circuit simulations, one for the original network and the other for the adjoint network. The sensitivity for capacitive elements can then be expressed as:

$$s_{ij} = \int_0^T v'_{i,j}(T-t) \times \dot{v}_i(t)dt, \quad (i = 1,2,...,n) \tag{3}$$

where $\dot{v}_i(t)$ is the derivative of voltage waveform at node i in the original network, and $v'_{i,j}(T-t)$ is the waveform at node i in the adjoint network under unit step current excitation at violation node j.

In previous work [11], the total violation area is chosen as objective function, and optimized by a quadratic programming solver. The iteration continues until all violations are eliminated. In order to obtain the gradient value of the objective function to each decap value, sensitivity of every violation node with respect to every decap node is computed within each iteration. Therefore the time complexity is approximately $O(2n^{1.5}lmh)$, where m is the number of violation nodes, l the number of sampling time points, and h the number of optimization iterations. The term h depends on the convergence rate of the optimization method, and $n^{1.5}$ is the typical time complexity for solving sparse matrices [12]. It is clear that the speed of this algorithm depends on the number of violation nodes, and will become intolerable for extremely large circuits. In addition, the decap area could be overestimated because it is not explicitly considered in the objective function. Recent work [5] improved the sensitivity calculation by introducing the time domain merged adjoint network method, and decap area is incorporated into the objective function as:

$$(\sum_{i \in N_{deap}} c_i) + \alpha(\sum_{j \in N_{vio}} g_j) \tag{4}$$

where N_{decap} is the set of all the decap nodes and N_{vio} is the set of all the violation nodes. The new objective function is solved by traditional conjugate gradient (CG) method in each optimization iteration. The weighting factor α in (4) is used to balance the two terms, and will keep changing in each CG iteration. However, such balance could be misleading since the optimization direction may be decided by the value of α rather than the sensitivity of each decap value to the objective function. Improper α will cause extra line searches, or even optimization failure. What's more, each line search during CG optimization also requires the evaluation of the objective function at the cost of a full transient simulation, which renders the algorithm inefficient and unreliable.

In this paper, we present our improved conjugate gradient (iCG) algorithm, which takes the advantage of the fast sensitivity computation in [5], while avoids the ambiguousness of the old optimization objective and the inefficient line search phase.

3. IMPROVED CONJUGATE GRADIENT ALGORITHM

In our new iCG algorithm, we formulate the optimization problem as:

Objective function:

$$\min \sum_{j=1}^{m} g_j(c_1,...,c_n) \tag{5}$$

subject to constraints:

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

where m is the number of violation nodes, and n the number of decap nodes. d_i is the maximum decap allowed at node i, a parameter decided by the available layout white space around node i.

Note that the decap budgeting problem defined in Eq.(5) and Eq.(6) can be used in early floor-planning stage for decap budget estimation, where d_i is the available white-space that can be used for adding decaps. The same optimization formulation can also be used in the later stage of physical design to minimize the existing decap budget, since some decaps are placed heuristically in the previous design stage. The d_i then represents the decap value already allocated at node i.

As mentioned in section 2, the sensitivity of the objective function (5) with respect to each decap value can be computed by the merged adjoint network method. In conventional adjoint network, a unit step current source is placed at each violation node when calculating the sensitivity described in Eq.(2). However, since all the adjoint networks for each violation node share the same topology, we may combine the step current sources together based on circuit superposition. And the sum of all the decap sensitivities would turn into:

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

where $v'_{i,all}(T-t)$ is calculated from the merged adjoint network with combined step current sources. The merit of this method can be observed immediately since

$$\sum_{j=1}^{m} s_{ij} = \frac{\partial \sum_{j=1}^{m} g_j(c_1,...,c_n)}{\partial c_i} \tag{8}$$

where the right-hand-side is just the sensitivity of the objective function we need in our problem formulation. Therefore, all the decap's sensitivity to the objective function can be calculated in only two transient simulations, which greatly improves the efficiency of the algorithm.

Although the merged adjoint method can significantly reduce the number of full circuit transient simulations, a number of simulations are still needed at each step in the CG method. The reason is that the direct application of the CG method in [5] requires evaluations of the objective function at different points along the current gradient direction to find the minimum cost in the objective function. But the cost of simulation in this line search phase is expensive, and could offset the efficiency gained by the merged adjoint method.

In order to avoid this, 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. In other words, we determine the maximum step we can take in current direction, and set each decap values according to this step. We will continue to do so until violation criterion has been met. One problem with such a maximum-allowable-step scheme is that it may overestimate the decap areas. To alleviate this problem, a binary search will be performed to find the best step such that all violations are just removed and decap areas are minimized. The main difference of our method from the traditional CG method is that we only need to do binary search just *once* in the entire optimization process, which is in contrast to the traditional CG method, where the line search is carried out at every CG step during optimization.

