國立交通大學

電子工程學系 電子研究所

碩 士 論 文

考慮電壓島產生之低功率平面規劃方法

Low Power Floorplanning Methodology Considering Voltage Island Generation

研 究 生 ：李泓懌

指導教授 ：陳宏明 教授

江蕙如 教授

中 華 民 國 九 十 八 年 十 月

考慮電壓島產生之低功率平面規劃方法

Low Power Floorplanning Methodology Considering Voltage Island Generation

學生：李泓懌　　Student: Houng-Yi Li

指導教授：陳宏明　　Advisor: Prof. Hong-Ming Chen

江蕙如　　Prof. Iris Hui-Ru Jiang

國立交通大學

電子工程學系　電子研究所碩士班

碩士論文

A Thesis
Submitted to Department of Electronic Engineering & Institute of Electronics
College of Electrical Engineering and Computer Science
National Chiao Tung University
In Partial Fulfillment of Requirements
for the Degree of
Master of Science
In
Electronic Engineering
Oct. 2009
Hsinchu, Taiwan, ROC

中華民國九十八年十月

考慮電壓島產生之低功率平面規劃方法

學生：李泓懌　　　　　　　　　　　　　　　　　　　　　　　　　　指導教授：陳宏明
江蕙如

國立交通大學　電子工程學系　　電子研究所　碩士班

摘　要

隨著製程技術進入了奈米的紀元，以往隨著製程縮小所得到的功率節省也因此慢下來；然而，在現代的設計中，高時脈頻率和複雜的功能造成顯著的功率密度增加。多重供應電壓是一項可以平衡功率和效能的既普遍又有效率的技術。考慮多重供應電壓的技術，我們把一個設計分割成多個電壓島，每個電壓島分別在平面規劃上佔有它的區域並且工作在一個特定的電壓。 在這篇論文中，我們把電壓島的產生跟平面規劃合併在一個有效率並且使用決定論演算法的平面規劃器裡面。給予一組區塊和所對應的可工作的電壓，我們使用動態規劃去產生一組已經有電壓指定並且功率消耗已經被降至最低的平面規劃。與前人的作品比較起來，實驗結果顯示我們的演算法在執行時間和功率消耗上可以保證有絕對的勝出，尤其在比較龐大的設計上。

Low Power Floorplanning Methodology Considering Voltage Island Generation

Student: Houng-Yi Li

Advisor: Hung-Ming Chen
Iris Hui-Ru Jiang

Department of Electronic Engineering
Institute of Electronics
National Chiao Tung University

Abstract

As technology advances into nanometer era, the power benefit from process scaling slows down; however, the high clock rate and the complex functionality of a modern design result in a significant growth in power density. Multiple supply voltage is a prevalent and effective technique to balance power and performance. Considering multiple supply voltage, a design is divided into voltage islands, where each island occupies a physical region of the floorplan and operates at a certain level of supply voltage. In this thesis, we combine voltage island creation with floorplanning based on an efficient and deterministic floorplanner. Given a set of blocks and the acceptable voltage levels for each block, we use dynamic programming to generate a floorplan with voltage assignment such that power consumption is minimized. Compared with prior work, experimental results show that our algorithm is promising in running time and power, especially for large design cases.

