國 立 清 華 大 學

碩 士 論 文

題目：考慮單元轉換時間之去耦合電容配置方法以用於減少供電干擾

Transition-Aware Decoupling-Capacitor Allocation in Power Noise Reduction

系 所 別： 資訊工程學系碩士班

學號姓名： 9562529 劉哲宇 (Che-Yu Liu)

指導教授： 黃婷婷 博士 (TingTing Hwang)

中華民國九十七年七月

中文摘要

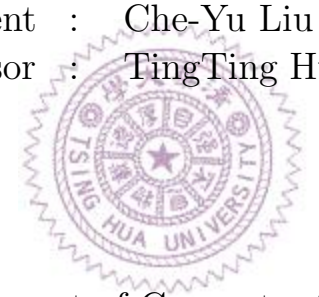
隨著製程的不斷進步，現在晶圓的密度變得越來越高，加上運行速度也不斷地在上升，使得有越來越多項目會在同時發生轉換，因此使得晶圓上電力供應被干擾的情況變得越來越嚴重。電力供應的不穩，會引發許多嚴重的問題，其中最重要的就是會使整個晶圓的效能降低，更甚者，連功能都會發生錯誤。所以穩定的電力供應在現在的設計上是必須的。

一般來說，最常被用來解決供電干擾的方法是在晶圓上配置一些去耦合電容；去耦合電容的作用如同一個電子銀行，在平時它會儲存著電荷，當單元做轉換有需要時，它便會將其電荷釋放出來供單元做轉換使用，因此能夠有效降低供電干擾。我們發現影響供電干擾的因素，除了單元的位置之外，單元的轉換時間也是相當重要的。

在這篇論文裡，我們提出了兩個解決供電干擾的方法：首先，我們會在佈局之前，先去考慮單元的轉換時間以及單元的位置，預測哪些單元可能會發生很嚴重的供電干擾，在這些供電干擾嚴重的單元旁邊，我們先綁住一些去耦合電容，藉由這些被綁在單元旁邊的去耦合電容，我們可以在佈局之前先解決部分的供電干擾的問題；接下來，在佈局之後，我們提出另一個搬動單元的方法進一步解決供電干擾。在這個方法裡，我們會藉由改變原本單元擺放的位置，去減少原本供電干擾很嚴重的地方的干擾來源，使得整個晶圓上的電力需求變得更平均，因此也就能確實解決供電干擾的問題。

Transition-Aware Decoupling-Capacitor Allocation in Power Noise Reduction

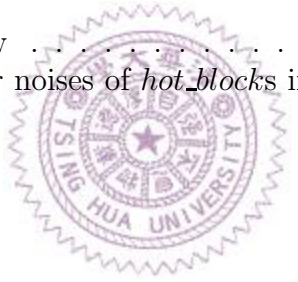
Student : Che-Yu Liu
Advisor : TingTing Hwang



Department of Computer Science
National Tsing Hua University
HsinChu, Taiwan 30043

Contents

List of Figures



List of Tables

Abstract

Dynamic power noises may not only degrade the circuit performance but also reduce the noise margin which may result in the functional errors in integrated circuit. Decoupling capacitor (*decap*) allocation is one of the most effective way in reducing serious dynamic power noises (*hotspot*s). To allocate *decap before placement*, we observed that not only locations but also rising time of functional cells are required to accurately predict power noises. Compared to a previous work which only takes neighborhood relation into consideration, our method is more efficient in reducing *hotspot*s. Furthermore, to reduce the *hotspot*s *after placement*, instead of only using the empty space as proposed in the previous work, we move out cells in the area with serious power noise area (hot area). The obtained empty space can be used to accommodate *decap*s to further reduce the *hotspot*s. The experimental result shows, compared to the previous work [1], our estimation function to allocate *decap* before placement is 23% better in reducing power noises. Moreover, compared to a method which fills *decap*s to all remaining empty space, our cell move algorithm can further reduce 12% maximum

power noises and eliminate most of remaining *hotspot*s. In summary, compared to the original circuits (without *decap*), about 60% of *hotspot*s can be removed using our prediction function before placement, and most of the remaining *hotspot*s are removed by our cell moving step after placement.

Chapter 1

Introduction

To produce high speed, low power and less cost integrated circuits (IC), the feature size of IC has been aggressively reduced and the operating frequency has been increased. The scaling of CMOS is expected to continue with time. Future nanometer circuits will contain more than a billion transistors and operate at clock speeds well over 10GHz. A robust and reliable power supply network is necessary for such a high speed and density IC.

However, as fabrication technology scale progresses and nominal supply voltage decreases, modern designs become more sensitive to dynamic power noises. The dynamic power noises are becoming worse because a large number of events in the circuit switches simultaneously within a short period of time. The power noises decrease the drive capability of transistors due to reduced effective supplied voltage to the events. This degrades the circuit performance [1, 2, 3]. The dynamic power noise may also cause functional errors [4, 5] since the noise margin is lower with the decreased supplied voltage

decreasing. Therefore, reducing dynamic power noises is important.

Recently, many researches have been developed directly or indirectly toward the power-supply network optimization including wire sizing [13], topology optimization [14], onchip voltage regulation [15], thermal placement [16] and decoupling capacitance (*decap*) allocation. Among others, *Decap* allocation is most widely used technique to reduce dynamic power noises [1, 2, 4].

*Decap*s can reserve electronic charge and release it while cells make switches. Therefore, voltage drops on power supply networks can be reduced. However, the allocation effort is effective only when the *decap*s are placed near the *hotspot* cells (cells that have the most serious dynamic power noises). Previous work on *decap* allocation can be categorized into two types. The first one is performed $after\ placement$ [2, 4]. Since IR drop on the supply is analyzed after cells are placed, more accurate calculation of power noises can be obtained. However, the drawback of this approach is that large *decap* is hard to be reallocated after placement. The most suitable location for *decap* cells may not be empty, and thus cells (including *decap* and functional cells) may be moved far away from their optimal positions. The second one is to determine which cells decoupling cells are bound to $before\ placement$ [1]. Then, as the second step, cells are only fine tuned to compensate the inaccuracy of prediction after placement. The challenge of this $before\ placement$ method is to predict *hotspot* cells accurately so that the number of cells moved is as

few as possible. In this paper, the second approach will be taken.

