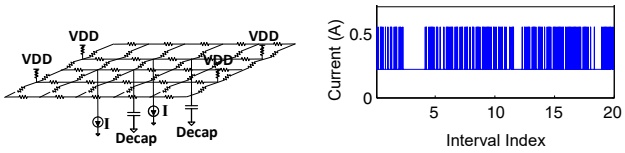
An Improved Leakage-Driven Runtime Decap Modulation Algorithm for Microprocessors

Leilei Wang and Pingqiang Zhou
School of Information Science and Technology, ShanghaiTech University, Shanghai, China
Corresponding Author's Email: zhoupq@shanghaitech.edu.cn

***Abstract*—The leakage power of decaps is a significant portion of the total chip power. Previous work [1] proposed a method to dynamically modulate the decap at runtime to save leakage power consumed by the unused decap. However, it did not consider the fact that the pre-placed decap at chip design stage actually has limited capacity, thus the estimated capacitance by the method [1] may exceed the decap's capacity. In this paper, we address this issue, and propose a method to improve the prior work to make it feasible. The effectiveness of our idea is tested on four benchmarks.**

I. Introduction

Power is delivered from the board-level voltage regulator to each on-chip transistor by a power delivery network (PDN) as shown in Fig. 1(a). The parasitics in the PDN, together with the temporal power variations in the switching circuit blocks, result in transient voltage noise in the power grid, which may impair the performance and reliability of a chip. The on-chip decoupling capacitors (decaps) in a processor chip can serve as temporary power reservoirs, and provide the needed power to its nearby large switching circuit blocks to suppress the supply noise, and thus increase the reliability of the power grid [2]. Unfortunately, the deliberately-added decap can occupy more than 20% of the total chip area in high-end processors and its leakage can contribute to more than 20% of the total power consumption [3].



(a) The model of power grid for a given chip, with VDD sources, decaps and circuit block loads.

(b) The simulated power profile of one circuit block in a processor chip. One time interval = 10,000 cycles.

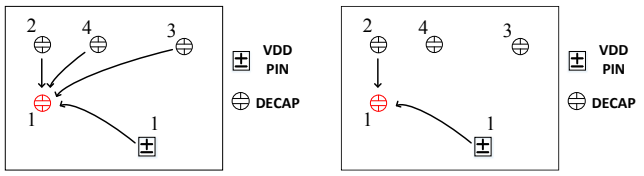
Fig. 1. The model of a power grid and one sample of the current trace for the current loads in the power grid.

Previous work in [1] proposed a method to adapt the amount of "on" capacitance of each pre-placed decap dynamically to the varying need of the power loads (see Fig. 1(b)) at runtime, thus to reduce the decap leakage consumed by the unused "on" capacitance. Although it has been shown that the idea in [1] can effectively reduce the decap leakage power, it ignores one important fact that the amount of the pre-allocated decaps distributed across the chip is limited, and when estimated amount of needed "on" capacitance by the method in [1] is larger than the pre-allocated capacitance, the method in [1] is unfeasible.

In this paper, we improve the method proposed in [1], and address the aforementioned issue. In Section II, we present our proposed method. Then we show the results in Section III, which is followed by the conclusion in Section IV.

II. Improved Decap Modulation Approach

Generally speaking, if one decap could not afford the demand charge due to limited capacity, its voltage drop will be larger than its surrounding power grid nodes. As a result, its insufficient amount of charge will have to be amortized among this decap's nearby VDD pins and other decaps with enough capacity. Fig. 2(a) shows one example. Decap1 is insufficient,



(a) Insufficient decap1 and its surrounding potential helpers.

(b) Decap1 is helped only by VDD1 and decap2, because decap3 is remote and decap4 itself is insufficient.

Fig. 2. A simple example showing an insufficient decap and its potential helpers.

so after turning on all its available capacitance, it has to borrow the remaining charge from nearby VDD pin VDD1 and three decaps decap2, decap3 and decap4.

Assume the amount of charge form each helper to this insufficient decap is proportional to the equivalent conductance between the helper and the insufficient decap [1], we can calculate the amortized charge for each decap helper, which is transformed to the extra "on" capacitance of the decap helper. The equivalent conductance matrix **A** for the example shown in Fig. 2(a) is given in Fig. 3, where the absolute value of each non-diagonal element represents the equivalent conductance between a decap/vdd and another decap/vdd.