致　　謝

這本書得以完成，首先要感謝兩位指導老師陳宏明老師和江蕙如老師對我的耐心協助以及適時的給予意見，讓我能夠順利的把研究完成。

感謝愛荷華州大的 Jackey Z. Yan 和 Chris Chu 在我實作上給予我很多建議。

感謝香港中文大學的 Qiang Ma 和 Evangeline F.Y. Young 提供的執行檔讓我得以比較。

感謝柏丞同學總是不厭其煩的對我伸出援手，感謝仁傑學長時常為我解惑， 感謝篤雄學長的幫忙，感謝實驗室的大家提供了這麼好的氣氛跟研究環境，還有不時的給予我幫助。

感謝家人的支持與鼓勵，是支撐我的精神食糧！

Contents

1 Introduction **1**

1.1 Previous Work…...2

1.2 Our Contributions…..3

2 Preliminaries **5**

2.1 Review of The Slicing Tree Representation….5

2.2 The Shape Curve of A Slicing Tree…....6

2.3 Define The Number of Voltage Islands…...7

2.4 Problem Formulation….9

3 Methodology **10**

3.1 Overview….10

3.2 More Conditions On Partitioning…...12

3.3 Modified The Generalized Combined Operator…..13

3.4 High Level Combining and Back-tracing…...17

4 Experimental Results **20**

5 Conclusion and Future Work **24**

List of Figures

2.1 An example of a slicing tree and the corresponding slicing floorplan……………………………..5
2.2 The shape curve of a slicing tree………………………………………………………………...6
2.3 Voltage islands in a rectangular shape…………………………………………………………..7
2.4 Voltage islands in an arbitrary shape…………………………………………………………....8
2.5 An extreme case of voltage assignment……………………………………………………….....8
2.6 An example of physical layout of a slicing tree………………………………………………....9
3.1 An example to illustrate '⊕'……………………………………………………………………...14
3.2 Input information of an example…………………………………………………………………15
3.3 The Structure Table and the Level Table of Fig. 3.1……………………………………………...15
3.4 The pruned Level Table and corresponding Cost Table…………………………………………..16
3.5 An example to illustrate the swapping process…………………………………………………...17
3.6 An example of the back tracing step……………………………………………………………...18
4.1 The Resultant Floorplan for n300………………………………………………………………...23

List of Tables

4.1 Experimental results on GSRC benchmark n10……….…………………..…………………………….…...21
4.2 Experimental results on GSRC benchmark n30………………………………………………………..21
4.3 Experimental results on GSRC benchmark n50…………..………………………………………21
4.4 Experimental results on GSRC benchmark n100………...……………………...……….……..…22
4.5 Experimental results on GSRC benchmark n200……………..…………………...…….……..…22
4.6 Experimental results on GSRC benchmark n300…………..……………………...…….……..……22

Chapter 1

Introduction

As the continuous scaling-down of CMOS technology, the circuit size is becoming larger. To cope with the increasing design complexity, hierarchical design and reuse IP (Intellectual Property) is widely used [1] [7]. And as the first stage of the physical design hierarchy, an efficient and effective floorplan would highly affect the overall chip performance. Meanwhile, how to efficient power dissipation has become one of the most important issues since the power density is also increasing as the circuit size is increasing. There are many techniques has been applied to reduce the power consumption of a VLSI design. Of all low power techniques, multiple supply voltage (MSV) is the most effective method.

The idea of MSV is to provide the voltage level that is just enough for a block to work correctly. Not only dynamic power could be reduced by MSV since the consumption of dynamic power is proportional to the square of the supply voltage, but the consumption of static power could also be lower by MSV. In order to lower the usage of power network resources, we partition the chip into areas called “voltage islands.” Blocks in each voltage island are assigned to the same voltage level.

1.1 PREVIOUS WORK

There are several previous papers addressing similar low power floorplanning using voltage islands problem. Ma and Young [2] proposed a floorplanning method for SOC designs that is tightly integrated with the island partitioning and voltage assignment steps. Annealing based floorplanner is applied in their approach. First, they recursively assign voltage level to the sub-trees under a root of an initial slicing tree which represents a slicing floorplan. For each sub-tree, they are given the number of voltage islands that is available. Then they would optimize the power cost under this sub-tree by using dynamic programming. After voltage assignment, they start to perturb the initial floorplan and see if they need to take more iteration. The voltage islands formed by their approach would only be in rectangular shape since a sub-tree could always be in rectangular shape.

Lee et al. [4] generate a voltage island aware floorplan and the power is optimized under a timing constraint. There are three phases in this work. First, given a netlist without reconvergent fanouts, they handle voltage islands partitioning by dynamic programming which can guarantee an optimal solution for the voltage assignment in linear time. In the second phase, because level shifters are needed when a VDDL block drives a VDDH block, level shifters are introduced and treated as soft blocks during floorplanning. In the last phase, a power-network aware floorplanner would be conducted to pack the blocks such that the power-network resource will be minimized while the critical paths satisfy the timing constraint. The voltage assignment step and the floorplanning step are done separately in their algorithm flow.

Ma and Young [5] proved that the voltage assignment problem could be solved optimally under timing constraint by their approach. They solve this problem by presenting a general formulation as a set of linear inequalities to represent the timing requirements. Then they transform this problem into a convex cost network flow problem, and they solve it optimally by a cost-scaling algorithm in polynomial time. This is still a work using floorplanner based on simulate annealing.

One of the recent works by Mak and Chen [6] has also formulated this problem on SOC designs. Given a floorplanning input, the voltage assignment and island partitioning problem is formulated as a 0-1 integer linear program. In their approach, they take advantage of the voltage island technique to reduce the power consumption while carefully taking the level shifter overhead and power network complexity into account. Although a fragmentation cost (number of adjacent cores operating at different voltages) is used to model the power network complexity, this cost is not related to the number of islands directly.