A previous work [1] taking the *before placement* approach was proposed. It has shown 19% more efficient on reducing dynamic power noises as compared to the method which distributes *decap*s evenly to cells. To determine the quantity of bound *decap* for each cell, the authors predict the neighborhood current consumption (NCC). To compute the NCC of each cell, they first compute the mutual contraction value for each connection before placement. Then, the connections whose mutual contraction value is among top 30% are classified as strong connections. At last, the NCC value for each cell is computed by taking neighboring cells with strong connections and neighboring cells' switching current consumption into consideration. The cell with large NCC value will be bound with large *decap*. In essence, presumed locations of cells are used to allocate *decap*. We observe NCC value is not adequate to model the phenomenon of dynamic power noises.

The worst case of dynamic power noises occur at the beginning of a clock cycle. For instance, we conduct an experiment of the number-of-*hotspot* distribution on benchmark circuit s9234 (in ISCAS89 benchmark set) by the vectorless approach suggested in [7, 8] which has been proven to be efficient in evaluating leakage current and dynamic power noises in the worse case. In the vectorless approach, a clock cycle is divided into several time intervals. For each time interval, the modified nodal analysis (MNA) [9] is applied

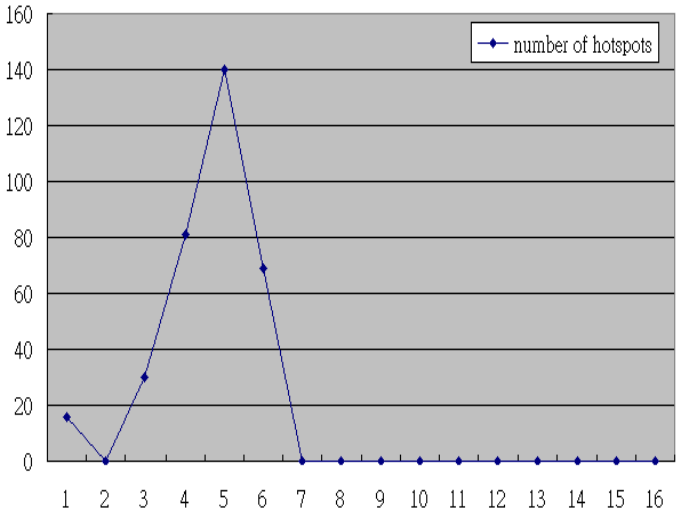


Figure 1.1: The number-of-*hotspot*s distribution in one clock cycle

to evaluate the dynamic power noises. Figure 1.1 shows the result. The horizontal axis shows the time intervals in a clock cycle and the vertical axis shows the distribution of the number-of-*hotspot*s in a clock cycle. Here, a *hot_spot* is defined as a placed cell whose power noise is larger than 5% of Vdd. From this figure, we observe that a large number of *hotsopt*s occur at the beginning of a clock cycle. Therefore, timing information should be taken into consideration in predicting power noises.

In this thesis, we will propose a more accurate prediction function to bind *decap* to functional cells before placement. Then, as the second step, we propose a cell moving method to move cells from hot area which suffers serious dynamic power noises. Next, the empty space obtained by moving cells can accommodate *decap*s to reduce the dynamic power noises.

The organization of this thesis is as follows. In Chapter 2, previous work on *decap* allocation is presented. In Chapter 3, the modeling and analysis of power supply network are presented. In Chapter 4 and Chapter 5, design flow and algorithms are discussed. Experimental result is shown in Chapter 6. Finally, conclusions are made in Chapter 7.

Chapter 2

Previous Work

In this chapter, previous work is reviewed. First, we introduce the work of decap allocation *at floorplan level* in Section 2.1. Then, Section 2.2 presents the algorithm for decap allocation *after placement*. Finally, the method of decap allocation *before placement* is described in Section 2.3.

2.1 Decap Allocation at Floorplan Level

Zhao, Roy and Kon [12] focuse on the problem of onchip decap deployment at floorplan level. The objective is to find an area efficient scheme to deploy the decap such that the power-supply noises at each module are below a specified limit. With floorplan information and switching-activity profile of each circuit module, they propose two methods to solve this problem. The fist one is considered as a post floorplan step; whereas the second one is regarded as noise-aware floorplanning.

In the first method, the worst case noise of each circuit module is es-

timated after floorplan step is performed. Based on the worst case power-supply noise, an iterative method to figure out decap budget for each circuit module is applied. After that, white space (WS) for decoupling capacitors is allocated in the following two steps. In the first step, existing WS in the chip is assigned to its neighboring modules with a linear programming (LP). If the existing WS is not enough to meet the total decap demand, additional WS is inserted into the floorplan by extending the floorplan dimensions in both $|x|$ direction and $|y|$ direction.

In the second method, a noise-aware floorplanning methodology is proposed. In addition to wire length and area, power noises are taken into consideration. To achieve the goal, the cost function is modified so that power-noise factor is considered. Then, a simulated annealing algorithm to implement the noise-aware floorplanning methodology is used.

This work has proposed two efficient methods to solve power-noise problem at floorplanning level.

2.2 Decap Allocation After Placement

Su, Sapatnekar and Nassif [4] investigate the decap allocation problem in standard-cell design. The target is to incorporate decaps with each functional block. Their decap allocation methodology is proposed for individual functional block after placement. The white spaces within each row is utilized to place decaps. After applying the method, the power noises are minimized.

Although the exact position of each cell may be changed, the order and the relative positions of each cell are fixed. Hence, impacts on the original structure are slight.

This approach is implemented by a nonlinear programming based scheme. First, adjoint sensitivity analysis is utilized to compute the sensitivity of the objective function with respect to the widths of all decaps in the network. Then, the problem is formulated as a linearly constrained nonlinear optimization problem. The constraints restrict the total decap widths within the total amount of empty spaces in a row. After that, the constraint functions are added into the objective function by applying the Lagrangian relaxation technique. Finally, a standard QP solver is chosen to solve this problem.