The new proposed decap budgeting algorithm can be summarized in Fig. 3, which has two stages. First, the existence of solution is checked in the present direction computed by the CG method. Then a solution is located by binary searches. Within each CG iteration, only two transient 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 in which binary search is attempted.

```
DECAPBUDGET ICG (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 the merged adjoint network;
5   Solve the 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 budget with the smallest possible step obtained;
```

Figure 3: Proposed Decap Budgeting Algorithm

Time complexity of [5] is derived in the original paper and can be expressed as $O(n^{1.5}lh(2+r'))$, where all variables are defined as 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 avoid the hr' term in the time complexity. Compared with decap algorithm in [11], the proposed method also leads to a great run time reduction, as it is independent with the number of violation nodes. Also, the faster convergence and less computation cost of conjugate gradient than quadratic programming contributes to the time efficiency as well.

4. PARTITIONING-BASED ANALYSIS OF P/G NETWORK AND DECAP BUDGETING

Accurate decap budgeting relies heavily on repeated transient analysis of the entire power grid. With a full-chip power grid model consisting of millions of nodes and elements, the burden on computation and memory storage during optimization is huge. Recently, some new techniques are published to explore the efficient analysis on huge P/G networks, such as model-reduction techniques in [4], or random walk algorithm in [8]. However, they are either less accurate or only efficient in DC analysis at a certain circuit topology. For decap optimization, which asks for full circuit simulation at each time point, it is desirable to reduce the problem size of analysis by partitioning the large circuit into several smaller sub-circuits, and optimize decap budget of each partition separately, or even in parallel.

The partitioning-based strategy is also supported by the fact that adding decap to remove IR drop violation is a local effect. Normally, the violation occurs in the center or a certain area on the chip. For the emerging flip-chip packaging technology, this local effect becomes more prominent [3]. The number of violation nodes compared to the total circuit node number is also supposed to be small. Otherwise, the entire power network should be re-designed. For each violation node, the most effective decap locations are always the nodes close to it. Thus we should expect only a few good decap candidate locations through the entire chip for each violation node.

4.1 General Partition Algorithm

In our paper, we use the graph-based multilevel minimum cut algorithm for partitioning task [6], which provides an extremely fast speed on large graph sizes, under which a million vertices graph can be processed within one minute. This ensures that the partition phase will not bottleneck our entire partitioning-based optimization flow.

Table 1: Decap Budget Comparison before/after Partition

Subcircuit Name	Original Budget	w/ boundary		w/o boundary	
		budget	deviation	budget	deviation
1	7.48	8.32	11.2%	11.08	48.1%
2	3.38	2.61	-22.9%	5.55	64.0%
17	9.02	11.06	22.6%	11.06	22.6%
18	6.40	8.88	38.8%	10.99	71.7%

A direct use of partitioning without considering the IR drop violation could have adverse impacts on the entire decap optimization. Because we should avoid putting violation nodes as the boundary nodes, whose node voltage can't be reduced by adding decaps, since boundary node voltages are treated as independent voltage sources as discussed below.

4.2 Boundary Condition for Sub-circuits

For each sub-circuit, we need to consider the influence of other sub-circuits on its decap budget optimization. We call this *boundary condition* of the sub-circuit. If boundary condition for each individual partition is not considered, the decap budget will be conservative compared to the one presented in the original netlist. The reason is that the sub-circuit, as opposed to the full circuit, confines currents to flow in a smaller area, instead of the global P/G network. As consequence, more currents (due to smaller resistance to true ground) occur in the same area after partitioning. These over-estimated currents will cause greater IR drop than it actually is.

To solve this problem, we keep the boundary node waveforms in piece wise linear (PWL) form derived from a full circuit transient simulation at the beginning of the optimization. In this way, the voltage waveforms are the same even when each sub-circuit is simulated independently.

In Table 1[1], a circuit with 240K nodes is partitioned into 20 pieces. Decap budgets are compared between several sub-circuits with PWL boundary conditions and the ones without. Deviation is calculated as the difference between the budget of each sub-circuit and the budget of the same sub-circuit in the original circuit optimization. We notice that there are only 4 sub-circuits containing violation nodes, and the deviation of the simple partition budget without the boundary conditions is much greater than the one with boundary conditions.