1.2 OUR CONTRIBUTIONS

How to generate a floorplan efficiently is a very important issue since floorplanning is the beginning of modern physical design flow. An efficient floorplanning would extend the space for doing the rest steps in physical design flow such as placement, routing, and manufacturing. Simulated annealing has been the most popular method of exploring good solutions on the floorplanning problem. However, one common drawback of most annealing based floorplanners is that the execution speed would become quite slow when the size of circuits grows large. Furthermore, it is always a difficult topic for the annealing based floorplanners to handle circuits with soft modules, because they need to search an extremely large solution space, which takes a long time to cool down and reach the target temperature.

Our method could bottom up generate a great number of possible layouts of the input block set, and the voltage assignment of each block is also done. Then we could pick one solution that is most fit into our requirements. The requirement includes area, wire length, and by our method, power cost. No more iteration needs to be taken since all the conditions are already acceptable. In our approach, the voltage assignment step and the floorplanning step are done simultaneously.

In this thesis, we handle the low power floorplan by using MSV technique and slicing tree representation. The experimental result shows that our method would efficiently generate a floorplan with voltage islands. Meanwhile, the cost which includes power, area, and wire length is acceptable. Moreover, if we want to change the specific weight of the cost, which means if we want to change the weight of area, power, wire length, or even to minimize any one of them, we do not need to take any other iteration, we could just pick a solution that is most fit into this new cost from the solution space. The back-tracing step is the step which after we select a solution and then we trace back from the root of this slicing tree to find its corresponding voltage assignment. And the back-tracing step would not take extra effort to map to the assigned voltage level since our method could generate the corresponding voltage assignment of this floorplan by dynamic programming in a table which could directly map to its structure without searching. Experimental results based on the GSRC benchmark with voltage information that is randomly generated.

The remainder of this thesis is organized as follows. In Chapter 2 we briefly review slicing tree representation. Also, we would describe the meaning of a shape curve of a slicing tree, definition of voltage islands, and give the problem formulation. In Chapter 3, we discuss our methodology for dealing with voltage assignment based on a deterministic floorplanner. Our experimental results are presented in Chapter 4. Finally we give the conclusion of this thesis and future work in Chapter 5.

Chapter 2

Preliminaries

2.1 REVIEW OF THE SLICING TREE REPRESENTATION

A slicing floorplan [3] could be represented by a binary tree. And this binary tree which is called slicing tree is a binary tree with two circuits at the two leaves and cut types at the internal nodes. The cut types include horizontal cut (H cut) and vertical cut (V cut). The H cut divides the floorplan horizontally, and the left child represents the bottom sub-floorplan while the right child represents the top one. Similarly, the V cut divides the floorplan vertically, and the left child represents the left sub-floorplan while the right child represents the right one. There is an example of a slicing tree and the corresponding slicing floorplan in Fig. 2.1.

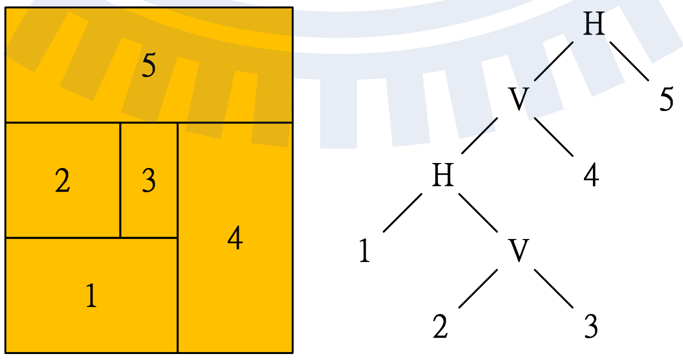
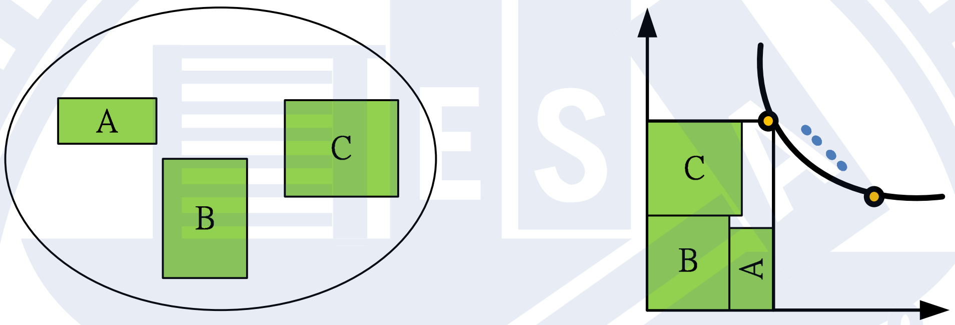


Fig. 2.1 an example of a slicing tree and the corresponding slicing floorplan

2.2 THE SHAPE CURVE OF A SLICING TREE

The idea of a shape curve is first introduced in [10] to record the possible layouts of a circuit so that we could find an optimal floorplan among all slicing structures consistent with a given plane point placement by some shape curve operations and constraints.