Although this approach can solve power-noise problem after placement, its effectiveness is limited because placement structure is fixed and large decap is hard to be reallocated. The most suitable location for decap cells may not be empty and the remaining white spaces may not be enough for the requirements of decap cells.

2.3 Decap Allocation Before Placement

Yeh and Marek-Sadowska [1] apply the decap allocation method before placement. First, the weights of each connection are computed by mutual contraction. Mutual contraction introduced in [10] is a scheme to predict wire lengths before placement. Cell-pairs with larger mutual contraction tend to

be placed closer. The connections among top 30% are classified as strong connections. Those strong connections are expected to have short predicted lengths. The first level neighbors of a cell n is defined as the nodes linked to it by strong connections, and n itself.

After defining the first level neighbors of each cell, they compute the cell *current consumption* (CC) value. CC is a function of the switching frequency and switching capacitance of a cell. The cells with higher CC value impact the power-noise more aggressively. Then, the neighbor cells are taken into consideration. Cell neighbor-CCs (NCC) is a metric to calculate the influences of the neighbor cells on the cell. The value of NCC depends on the amount of neighbor cell involved in consideration. The amount of a cell's neighborhood is controlled by the neighborhood level. As the level increases, the amount of neighbor cell increases, and the NCC of a cell is influenced by more and more other cells. The noise weight of a cell is computed by its NCC. After computing the noise weight of a cell, the timing weight is computed where critical path information is considered. Padding a large decap to cells on a critical path may increase distance between them and consequently increase the interconnect delay. The slack of a cell is used to avoid padding too large decap to critical cells. Finally, the decap size which pad to the cell is quantified by noise weight and timing weight of a cell.

After assigning decap padding to cells, placement is performed. After

placement, a gate-sizing algorithm reducing the power noises further is proposed. The algorithm is implement by a Sequence of Linear Programs (SLP) technique. The objective of each linear program is to minimize total power consumption and to reduce power noises. The constraints considered include timing, area, and power noise.

As we mention above, besides the position of a cell, the switching time should also be taken into consideration when predicting the decap padding. If cells are placed close, [1] consider that they will be influenced by each other. However, if they switch in different time intervals within a clock cycle, the mutual impacts between them become slight. Figure 1.1 also shows that the worst case of dynamic power noises occur at the beginning of a clock cycle. Based on these observations, timing information should be considered in predicting power noises.

Chapter 3

Modeling and Analysis of Power Supply Network

In our experiment, we divide the chip core into several blocks and each block corresponds to a grid node [1]. The discharge current of each gate is modeled as a triangular waveform and the power grid network is as a RC network as shown in Figure 3.1 where net resisters and net capacitance are included. To calculate the dynamic power noises, in each block, we lump the decoupling capacitors as single capacitor and connect it to the grid node. In addition to the decoupling capacitors, we also model the discharge current of each gate as a current source and lump them as a single current source connecting to the grid node similar to [1].

The modified nodal analysis (MNA) [9] is adopted to calculate the dynamic power noises. After applying the Backward Euler technique, the relationship of components are modeled as follows:

$$[G + \frac{C}{h}]v[k] = I[k] + \frac{C}{h}v[k-1]$$

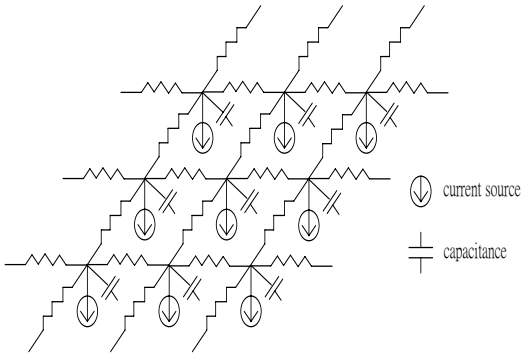


Figure 3.1: The power supply network

where G is the conductance matrix, C capacitance, v a vector of nodal voltages, k the kth time interval, I the vector of current sources, and h the time for transient analysis.

Chapter 4

Design Flow

Figure 4.1 presents our design flow. The input is a synthesized circuit. We propose two algorithms in the design flow. The first one, *Decap Padding*, is performed $before\ placement$, and the second, $Cell\ Moving$, is $after\ placement$. In the first step, based on our observation, we will develop a new *decap* padding function to determine the quantity of *decap* bound to each cell. Since *decap*s are bound with cells, they are placed near the corresponding cells after placement. By this method, dynamic power noises can be effectively reduced. Although the area of *decap*s is reserved before placement, *hotspot*s may still exist after placement. Therefore, we propose in the third step to further eliminate hot area by utilizing more accurate $after$-*placement* information. The main idea of the third step is to move cells from hot area. Then, the empty space obtained by moving cells can be further allocated to *decap*s, and hence the dynamic power noises in the hot area are reduced.