4.3 Noise-Aware Partitioning (NAP)

Keeping boundary node voltage as independent voltage sources introduces another problem. The violation nodes may appear as the boundary node in each partition, and there is no way to reduce voltage drop of the PWL voltage source at the violation node by adding decaps. Therefore we should explicitly avoid putting violation nodes into the boundary node set during partitioning. This can be easily implemented by assigning a relatively heavy edge weight between each violation node, as well as the nodes adjacent to them. Since the partition algorithm attempts to minimize the total cut weights on the boundary, those weighted violation node edges have a very low possibility to be cut.

Another issue is that we should keep the good decap candidate nodes for violation in the same sub-circuit. Otherwise, it will be more expensive to reduce the IR drops of the violation nodes by using less effective nodes available in a sub-circuit. As a result, we need to consider adding decap *range* for violation nodes during partitioning. The current sources at the violation nodes typically are the cause of IR drop violation, and will be assigned into one sub-circuit along with all the nodes they are directly connected to or in nearby locations. To achieve this, we can assign a relatively small vertex weight to each violation node containing current source, as well as a pre-defined radius, within which the nearest non-violation nodes close to it are also given the same weight. Given the fact that the partitions should be balanced in terms of each partition's total vertex weight, the violation nodes will be aggregated with a host of good non-violation decap candidate nodes.

[1]The decap budget values in this table, as well as in other tables in this paper, are all normalized to the same order for comparison without units.

Table 2: Decap Budget Comparison under NAP

Subcircuit Name	Original Budget	Partitioned Budget
4	17.25	13.84
5	7.16	4.55
14	10.21	10.30

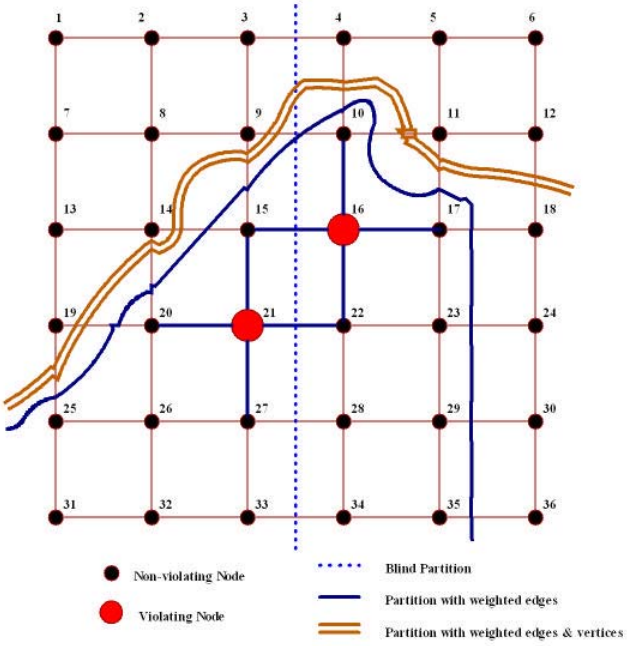

Figure 4: Example of noise-aware partition scheme

Fig 4 gives a simple example of above proposed partition scheme. We observe that the violation nodes 16 and 21 are easily separated into two partitions without any differentiation between them and the non-violating ones. After we add larger weights on the edges around them, they are captured into one partition. If we further assign a smaller vertex weight (as compared to other vertices) to them, as well as the ones in adjacent to them, which are node 10, 15, 17, 20 and 22, more surrounding nodes will go with node 16 and 21 into the same sub-circuit, which is exactly what we expected.

By using this noise-aware partition (NAP) scheme, we did the test on previous 240K circuit again. Results in Table 2 demonstrate that the new partition budget is very close to or even smaller than the one from the original circuit. Further discussions on the quality of the partitioned decap budget is given in section 5.

4.4 Partitioning-based Decap Optimization Flow

The whole partitioning-based optimization flow is given in Fig 5. It performs only two full circuit transient simulations, one at the very beginning to report all the violation nodes and record the original boundary node waveform, another at the very last to verify the optimization result. Compared to many simulations carried out in a flat run mode, the time overhead from these two full circuit sim-

```
PARTITIONING-BASED DECAP OPTIMIZATION
1  Solve input circuit, identify all the violation nodes;
2  Transfer netlist into graph, feed into partition solver;
3  Find all the boundary nodes, extract their PWL form;
4  Generate each partition netlist with PWL information;
5  For each partition{
6     Call iCG solver to do the individual optimization;}
7  Combine all the decap values, generate new netlist file;
8  Solve the new circuit netlist with decaps
9  if(violation criterion met){optimization successful;}
     else{decrease previous partition number, and go back to 2;}
```

Figure 5: Partitioning-Based Optimization Algorithm

ulations becomes less significant. The rest of simulations will be conducted on sub-circuits only, which are faster than full circuit simulation even done sequentially.