(a) The input block set (b) The possible layouts of (a) are kept in a shape curve

Fig. 2.2 The shape curve of a slicing tree

In Fig. 2.2 (a), there are three blocks in the input block set. The shape curve in Fig. 2.2 (b) is formed by all the possible layouts of the input block set. Each point in this shape curve could map to the corresponding slicing tree structure which represents a slicing floorplan. An operator '$\oplus$' [9] is introduced here to replace H or V in a slicing tree in this thesis. And the idea of using '$\oplus$' [9] to replace H or V in a slicing tree is to defer the decision of 'H' or 'V' of a slicing tree rather than only generalize the orientation for each module as in [11]. So when we use this operator in a slicing tree, there are many possible layouts under this operator since we have not decided what kind of cut this tree has. We keep these layouts in a shape curve. The three steps to actualize the combination of two shape curves are fusing with the operator '$\oplus$'. The three steps to combine two child curves A and B into curve C are as follows.

1. **Addition:** First, adding two curves A and B horizontally to get curve Ch, on which each point maps to a horizontal combination of two subfloorplan layouts from A and B.
2. **Flipping:** Next, flipping curve Ch symmetrically based on the W = H line to derive curve Cv. The purpose of this step is to generate the curve that contains the corresponding vertical combination cases from the two subfloorplan layouts.
3. **Merging:** finally, merging Ch and Cv into the parent curve C. In here, for a given height, the point with a smaller width out of Ch and Cv will be reserved while others would be pruned.

2.3 DEFINE THE NUMBER OF VOLTAGE ISLANDS

In this section, we would explain how we define the number of voltage islands. In Fig. 2.3, the blocks in the same color could be assigned to the same voltage level, and obviously there are two rectangular shape voltage islands in this circuit. The corresponding working level of this two voltage islands is 1.1V and 1.2V. Furthermore, not only rectangular shape could form voltage islands, but there are also some other shape could form voltage islands.

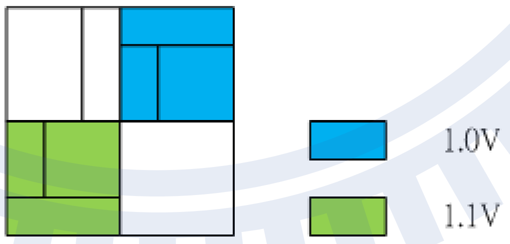


Fig. 2.3 Voltage islands in a rectangular shape

In Fig. 2.4, it could still be counted as two voltage islands since the blocks in the same color could be assigned to the same voltage level, and these blocks are stick together in this figure.

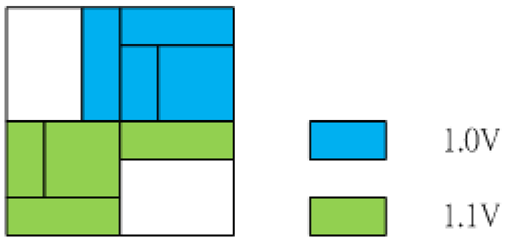


Fig. 2.4 Voltage islands in an arbitrary shape

An extreme case would happen in our approach. In Fig. 2.5, block C is in the middle of a voltage island which is formed by five blocks A, B, D, E, and F. There would be a serious problem. That is, if we merge these five blocks into one voltage island, then this voltage island's supply voltage would be one I/O pad, however, block C is assign to another voltage level, how do block C connect to its I/O pad since it is isolated.

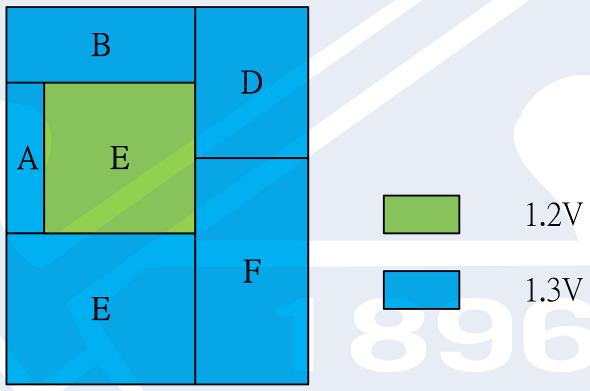


Fig. 2.5 An extreme case of voltage assignment

Here we take advantage of the slicing tree representation. The left side of Fig. 2.6 is showing that a slicing tree of block A and B. The middle of Fig. 2.6 is showing the corresponding floorplan. The right side of Fig. 2.6 is showing that the physical layout of block A and B is actually not perfectly match. Therefore, block C in Fig. 2.5 could physically find a way out and connect to its supply pad. There is no need to assign other common voltage level between block C and surrounding blocks in order to connect to the I/O pad. In the case of Fig. 2.5, we could still count it as two voltage islands.