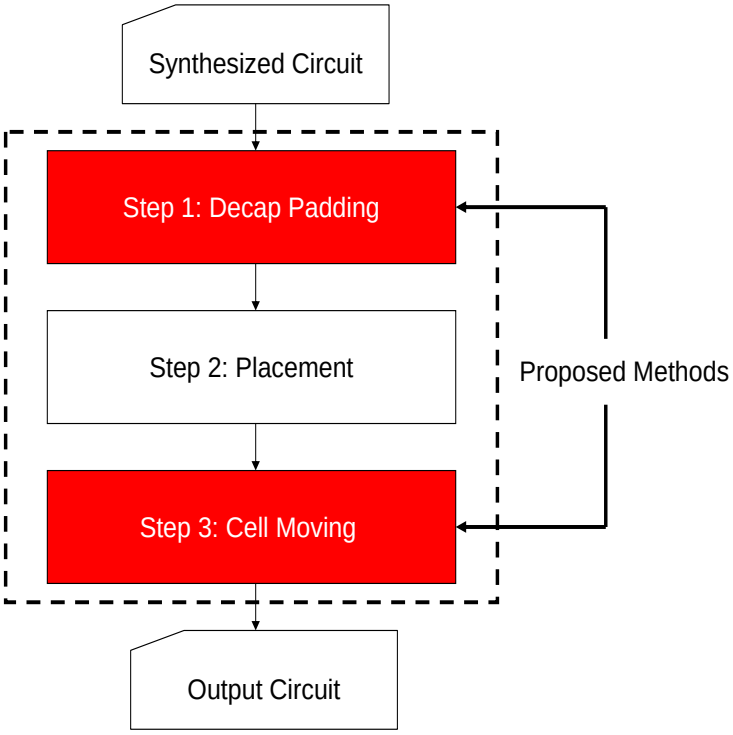


Figure 4.1: Design Flow

Chapter 5

Algorithms

In this chapter, *Decap Padding* and *Cell Moving* algorithms will be presented in following subsections.

5.1 Decap Padding

By the experimental result in Figure 1.1, we observe a large number of *hotspot*s occurs at the beginning of a clock cycle. To accurately pad *Decap*s to cells, we apply another experiment on circuit s9234 (in ISCAS89 benchmark set). Similar to the experiment in Figure 1.1, this experiment divides a clock cycle into several time intervals. Then, the number of rising cells in each time interval in the worst case is calculated by the vectorless approach in [7]. The experimental result is shown in Figure 5.1. The horizontal axis shows the time intervals in a clock cycle and the vertical axis shows the number of rising cells in each time interval. By Figures 1.1 and 5.1, we observe the *hotspot*s happen while a large number of cells rise simultaneously.

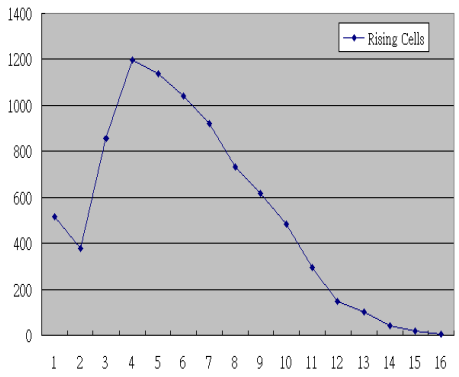


Figure 5.1: The number of rising cells in time intervals

In the previous work [1], neighboring cells with strong connections and neighboring cells' switching current consumption (computed as switching probability × output loading) are used to determine the amount of decoupling capacitor bound to a cell. Since cells are assumed to make switching at the same time in one clock cycle, only location of cells are taken into consideration to reduce dynamic power noises in [1]. However, if a clock cycle is carefully examined, we found that cells make transitions at different intervals. Therefore, except NCC values, rising interval of cells in a clock cycle should be taken into consideration to allocate *decap*s. Furthermore, by these two figures, we also observe that the relationship between the number of *hotspot*s and the number of rising cells is not linear. As the number of rising cells decreases, the number of the *hotspot*s decreases dramatically as shown in these two figures. Hence, the methodology to distribute *decap*s to cells should be in an exponential manner. For a more accurate padding, we

first define the weight for each time interval as follows:

$$interval\_weight_i = (rising\_cells\_number_i)^{exp} \tag{5.1}$$

where $interval\_weight_i$ is the weight of the ith time interval, $rising\_cells\_number_i$ is the number of rising cells in the ith time interval, and exp is a user-specified weight. Next, for each $cell_j$, the $tran\_weight_j$ of $cell_j$ is defined by:

$$tran\_weight_j = \sum_{i=1}^{|time\_interval|} W(i,j) \tag{5.2}$$

and

$$W(i,j) = \begin{cases} interval\_weight_i & if\ cell_j\ rises\ in\ time\_interval_i \\ 0 & otherwise \end{cases}$$

where $|time\_interval|$ is the number of total time intervals. For instance, Figure 5.2(a) shows an example circuit composed of seven cells. The primary inputs in the example circuit are P_0 to P_3, and the primary outputs are P_4 to P_6. For simplicity, we assume, in this example, the rising and falling time of each pin in a cell are the same, and the delay of each pin in a cell is also the same. The number inside the cell represents the cell delay, and the cell name is above the cell. Figure 5.2(b) shows the possible rising cells in each time interval after applying vectorless approach where the length of a time interval is set to be one time unit. Note that a cell may rise in several time intervals. For example, for path P_3-c_6, c_6 rises in the 1st time interval, and for path P_2-c_3-c_6, c_6 rises in the 2nd one. From this timing bar shown in Figure 5.2(b) and exp in Equation (5.1) being set to be 2, the $interval\_weight_i$ for $i = 1$