We assume a number of N partitions (sub-circuits) with approximately the same amount of nodes in each partition. Since the partitions are processed sequentially in our experiment, and parameters l, h, and r won't change too much with partitioning, the new time complexity becomes

$$N[2(\frac{n}{N})^{1.5} l(h+r)] = \frac{[2n^{1.5} l(h+r)]}{\sqrt{N}} \quad (9)$$

Therefore, the time complexity of partitioning-based optimization algorithm will decrease with the square root of the number of partitions. However, the number should not be very large either. Because a very small sub-circuit could result in no solution of optimization due to reduced decap candidate positions. In case of no solution is found in the sub-circuits, we will halve the partition number, and re-run the whole algorithm again for a relaxed partition area. Since we snapshot the IR drop violation by defining an effective area for adding decap in partitioning, the optimization effect is supposed to be guaranteed when all the partitions are combined together.

If parallel computing is allowed, the time complexity can be further reduced to

$$\frac{[2n^{1.5} l(h+r)]}{N^{1.5}} \quad (10)$$

since there is no communication needed between different partitions during decap optimization.

5. EXPERIMENTAL RESULTS

We implement our proposed algorithm in C++. All experiments are carried out on a Linux PC with dual 3.0Ghz Xeon CPUs and 2GB memory. All test circuits are generated by the authors with realistic parameters for R, C and current sources based on industry designs. The off-chip inductive parasitic effects are also considered. Some figures are exaggerated in order to test the versatility of our algorithm. For each test case, we artificially set the power noise level such that the number of violation nodes presented in the circuit is within 20% range of the total node count. Keep in mind that we can not count solely on adding decaps to eliminate all IR drop violations, and a huge amount of violation is not reasonable for decap solution.

We first compare our method with [5] for small circuits without partitioning. To make comparison possible, we implement [5] in such a way 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.

Table 3 summarizes the comparison, where CG1 denotes the method in [5] and CG2 denotes our iCG method. Columns 1, 2, 3 represent circuit names, total node numbers, and violation node numbers respectively. Parameters including voltage drop tolerance, the maximum decap at each node can be specified by users and are the same for both methods. The last column compares total optimization CPU time for the two algorithms. For all these circuits, violation elimination requirement is successfully achieved after both decap optimization. The CPU time efficiency of the proposed method is usually more than 10 times faster than the method in [5].

We apply our partitioning-based optimization algorithm for larger power grid circuits. As shown in Table 4, comparisons are made between the budget and CPU time of the flat CG1, CG2, and the partitioning-based CG2 algorithm. The circuit sizes range from 240K to 1M. While CG1 is still capable of solving ckt6 and ckt7, it fails to work on 800K and 1M cases. The main reason arises from the memory limitation (we have 2G memory in our Linux workstation) for LU decomposition and waveform storage in sensitivity calculation during each transient simulation. CG2 suffers the same problem when doing the optimization flatly. The partitioning-based algorithm, on the other hand, can handle these cases very easily.

As can be seen from Table 4, the partitioning-based algorithm optimizes all circuits successfully, and the budget achieved is comparable, or even smaller than the flat optimization runs. The time advantage is also impressive. The circuit with one-million nodes can be optimized in about half a hour, as opposed to a 10-hour run time in [5] for the same circuit volume [2]. Another thing that needs to be pointed out is that we simulate the circuits based on direct LU decomposition, while a structure level reduction technique is applied in [5] for the simulation on an actually smaller circuit size. The time efficiency of the algorithm is therefore more obvious.

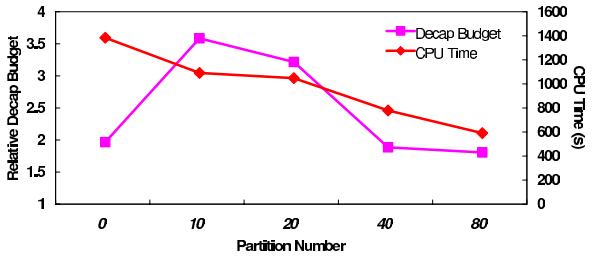

Figure 6: Comparison of Decap Budget and CPU Time between different partition sizes

We also notice that the difference among various partition sizes. For each example, the larger the partition number, the faster the speed as we projected. However, the decap budget experiences a variation. As an example of the 400K circuit illustrated in Fig 6, at small partition numbers, the budget is overestimated compared to flat run result. This is due to the reduced sensitivity of each node in a partitioned area, therefore a larger decap value added in each optimization iteration.