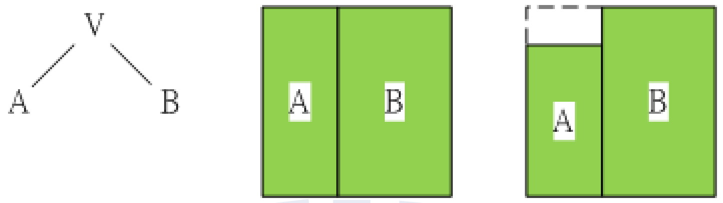


Fig. 2.6 An example of physical layout of a slicing tree

2.4 PROBLEM FORMULATION

Given a set of n soft blocks with area A_1, A_2 … A_n, and the corresponding aspect ratio bounds [l_i, U_i] for i = 1…n. Each block i is associated with a Cost Table Ti that records the legal voltage levels for the block and the corresponding average power consumption values. The legal voltage levels of a block are characterized by designer. As long as the rules of timing constraint could be satisfied after simulations, designers could regard this voltage as a legal one. The power consumption corresponding to each legal voltage can then be estimated. In this work, we compute the power cost of a block i operated at voltage v as v^2A_i. Also, we are given a set of m nets {N_1, N_2, …, N_m}, and the connections of these nets are among blocks. Given a cost function $\psi = \lambda_A A + \lambda w W + \lambda p P$ where A is the area of the floorplan, W is the total wire length estimated by the half perimeter bounding box method and P is the total power cost. Our goal is to generate a floorplan F that the cost ψ is minimized. Or we could just select a local minimized solution (for example, while focus on area minimization, then λw and λp could be tuned to zero) in the solution space which is built by us in a bottom up process.

Chapter 3

Methodology

3.1 OVERVIEW :

There are three main steps in our methodology. They are as follows.

(1) Partitioning :

We cannot take the price to explore the whole slicing layout of all blocks since the price would be too expensive. Therefore, we divide the original input block set into several small sets by hMetis [8], the state-of-the-art hypergraph partitioner. In this step, we not only initially minimize the interconnections among blocks, but by treating voltage level as a cut parameter, we could also initially divide the originally input block set into voltage island with higher probability.The limitation of block number to each sub-circuit in this step is set to ten by default. For each sub-circuit, we will use enumerative packing to generate all the possible tree structures and layouts. While packing, we also modify the generalized operator of a slicing tree.

(2) Combining and swapping :

In this step, a modified generalized operator is introduced. Enumerative packing (EP) [9] is applied to find all possible combinations of blocks within a sub-circuit we get from last step. While packing, we use modified generalized operator to combine two shape curves and their corresponding power information. The origin idea of this operator is to defer the decision on a slicing tree structure, and use shape curve to represent the possible layouts under this operator. But the function of this operator cannot just combine the shape curve, which represents the possible layouts of a sub-circuit, it also has to generate the power information of this sub-circuit since we want to solve low power problem.

Here we use dynamic programming to grab the power information we need. Each modified generalized operator would record all the possible voltage level combinations (or the voltage level combinations sifted by a cost function) of the left child and the right child of this operator. Under this operator, we are also able to search whether there are islands could be merged if they are assigned the same working level.

The total framework of step is first to get all tree structures by EP and their corresponding available working levels. After obtaining the shape curves and the power information of all sub-circuits, we apply enumerative packing on high level to obtain the final shape curve. The whole framework is built in a bottom-up process.

(3) Back-tracing :

Now, we already have the shape curve and the corresponding power information. We could select a point that is fitting into our limitation from the final shape curve. This point we select from the final shape curve represents the floorplan of whole input block set. We can start the back-tracing process from this point which is the root of the final slicing tree. Whichever point pick from the final shape curve, the orientation of each block and the voltage assignment of each block are already be decided.

From the overview of the total methodology, we know that first we partition our circuit into many sub-circuits in step one. Second, in step two we generate a huge collection of possible layouts which is kept in a shape curve and their corresponding power information which is explored by dynamic programming. Finally we choose a point which is sifted by a cost function from the final shape curve in step three. We are going to discuss the detail of each part in our methodology in the following sections.