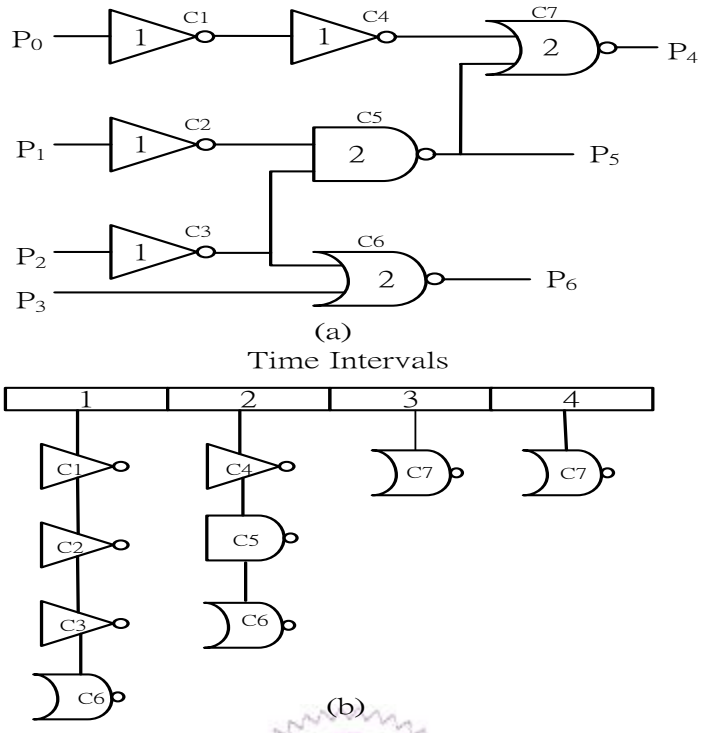


Figure 5.2: (a) the example circuit (b) the rising gates in time intervals

to 4 are $16(4^2)$, $9(3^2)$, $1(1^2)$ and $1(1^2)$, respectively. Then, the $tran\_weight_j$ for a $cell_j$ can be calculated. For instance, for cell c_6, because c_6 rises in time intervals 1 and 2, $tran\_weight_6 = interval\_weight_1 + interval\_weight_2 = 25$. Similarly, we can compute $tran\_weight$ for all cells and $tran_weight_j$s for $j = 1$ to 7 are 16, 16, 16, 9, 9, 25, and 2, respectively.

After defining the cost function taking into consideration the transition time of a cell, we now define a function to predict the amount of power noises of a cell. Since both *timing* and *location* are two important factors to induce power noises, we should not ignore the effect of location. Hence, the concept

of strong connections borrowed from [10] is utilized. The strong connections are links whose two ends (cells) are predicted to be close after placement. To collect the strong connections, the connecting weight of each link is defined by following steps. First, if a net k connects $d(k)$ nodes, each link (u, v) in the net is assigned a weight by:

$$link\_weight(u, v) = \frac{2}{d(k) \times (d(k) - 1)}$$

Then, the normalized link weight of link (u, v), $nlink\_weight(u, v)$, is defined as:

$$nlink\_weight(u, v) = \frac{link\_weight(u, v)}{\sum_x link\_weight(u, x)}$$

where $\sum_x(u, x)$ is the sum of all $link_weight$s of links incident to u. Finally, the mutual contraction MC for link (u, v) is computed by

$$MC(u, v) = nlink\_weight(u, v) \times nlink\_weight(v, u)$$

In [10], mutual contraction has been proven to be able to predict the wire length before placement. The larger the value of mutual contraction between two nodes is, the shorter the wire length is. Next, strong connections are links whose mutual contraction values are top 30% among all links as defined in [1]. The strong connections are predicted to have short lengths. Then, we define the neighboring transition weight, ntw_j, for $cell_j$ as:

$$ntw_j = \frac{\sum_{z \in n\_set(j)} tran\_weight_z}{|n\_set(j)|} \tag{5.3}$$

where $n\_set(j)$ is a collected set of neighbors linked by strong connections to $cell_j$ and $|n\_set(j)|$ is the size of this collected set. $tran\_weight_z$ is the transition weight of a $cell_z$ in $n\_set(j)$.

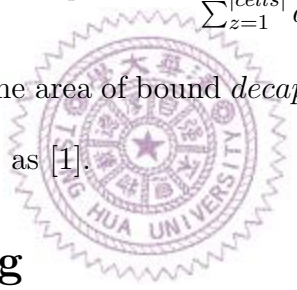
Then, our *decap* weight for any $cell_j$ is:

$$decap\_weight(j) = \alpha \times tran\_weight_j + \beta \times ntw_j \tag{5.4}$$

where α and β are specified by designer. $tran\_weight_j$ is calculated by Equation (5.2) and ntw_j is calculated by Equation (5.3). Finally, the quantity of bound *decap* for $cell_j$ is:

$$decap(j) = total\_decap\_area \times \frac{decap\_weight(j)}{\sum_{z=1}^{|cells|} decap\_weight(z)}$$

where *total_decap_area* is the area of bound *decap*s which is set to be 20% of total cell area in this paper as [1].



5.2 Cell Moving

In section 5.1, we propose a method to pad *decap*s to the cells before placement. Although the area of *decap*s is reserved before placement, *hotspot*s may still exist after placement. Therefore, in this section, we propose a method to further eliminate hot area after placement.

Figure 5.3 shows the algorithm. At first, since cells are already placed, more accurate timing model such as half-perimeter wirelength [11] on connecting wires can be used. With more accurate timing informatin, dynamic

power noises are analyzed. Then, *hot_block_list* is constructed. Each element in the list is modeled as hot_block_i, where i represents the ith block in the circuit. hot_block_i gives the value of maximum dynamic power noises of the ith block among all time intervals. The *hot_block_list* is sorted in decreasing order. If the dynamic power noises of the current *hot_block* is above a user-specified threshold, $Cell\_Moving$ step will start adding *decap*s to the block to reduce the power noises. Next, our algorithm will process each *hot_block* sequentially. If the ratio of *current_sum* to *decap_area* is still bigger than the threshold where *current_sum* is the lumped value of total current sources in the block at the time interval in which maximum power noises occur and *decap_area* is the area of total *decap*s in the block, we begin to move out cells in the block to other appropriate block until the ratio is smaller than the threshold. The reason behind this threshold is the assumption that the ratio of maximum current to *decap* is an effective index to power noises. The cell moving step is described as follows.