	Vdd1	Decap1	Decap2	Decap3	Decap4
Vdd1	0.4905	−0.1626	−0.0514	−0.1203	−0.1562
Decap1	−0.1626	0.7035	−0.3051	−0.0203	−0.2155
Decap2	−0.0514	−0.3051	0.7824	−0.1277	−0.2982
Decap3	−0.1203	−0.0203	−0.1277	0.4121	−0.1438
Decap4	−0.1562	−0.2155	−0.2982	−0.1438	0.8137

Fig. 3. Matrix **A** of the example in Fig. 2(a).

A. How to Find the Equivalent Conductance Matrix A?

Given an RC power grid with N nodes, for the calculation of amortized charge, we can approximate it as a purely resistive grid whose system equation can be expressed as [4]

$$GV = J, \quad G \in R^{N\times N}, V \in R^N, J \in R^N. \tag{1}$$

where G is the conductance matrix, V is the voltage vector and J is the vector of current sources. Among the N nodes, we treat the nodes that the M VDD sources and K decaps connected at as port nodes, while other nodes as internal nodes. Then we can apply the macromodeling approach [5] to have

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{int} \\ \mathbf{V}_{port} \end{bmatrix} = \begin{bmatrix} J_{int} \\ J_{port} + I_{port} \end{bmatrix} \tag{2}$$

where

- G_{11} is the conductance matrix of the internal nodes,
- G_{12} is the conductance matrix between the internal nodes and the port nodes,
- G_{22} is the conductance matrix of the port nodes,
- V_{int} and V_{port} are the voltage vectors of the internal and port nodes respectively,
- J_{int} and J_{port} are the current sources connected at the internal nodes and ports respectively.
- I_{port} is the vector of currents flowing into the ports.

The Eq. (2) can be finally transformed into

$$I_{port} = \underbrace{(G_{22} - G_{12}^T G_{11}^{-1} G_{12})}_{\text{Port admittance matrix } \mathbf{A}} V_{port} + (G_{12}^T G_{11}^{-1} J_{int} - J_{port}) \tag{3}$$

where $A = G_{22} - G_{12}^T G_{11}^{-1} G_{12}$ is the equivalent conductance matrix among the port nodes, which is exactly the matrix we want as shown in Fig. 3.

B. Amortization Charge among the Helpers

After obtaining the matrix A, we can start calculating the amortized charge for each decap helper, i.e., its extra required "on" capacitance.

Although all the surrounding decaps or VDDs of an insufficient decap could be the potential helpers, the following two types of decaps/VDDs can be removed from the helper list:

- *The remote helpers*. In reality, a remote helper can only provide very tiny charge to the insufficient decap, because they have a large equivalent impedance to the decap, so we can ignore such remote helpers in the calculation. For instance, in Fig. 2(b) since the equivalent conductance between decap1 and decap3 (0.0203 in Fig. 3) is less than 5% of the self-conductance of decap1 (i.e., the diagonal element , 0.7035), we can assume that the contribution of charge from decap3 is zero, and correspondingly, we can set its equivalent conductance to be zero, and thus achieve a new matrix $\hat{\mathbf{A}}$ shown in Fig. 4.
  We can use a list S_k to record all the effective helpers for one insufficient decap k. For the example shown in Fig. 2(b), S_k for decap1 is {`Vdd1, decap2 and decap4`}.
- *The surronuding insufficient decaps*. All the insufficient decaps have to turn "on" all its available capacitance and are unable to help the other insufficient decaps. We use a list C_inMax to record such decaps. For instance, in Fig. 2(b), decap4 is insufficient, it would be put into C_inMax. Notice that the list C_inMax is dynamically updated as we process each insufficient decap in the iteration (see Section II-C).