3.2 MORE CONDITIONS ON PARTITIONING :

Originally we only use hMetis to partition the input circuit into sub-circuits by min-cut. Here we not only take a net between two blocks as a cut if two blocks connected by a net are belong to different partitions, but voltage levels also must be treated as the connections among blocks.

How to decide the weight of each connection is a very important issue. Here we first transform a voltage level (V_k) as a net, and then we find the blocks which could be assigned to this level. Second, the sum of the square of a voltage level V_k multiplied by the area A_i ($i_k = 1, 2, \ldots, n_k$) of each block which is connected to this level would be treated as a parameter λ (see Eq. 1) while we want to calculate the weight of the net transformed by this voltage level. The parameter $\lambda_k$and $\lambda'_k$is calculated as follows:

$$\lambda_k = (\textstyle\sum_{i_k=1}^{n_k} A_{i_k}) \times {V_k}^2 \qquad (1)$$

$$\lambda_k = (\textstyle\sum_{i_k=1}^{n_k} A_{i_k}) \times {V_c}^2 - \lambda_k \qquad (2)$$

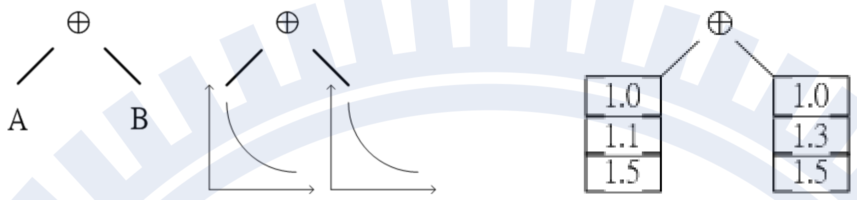
In Eq. 2, we define λ' to be the power saving of a net formed by V_k where V_c is the chip level voltage. We take λ' multiplied a constant to be the weight of this net, and the meaning of this weight is trying to seprate blocks which could be assigned to this level into the same partition as possible.

3.3 MODIFIED THE GENERALIZED COMBINE OPERATOR :

In this section, we modify the origin idea of the generalized combine operator to handle the power cost problem. The generalized combine operator '$\oplus$' is originally to defer the decision of 'H' or 'V' of a slicing tree. It only keeps some layouts of a sub-circuit in a shape curve efficiently. Now, we use dynamic programming to generate and record the possible power combinations of the two children under '$\oplus$'.

A. The specifics of the modified '$\oplus$' :

There are two main parts of this new operator. The first part is to deal with the shape curve of the two sub-circuits. In the mean while, the second part that combines the power information of the two sub-circuits is also take place. Under the operator '$\oplus$', the "left-right" or the "top-bottom" order have no difference since the shape curve would keep these possible solutions. Furthermore, the possible voltage assignment combinations are kept in the "Level Table". Fig. 3.1 gives an example of the two main parts of '$\oplus$'. In Fig. 3.1 (a), '$\oplus$' generates the shape curve of sub-circuit A and sub-circuit B. In the mean time, we collect the available voltage level information of each sub-circuit and '⊕' would generate the possible level combinations of these two sub-circuits and keep the information in the level table by dynamic programming.



(a) Shape curve part of '⊕' (b) Available level of each block

Fig. 3.1 An example to illustrate '⊕'

The Level Table is a table to record the possible power combinations, and the order to fill this table is completely the same to the "Structure Table". We use Structure Table to record the possible tree structure under an operator '⊕'. Also, we will have to record the cost information to pick a minimum cost solution in the back-tracing step.

B. The process of combining :

An example of this process would be shown in Fig. 3.2 and Fig. 3.3. In Fig. 3.2, there is a block set {A, B, C, D} includes four blocks. Each block has its own possible working levels. Now, we are going to generate the possible slicing tree structures of this block set. In Fig. 3.3 (a), we first record the shape curve of each block. If the input block is hard block, the number of point which is kept in the shape curve in the first floor of the Structure Table is two. At the same time, a Level Table is generated in Fig. 3.3 (b). The size of this table is just equal to its corresponding Structure Table. In the first floor of Level table, we record the corresponding possible level combinations in the same location as where the structure is in the Structure Table so that we could directly map to the corresponding voltage assignment without searching when back-tracing step.