First, a *hot_cell_list* is constructed for current hot_block_i. The cells in *hot_cell_list* are those cell located in $block_i$ and rising in time intervals with serious power noises. The cells in *hot_cell_list* are sorted by slack for current hot_block_i, in decreasing order. Then, the algorithm will select the first cell in *hot_cell_list* for process. The reason of this selection is that a cell with large slack is more flexible to be moved. After selecting, the selected cell is

Procedure *Cell_Moving*
begin
  Perform timing and dynamic power noises analysis;
  Construct *hot_block_list* in decreasing order;
  For(each block in *hot_block_list*)
    If(the power noises in the block $> threshold_{noise}$);
      Add *decap*s to the block
      If($\frac{current\_sum}{decap\_area} > threshold_{pcd}$ in the block)
        Construct *hot_cell_list* in deceasing order by slack;
        While($\frac{current\_sum}{decap\_area} > threshold_{pcd}$ in the block)
          If(*hot_cell_list* is not empty)
            Select the first hot cell;
            Remove it from *hot_cell_list*;
          else
            break;
          Find possible destination blocks;
          Prune off some destination blocks;
          Establish *candidate_list* by *hot_value*s;
          While($candidate\_list \neq \phi$)
            Select next destination block in *candidate_list*;
            If(timing checking passes)
              Move hot cell to this block;
              Update timing information;
              Break;
  endFor
  Fill Decaps to all remaining empty space;
end

Figure 5.3: The *Cell_Moving* Algorithm

removed from *hot_cell_list*. The blocks that surrounds the current *hot_block* will be the candidate-destination blocks. The destination block must satisfy two constraints. The first one is the empty area in the this block must larger than the area required by the moved cell. The second one is that the block cannot be a *hot_block* at any time interval that the selected cell rises. The block which violates either one of the two constraints will be pruned. The rest of the blocks are collected in *candidate_list* and sorted by *hot_value*s.

The *hot_value* of each block is computed by two numbers. The first one is the number of time intervals in which the block is a *hot_block* and the second one is the number of total time intervals. A small *hot_value* means that the block is not hot in most of time intervals. For example, if a block become *hot_block* in 3 time intervals and the total number of intervals is 10. The *hot_value* of this block is $\frac{3}{10}$. The *hot_value*s in *candidate_list* is sorted in increasing order. Then, *Cell_Moving* step will consider to move the hot cell to the block with the least *hot_value*. If the timing of the circuit is kept after moving cell, the cell is moved to the first block and the timing information is updated. Otherwise, the next block in the *candidate_list* is selected. After performing the above steps, all remaining empty space is filled with *Decaps*.

Take Table 5.1 as an example where the normalized dynamic power values to Vdd in partial blocks of the circuit in all time intervals are shown. The column index is the name of the block, and the row index shows the time

interval. We assume power noises threshold is 5% of Vdd value and $block_5$ at $time\_interval_1$, hot_block_5, has the most serious dynamic power noises among all blocks. $Cell\_Moving$ will try to move cells from $block_5$ to nearby blocks. Figure 5.4 shows partial blocks of the circuit and their corresponding maximum normalized dynamic power noises. As shown in Figure 5.4, the candidate-destination blocks are $block_1$ to $block_4$, and $block_6$ to $block_9$. We assume empty area to accommodate moved cell in each candidate block is sufficient. Then, $Cell\_Moving$ algorithm will check the second constraint. Assume that the selected cell rises at time intervals 1 and 2. As shown in Table 5.1, $block_2$, $block_7$, $block_8$, and $block_9$ are *hot_block* at intervals {1,2}, {1}, {2}, {1}, respectively, in which, the selected cell rises. Therefore, these blocks will be pruned by pruning step. Then, the *candidate_list* will be constructed by the rest of blocks, $block_1$, $block_3$, $block_4$, and $block_6$. Next, the key values of each candidate block are $0(block_1)$, $\frac{1}{4}(block_3)$, $0(block_4)$, and $0(block_6)$. Therefore, *candidate_list* will be sorted as $block_1$, $block_4$, $block_6$, and $block_3$. Finally, $Cell\_Moving$ algorithm will try to move the hot cell to $block_1$ first. If the timing check for moving the cell to $block_1$ is passed, moving is applied and timing information is updated. Otherwise, $Cell\_Moving$ algorithm will move the hot cell to the next block, $block_4$, in *candidate_list*. $Cell\_Moving$ algorithm will move cells from $block_5$ until $\frac{current\_sum}{decap\_area}$ is less than a user-specified value.

Table 5.1: The dynamic power noise (%) of each block in time intervals

	Time Intervals			
	1	2	3	4
$block_1$	3%	2%	1%	0%
$block_2$	**6%**	**7%**	4%	2%
$block_3$	1%	3%	4%	**6%**
$block_4$	4%	2%	1%	2%
$block_5$	**9%**	**7%**	4%	2%
$block_6$	2%	1%	1%	1%
$block_7$	**7%**	4%	**6%**	4%
$block_8$	3%	**6%**	**7%**	4%
$block_9$	**7%**	4%	2%	1%

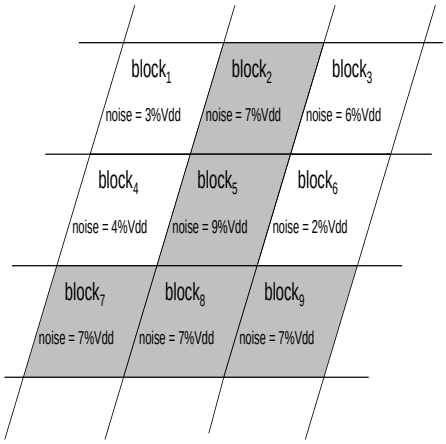


Figure 5.4: The distribution of maximum dynamic power noises

Chapter 6

Experimental Result

Figure 6.1 shows our experimental flow. The experiments are conducted using ISCAS89 benchmark circuits. The switching current of each cell is obtained by *Hspice* and modeled as a triangular waveform. First, all benchmark circuits are synthesized with *TSMC* 0.13μm cell library by *Design Compiler*. After synthesis, the *Decap_Padding* algorithm is applied to bind *decap*s to functional cells before placement. Then, benchmark netlist is placed by *SOC Encounter* and power noises analysis is performed after placement. Finally, the *Cell_Moving* algorithm with timing and power noise information is applied to further reduce *hotspot*s.

Table 6.1 shows the characteristics of the benchmark circuits. The names of the benchmark circuits are listed in the first column. The second and third columns report the number of cells and cell area in μm^2 after synthesis in each benchmark circuit.