	Vdd1	Decap1	Decap2	Decap3	Decap4
Vdd1	0.4905	−0.1626	−0.0514	−0.1203	−0.1562
Decap1	−0.1626	0.7035	−0.3051	0	−0.2155
Decap2	−0.0514	−0.3051	0.7824	−0.1277	−0.2982
Decap3	−0.1203	0	−0.1277	0.4121	−0.1438
Decap4	−0.1562	−0.2155	−0.2982	−0.1438	0.8137

Fig. 4. Matrix $\hat{\mathbf{A}}$, obtained by setting the remote helper's conductance to be zero in matrix A, as shown in Fig. 3.

As a result, the effective helpers for decap k would be those in S_k, but not in C_inMax. For an effective helper $j \in S_k, j \notin C\_inMax$, the extra amount of "on" capacitance it must provide to help decap k, $\triangle C_j^{'}$, is

$$\triangle C_j^{'} = \frac{a_{jk} \cdot (C_k - C_{k,max})}{\sum_{j\in S_k, j\notin C\_inMax} a_{jk}} \tag{4}$$

where

- a_{jk} is the equivalent conductance between the helper j and decap k.
- C_k is the required "on" capacitance of decap k estimated by the method in [1].
- $C_{k,max}$ is the maximum available capacitance of pre-placed decap k.

For example, in Fig. 2(b) the extra capacitance provided by decap2 to help decap1 is $\triangle C_2^{'} = \frac{0.3051\cdot(C_1 - C_{1,max})}{0.3051+0.1626}$.

C. Overall Solution and Discussion

The complete improved modulation approach is shown in Algorithm 1. We maintain a list C_toHelp to record all the insufficient decaps. After the pre-calculation of required "on" capacitance for each decap by the method in [1], we put the indexes of insufficient decaps into list C_toHelp (line $4 \sim 9$). Then for each insufficient decap k in C_toHelp, we estimate the amount of charge from its helpers with (4), until C_toHelp is empty (line $10 \sim 18$). Once the required amount of capacitance is determined for each decap, we can use the idea of Gated-decap [6] to set the right "on" portion of decaps across the chip.

The algorithm complexity is $O(K^2)$ theoretically, where K is the total number of decaps in the chip. Although new insufficient decaps may appear during the borrowing process in the iteration (line 16), the algorithm will terminate within limited iterations. This is because we know that though several decaps may not have enough available capacitance, in a local chip area, there should be enough decap resource to use, which is ensured at the design step of the processor chip.

Regarding the overhead of physical implementation of our proposed approach, the Gated-decap used in the proposed method (also in [1]) would bring about 5.36% area overhead of the decaps, while the energy overhead is negligible if an appropriate interval of modulation is adopted [6].

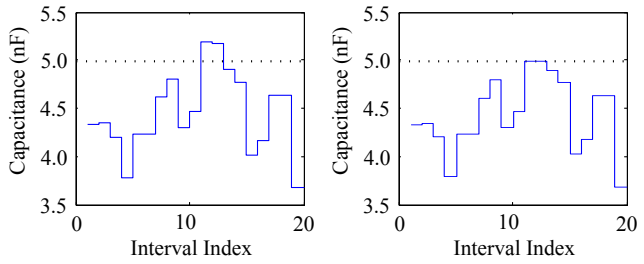
Algorithm 1 The Improved Online Decap Modulation Approach

Obtain matrix $\hat{\mathbf{A}}$. (see (3) and Fig. 4) (off-line step)
while a chip block switches **do**
  Create new lists $C_{\_inMax}$ and $C_{\_toHelp}$
  for $j = 1$ to K **do**
    Calculate C_j by the method in [1]
    if $C_j > C_{j,max}$ **then**
      add decap k to $C_{\_inMax}$, add k to $C_{\_toHelp}$
    end if
  end for
  while $C_{\_toHelp}$ is not empty **do**
    Pop a decap k from $C_{\_toHelp}$
    Find S_k of decap k (see Section II-B)
    for each $j \in S_k, j \notin C_{\_inMax}$ **do**
      re-calculate $C_j = C_j + \triangle C_j^{'}$ with (4)
      if $C_j > C_{j,max}$ **then**
        Add decap j to $C_{\_inMax}$, add j to $C_{\_toHelp}$
      end if
    end for
  end while
  Set the "on" capacitance of each decap
end while

III. Experimental Results

We test our method in four benchmarks. The power traces are obtained by running the PARSEC 2.1 benchmarks [7] on simulators MacPAT [8] and GEM5 [9], and the power grids are adapted from the IBM Power Grids [10].