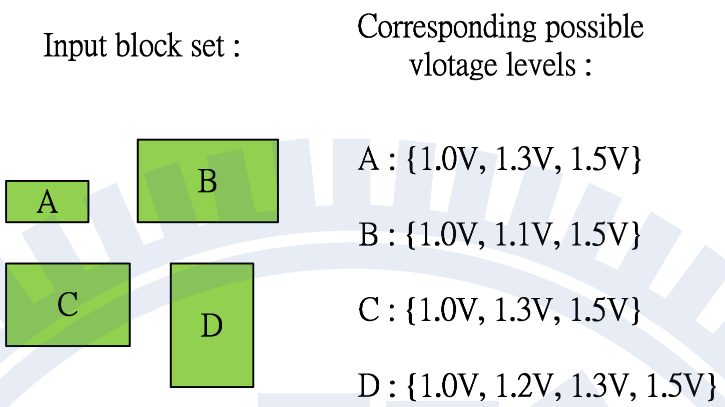


Fig. 3.2 Input information of an example

The second floor of the Structure Table is to record the combinations of any two blocks. Meanwhile, the possible working level combinations are also recorded in the second floor of the Level Table. Deduced by analogy, we could complete every floor by a bottom-up process. It is deserve to be mentioned that in Fig. 3.3 (a) we use dynamic programming to generate the shape curve, so when we build up the larger subset of modules we could just reuse the previously generated shape curves without redundant computations. And on the top floor are the final shape curve and the corresponding voltage assignment to this input block set {A, B, C, D}.

ABCD					
ABC	ABD	ACD	BCD		
AB	AC	AD	BC	BD	CD
A	B	C	D		

(a) Structure Table

ABCD					
AB/1.0 AB/1.1 ⋮	AB/1.0 ⋮	AC/1.0 ⋮	BC/1.0 ⋮		
1.0/1.0 1.0/1.1 1.0/1.5 1.3/1.0 ⋮	1.0/1.0 1.0/1.3 ⋮	1.0/1.0 1.0/1.2 ⋮	1.0/1.0 ⋮	1.0/1.0 ⋮	1.0/1.0 ⋮
1 1.3 1.5	1 1.1 1.5	1 1.3 1.5	1 1.2 1.4 1.5		

(b) The Corresponding Level Table to (a)

Fig. 3.3 The Structure Table and the Level Table of Fig. 3.1

ABCD					
AB/1.0 ~~AD/1.1~~ ⋮	AB/1.0 ⋮	AC/1.0 ⋮	BC/1.0 ⋮		
1.0/1.0 ~~1.0/1.1~~ ~~1.0/1.5~~ 1.3/1.0 ⋮	1.0/1.0 ~~1.0/1.3~~ ⋮	1.0/1.0 ~~1.0/1.2~~ ⋮	1.0/1.0 ⋮	1.0/1.0 ⋮	1.0/1.0 ⋮
1 1.3 1.5	1 1.1 1.5	1 1.3 1.5	1 1.2 1.4 1.5		

(a) A pruned Level Table

PABCD1 ⋮					
PABC1 ⋮	PABD1 ⋮	PACD1 ⋮	PBCD1 ⋮		
PAB1 PAB2 ⋮	PAC1 PAC2 ⋮	PAD1 PAD2 ⋮	PBC1 ⋮	PBD1 ⋮	PCD1 ⋮
PA1 PA2 PA3	PB1 PB2 PB3	PC1 PC2 PC3	PD1 PD2 PD3 PD4		

(b) The corresponding Cost Table of (a)

Fig. 3.4 The pruned Level Table and corresponding Cost Table

$$\varphi = \lambda_n n + \lambda_p p \qquad (3)$$

The Level Table would record the corresponding voltage assignment of the sub-circuit. However, if we want to generate the whole voltage assignment possibilities, the memory cost would be unacceptable. Here we introduce a cost function (Eq. 3) to sift the voltage assignment combinations with good solution quality, where n represents the number of voltage islands and p is the power cost. And then we keep these sifted solutions in a Cost Table. We sift the voltage level combinations in Fig 3.3 (b) by Eq. 3 and the sifted solutions are shown in Fig. 3.4 (a). Solutions that satisfy the inequality $\varphi - \epsilon \leq \varphi \leq \varphi + \epsilon$ would be kept, and others would be pruned, where ϵ is a constant.

C. Swapping details :

As the process of combining is proceeding, we will have to check whether we swap the left child and the right child of a '$\oplus$' would physically merge a voltage island or not. If yes, then the swapping action must be taken. Because of the definition of the number of voltage islands we mentioned in section 2.3, the shape of a voltage island could be not just rectangular. As a result, the swapping step would merge voltage islands without considering whether this island is rectangular or not. We would illustrate the swapping concept in Fig. 3.5.