In the first experiment, *Decap_Padding* algorithm is compared to the

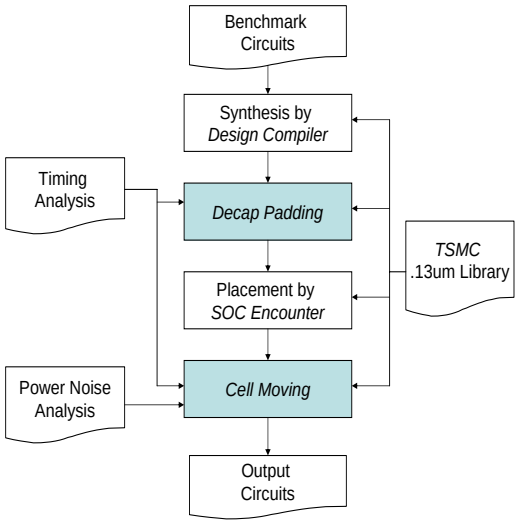


Figure 6.1: Experimental Flow

Table 6.1: Benchmark circuits

Circuits	# of cells	cell area (μm^2)
s9234	2022	26033
s13207	3378	49952
s35932	12032	164254
s38417	11633	162838
s38584	13956	182786

method, WGT, which also binding *decap*s to cells $before\ placement$ taking neighborhood current consumption (NCC) into consideration in [1]. As in [1], the area of bound *Decap*s is set to be 20% of total cell area. A *Decap* area is equal to the area of a $INVX1$ cell in TSMC .0.13μm library. Any reserved area whose area is less than one *Decap*, will not be allocated as *Decap*s. In this experiment, *exp* in Equation (5.1) is set to 1.8, and α and β in Equation (5.4) are 0.7 and 0.3, respectively. The experimental results of

original circuits without *decap*s allocation, $NoDecap$, are also reported as baseline for comparison. Table 6.2 shows the experimental result. The third, forth, and fifth columns report the value of maximum dynamic power noise, the number of hot grid nodes, and the number of hot cells. Here, hot grid nodes and hot cells are defined as those grid nodes and cells suffering the power noises larger than 5% of Vdd. The last row reports the normalized average value to the one of $NoDecap$. By this experiment, compared to WGT, our method is 7% more efficient in reducing maximum power noises, 24% and 23% better in reducing the number of hot grid nodes and the number of hot cells, respectively. Furthermore, compared to $NoDecap$ in average, the resultant maximum power noises of our method is only 57%, and the ratios of hot grid nodes and of hot cells are only 42% and 40%, respectively. That is, about 60% of *hotspot*s of original circuit is removed before placement.

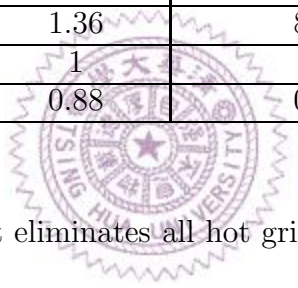
In the second experiment, we compare our result after performing $Decap\_Padding$ and $Cell\_Moving$ to the ones of $Baseline$ and $AllDecap$. $Baseline$ is the method only performing $Decap\_Padding$ algorithm, and $AllDecap$ is that fills *Decap*s to all remaining empty space after performing $Decap\_Padding$ algorithm. $Threshold_{pcd}$ in $Cell\_Moving$ algorithm is set to the average of $\frac{current\_sum}{decap\_area}$ for the non-hot blocks whose maximum power noises are within 1% to 3%. Table 6.3 shows the experimental result. By this experiment, compared to $AllDecap$, $Cell\_Moving$ is 12% more efficient in reducing maximum

Table 6.2: The Comparison of *before-placement* methods

Circuits	Methods	Max. Noise (V)	# of hot grid nodes	# of hot cells
s9234	*NoDecap*	0.172	159	1201
	WGT	0.112	152	1202
	Ours	0.131	128	1046
s13207	*NoDecap*	0.088	98	1044
	WGT	0.054	9	96
	Ours	0.054	16	161
s35932	*NoDecap*	0.114	305	2744
	WGT	0.056	63	538
	Ours	0.064	57	535
s38417	*NoDecap*	0.132	527	5426
	WGT	0.08	313	2991
	Ours	0.072	210	1888
s38584	*NoDecap*	0.178	866	7290
	WGT	0.135	757	6259
	Ours	0.072	410	3540
AVG	*NoDecap*	1	1	1
	WGT	0.64	0.66	0.63
	Ours	0.57	0.42	0.4

Table 6.3: The Comparison of *after-placement* methods

Circuits	Methods	Max. Noise	# of hot grid nodes	# of hot cells
s9234	*Baseline*	0.131	128	1046
	AllDecap	0.06	19	229
	Cell_Moving	0.055	2	23
s13207	*Baseline*	0.054	16	161
	AllDecap	0.051	2	22
	Cell_Moving	0.043	0	0
s35932	*Baseline*	0.064	57	535
	AllDecap	0.056	2	17
	Cell_Moving	0.066	3	30
s38417	*Baseline*	0.072	210	1888
	AllDecap	0.069	68	720
	Cell_Moving	0.047	0	0
s38584	*Baseline*	0.072	410	3540
	AllDecap	0.054	5	46
	Cell_Moving	0.043	0	0
AVG	*Baseline*	1.36	8.55	6.93
	AllDecap	1	1	1
	Cell_Moving	0.88	0.05	0.05



power noises, and it almost eliminates all hot grid nodes and hot cells. Our *Cell_Moving* is specially important when there is not enough empty space in a *hot_block*. For example, in benchmark s38417, after *AllDecap* method is performed, there are still 68 hot grid nodes. However, by our *Cell_Moving* step, there is no hot grid nodes left.