Fig. 5 shows that modulation result of one insufficient decap in Benchmark1. The demand capacitance estimated by method [1] is shown in Fig. 5(a). But since this decap has a maximum available capacitance of 5nF, it has to be modulated as shown in Fig. 5(b). Using our method, the



(a) The demand capacitance of one decap in Benchmark1.

(b) With maximum capacitance 5nF, this decap could only be modulated under 5nF.

Fig. 5. The estimated and the modulated capacitance of one insufficient decap in Benchmark1.

insufficient amount of this decap, 186pF, is amortized among its surrounding helpers as shown in Fig. 6. Note that part of the charge is provided by the nearby Vdds, whose numbers were not shown in the figure.

Table I shows the maximum available capacitance (**MaxCap**) and the estimated required capacitance (**EstCap**) of the worst insufficient decaps in all the four benchmarks.

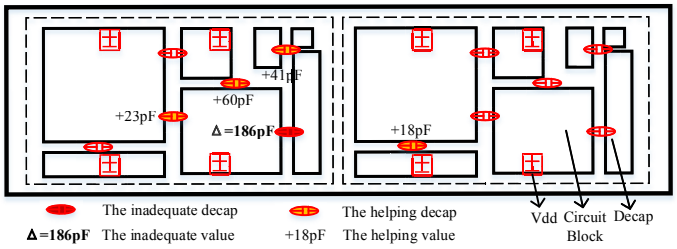


Fig. 6. The amortized charge (more specifically, capacitance) among the helpers in Benchmark1 for one insufficient decap in a two-core processor chip.

The Lowest voltage shows the minimum voltage of all the power grid nodes after using the proposed decap modulation method. From Table I we can see that our method is effective in modulating the insufficient decaps for all the benchmarks.

TABLE I
RESULTS OF ALL THE FOUR BENCHMARKS. THE VDD IS 1V AND WE SET 10% OF VDD AS THE ALLOWED VOLTAGE DROP IN THE EXPERIMENTS.

#Benchmark	Insufficient decap amount		Lowest voltage (V)
	MaxCap (nF)	EstCap (nF)	
Benchmark1	5.000	5.186	0.9039
Benchmark2	3.500	3.606	0.9086
Benchmark3	3.000	3.167	0.9185
Benchmark4	4.000	4.327	0.9123

IV. Conclusion

In this paper, we improve our prior work [1] by proposing a method to modulate the decaps for large function blocks in a processor chip, under the practical constraint that each pre-placed decap has limited capacity. Results on several benchmarks show that our method is feasible. In the future, we will test our method on more benchmarks of heterogeneous chips.

References

[1] L. Wang and P. Zhou, "Leakage power reduction in multicore chips via online decap modulation," in *CSTIC*, 2016, pp. 1–3.
[2] P. Zhou *et al.*, "Congestion-aware power grid optimization for 3D circuits using MIM and CMOS decoupling capacitors," in *ASP-DAC*, 2009, pp. 179–184.
[3] J. Gu *et al.*, "Design and implementation of active decoupling capacitor circuits for power supply regulation in digital ics," *TVLSI*, vol. 17, no. 2, pp. 292–301, Feb. 2009.
[4] C.-W. Ho *et al.*, "The modified nodal approach to network analysis," *TCAS*, vol. 22, no. 6, pp. 504–509, Jun. 1975.
[5] M. Zhao *et al.*, "Hierarchical analysis of power distribution networks," in *DAC*, 2000, pp. 150–155.
[6] Y. Chen *et al.*, "Gated decap: Gate leakage control of on-chip decoupling capacitors in scaled technologies," *TVLSI*, vol. 17, no. 12, pp. 1749–1752, Dec. 2009.
[7] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton, NJ, USA, 2011.
[8] S. Li *et al.*, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Micro*, 2009, pp. 469–480.
[9] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.
[10] "IBM power grid benchmarks," available at http://dropzone.tamu.edu/~pli/PGBench/.