As partition number increases, the decap budget drops. A even smaller decap budget than flat run can also be achieved by using the new partitioning-based algorithm, which demonstrates the advantage of our partitioning optimization scheme. The reason could be explained as follows. In a flat run, the sensitivity computed reflects the contribution of each decap node to the sum of all violation areas, instead of to a specific violation node. Each decap node that has sensitivity calculated will be added decap values. This gives rise to conservative decap estimation because the real sensitivities of each decap node to each violation node are different. And the expected decap value at each decap node should be based on these more accurate individual sensitivities, not the sum of all the sen-

[2]The CPUs used in two papers are different (Intel Xeon versus Sun UltraSparc). It is difficult to scale the CPU times in terms of raw clock speeds.

Table 3: Comparison with the existing CG method for P/G decap optimization

Circuit	#Nodes	#Vio Nodes	CG1 (existing)			CG2 (proposed)			speedup ratio
			decap	iter	time(s)	decap	iter	time(s)	
ckt1	88	29	1.82	8	2.3	2.12	1	0.1	23
ckt2	336	63	3.15	10	15.2	4.16	2	0.8	19
ckt3	1,233	143	2.43	10	132	2.62	1	2.4	55
ckt4	12,673	1,083	3.09	8	1,995	3.67	1	54	37
ckt5	89,496	592	1.94	5	7,241	3.10	1	394	18

Table 4: Comparison between flat and partitioning-based decap optimization

Circuit	#Nodes	#Vio Nodes	CG1		CG2		Partitioned CG2			
			budget	time(s)	budget	time(s)	partition no.	budget	time(s)	speedup ratio
ckt6	242,600	49,626	3.91	9,592	5.04	1,746	5	6.09	744	13
	–	–	–	–	–	–	10	4.46	713	13
	–	–	–	–	–	–	20	4.05	438	22
ckt7	421,320	26,843	1.7	15,555	1.94	1,370	10	3.56	1,077	14
	–	–	–	–	–	–	20	3.19	1,034	15
	–	–	–	–	–	–	40	1.86	765	20
ckt8	827,025	87,903	N/A	N/A	N/A	N/A	20	3.58	2,619	N/A
	–	–	–	–	–	–	40	2.19	1,711	N/A
	–	–	–	–	–	–	80	2.15	1,705	N/A
ckt9	1,004,960	67,105	N/A	N/A	N/A	N/A	25	4.17	2,812	N/A
	–	–	–	–	–	–	50	2.25	2,675	N/A
	–	–	–	–	–	–	100	2.05	2,093	N/A

sitivities. On the other hand, with an estimation of the effective decap range for a partition, we can prevent from adding decap for a less-sensitive violation node in another partition. This effect somehow alleviates the problem suffered in a flat mode. However, the partition number should not be too large either. Otherwise inaccuracy and non-solution will occur due to a large number of violation nodes with limited decap nodes. In such cases, more partition iteration will be conducted as mentioned in Fig 5, leading to a longer run time.

6. CONCLUSIONS AND FUTURE WORKS

This paper gives an extremely fast decap optimization solution targeting at very large circuit sizes. The combination of our proposed improved conjugate gradient algorithm and partitioning-based optimization scheme can efficiently optimize power grid circuits with million nodes in a short time. Our theoretical analysis on the time complexity shows that new algorithm outperforms the existing decap allocation algorithms. Experimental results on a number of power grid circuits demonstrate that the proposed algorithm achieves at least 10X speed-up over the fastest decap allocation method reported so far with similar or even better decap budget quality, and a power grid circuit with about one million nodes can be optimized in about half an hour on the latest Linux workstation. In the future, parallel simulation will be explored to further improve the efficiency of the decap budgeting algorithm.

7. REFERENCES

[1] 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.

[2] 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.

[3] E. Chiprout, "Fast flip-chip power grid analysis via locality and grid shells," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, Nov. 2004, pp. 485–488.

[4] E. Chiprout and T. Nguyen, "Power analysis of large interconnect grids with multiple sources using model reduction," in *Proc. European Conference on Circuit Theory and Design*, Sept. 1999.

[5] 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.

[6] G. Karypis, R. Aggarwal, and V. K. S. Shekhar, "Multilevel hypergraph partitioning: application in VLSI domain," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 69–79, March 1999.

[7] 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.

[8] H. F. Qian, S. R. Nassif, and S. S. Sapatnekar, "Random walks in a supply network," in *Proc. Design Automation Conf. (DAC)*, 2003, pp. 93–98.

[9] 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.

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

[11] 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.

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