To analyze the efficiency of *Cell_Moving* algorithm, the third experiment is conducted to observe average power noises in *hot_block*s. In this experiment, all *hot_block*s in *Baseline* are marked, and average values of power noises in these blocks are reported. Then, the average of power noises in these

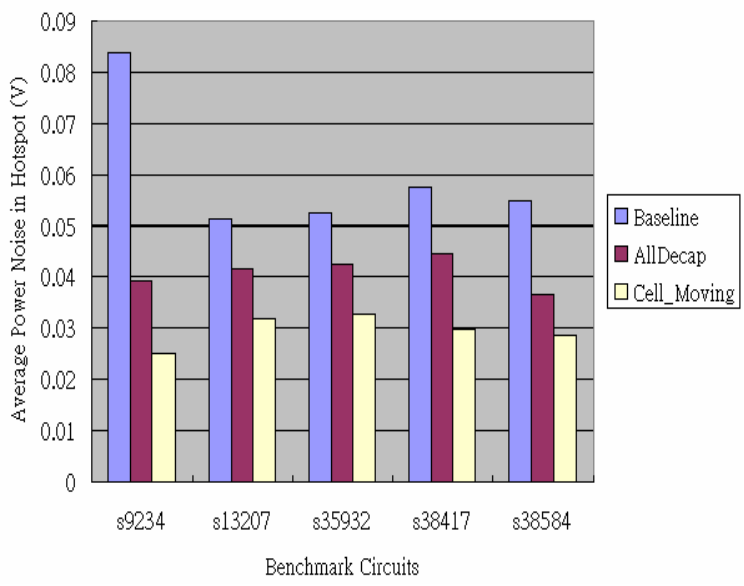


Figure 6.2: The average power noises of *hot_block*s in each benchmark

blocks after performing *AllDecap* and *Cell_Moving* are also reported. The result is shown in Figure 6.2. The horizontal axis shows benchmark circuits, and the vertical axis repots the average power noises values among *hot_block*s. By this experiment, we understand that the average amount of power noises reduction by our method is higher than the one by *AllDecap* methods because we judiciously move active cells out from and allocate *decap*s to *hot_block* in *Cell_Moving* step for each *hot_block*. The power distribution networks become even. That is why we can reduce more power noises. By these experiments, we have shown that our algorithms is very efficient in reducing dynamic power noises.

Chapter 7

Conclusions

In this paper, we have proposed two algorithms, *Decap_Padding* and *Cell_Moving*. The first algorithm, *Decap_Padding*, predicts the *hotspot* cells and binds *Decap*s to those cells. The second algorithm, *Cell_Moving*, moves cells out from *hot_block*s to further reduce *hotspot*s. The experimental result shows, compared to the previous work [1], our estimation function to allocate *decap* before placement is 23% better in reducing the number of *hotspot*s. Moreover, compared to a method which fills *decap*s to all remaining empty space, our *Cell_Moving* algorithm can further reduce 12% maximum power noises and almost eliminate all *hotspot*s.

Bibliography

[1] Chao-Yang Yeh and Malgorzata Marek-Sadowska, "Timing-aware Power Noise Reduction in Layout," *International Conference on Computer-Aided Design (ICCAD)*, pp. 627-634, Nov., 2005.

[2] Sanjay Pant and David Blaauw, "Timing-aware Decoupling Capacitance Allocation in Power Distribution Networks," *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 757-762, Jan., 2007.

[3] S. Pant, D. Blaauw, V. Zolotov, S. Sundareswaran, and R. Panda, "Vectorless Analysis of Supply Noise Induced Delay Variation," *International Conference on Computer-Aided Design (ICCAD)*, pp. 184-191 , Nov., 2003.

[4] H. Su, S. S. Sapatnekar, and S. R. Nassif, "Optimal Decoupling Capacitor Sizing and Placement for Standard-cell Layout Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, issue 4, pp. 428-436, Apr., 2003.

[5] Y. Zhong, and Martin D. F. Wong, "Fast Algorithms for IR Drop Analysis in Large Power Grid," *International Conference on Computer-Aided Design (ICCAD)*, pp. 351-357 , Nov., 2005.

[6] M. Popovich, E. G. Friedman, R. M. Secareanu, and O. L. Hartin, "Efficient Placement of Distributed On-chip Decoupling Capacitors in Nanoscale ICs," *International Conference on Computer-Aided Design (ICCAD)*, pp. 811-816 , Nov., 2007.

[7] Ang-Chih Hsieh, Tzu-Teng Lin, Tsuang-Wei Chang, and TingTing Hwang, "A functionality-directed clustering technique for low-power MTCMOS designXcomputation of simultaneously discharging current," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*.

[8] H. Kriplani, F. N. Najm, and I. N. Hajj, "Pattern Independent Maximum Current Estimation in Power and Ground Buses of CMOS VLSI Circuits: Algorithms, Signal Correlations, and Their Resolution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 998-1012, 1995.

[9] L. T. Pillage, R. A. Rohrer, and C. Visweswariah, "Electronic Circuit and System Simulaton Methods," *McGraw-Hill*, 1995.

[10] Bo Hu and Malgorzata Marek-Sadowska, "Wire Length Prediction Based Clustering and Its Application in Placement," *Design Automatin Conference (DAC)*, pp. 800-805, 2003.

[11] M. Pan, N. Viswanathan, and C. Chu, "An Efficient and Effective Detailed Placement algorithm," *International Conference on Computer-Aided Design (ICCAD)*, pp. 48-55, 2005.

[12] S. Zhao, K. Roy, and C. K. Kon, "Decoupling Capacitance Allocation and Its Application to Power-Supply Noise-Aware Floorplanning", *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems (Special Issue on Physical Design)*, 21(1), january 2002, pp. 81-92.

[13] R. Dutta and M. M. Sadowska, "Automatic sizing of power/ground networks in VLSI," in *Proc. Design Automation Conf.*, June 1989.

[14] K.-H. Erhard, F. M. Johannes, and R. Dachauer, "Topology optimization techniques for power/ground networks in VLSI," in *Proc. Eur. Design Automation Conf.*, pp. 362-367, Mar. 1992.

[15] M. Ang, R. Salem, and A. Taylor, "An on-chip voltage regulator using switched decoupling capacitors," in *Proc. Int. Solid-State Circuits Conf. Dig. Tech. Papers*, pp. 438-439, Feb. 2000.

[16] B. Obermeier, F. M. Johannes, "Temperature-aware global placement," *Proceeding of the 2004 conference on Asia South Pacific design automation: electronic design and solution fair 2004*, January 27-30, 2004, Yokohama, Japan.