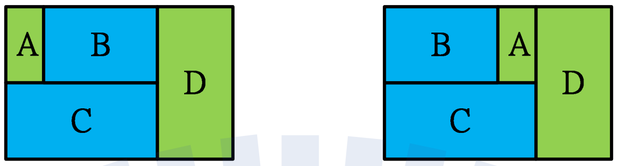


Fig. 3.5 An example to illustrate the swapping process

In Fig. 3.5, we assume that block A and D could be assigned to the same voltage level while block B and C could be assigned to the same voltage level but this level is different from block A and D. Now, as the layout shown in the left part of Fig. 3.5, we would count the number of voltage islands as three since block A and D is physically separated. But if we swap block A and B, we would be able to count the number of voltage islands as two. Here we would check the two children of a '$\oplus$' to see if there are any sub-tree under this '$\oplus$' could be merged with another sub-tree. If yes, then we would lock the '$\oplus$' of that sub-tree and record the power cost in the Cost Table after swapping. If a '$\oplus$' is locked, then we would physically swap this sub-tree in the back tracing step.

3.4 HIGH LEVEL COMBINING AND BACK-TRACING

In order to avoid the possibility of the form of large dead space caused by the usage of Intellectual property (IP) which contains numbers of big macros, we pack the sub-circuits by high level EP. Treat every sub-circuit as a block in EP, we pack these sub-circuit, which we already have their shape curves, to generate the final shape curve.

During high level combining of two circuits, we still could swap the sub-trees of the two circuits to merge the voltage island if the voltage level is the same. First we check the Level Table of the left child and the right child of each circuit whether they could be assigned to the same voltage level. If yes, then we could merge these two voltage island into one. Else we map to the Structure Table and trace back to check whether the sub-trees satisfy the merge conditions. If the merge conditions are satisfied and this sub-tree is not locked in the previous combining and swapping step, then we could lock this sub-tree and physically swap it in back tracing step.

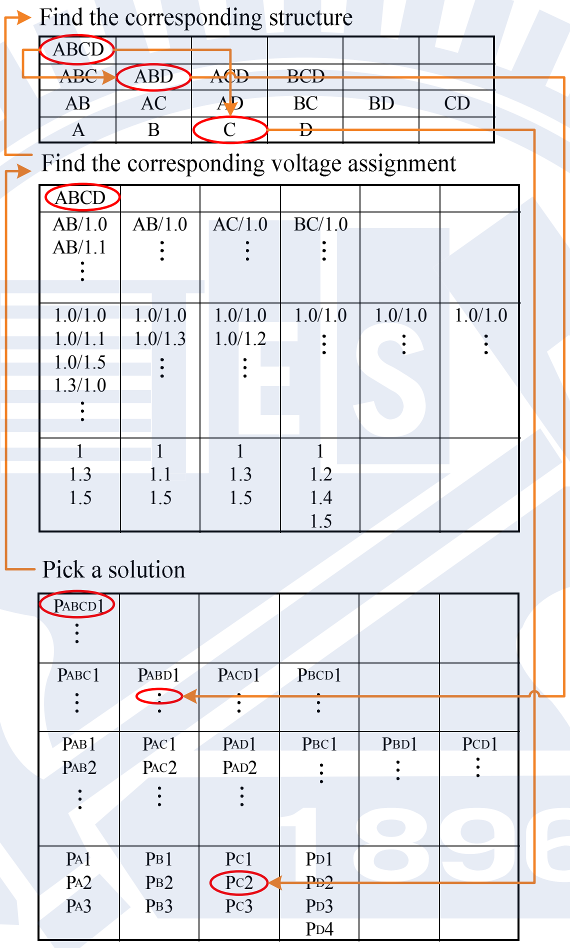


Fig. 3.6 An example of the back tracing step

Once the final shape curve has been generated, the voltage assignment to each circuit and the power cost to each voltage assignment are all available. Now we combine power, number of voltage islands, area, and wire length into our total cost, and then we pick the minimum cost one to be our final floorplan and the corresponding voltage assignment to each block.

Here we give an example in Fig. 3.6 to illustrate how the back tracing step proceed. There are three tables in this figure, a Cost Table, a Level Table and a Structure Table. These three tables are from Fig. 3.3 and Fig. 3.4. First, we pick a minimum cost solution from the Cost Table. Then we find the corresponding voltage assignment in exactly the same location as in the Cost Table in the Level Table. Similarly, we could find the tree structure from a shape curve in the Structure Table. Finally, we could top down and recursively trace back to get the final floorplan and the voltage assignment to each block.

Chapter 4

Experimental Results

We have done experiments on the GSRC soft blocks floorplanning benchmarks. Since no voltage information is provided in those benchmarks, we use the voltage information that is from Qiang Ma and Evangeline F.Y. Young's previous work [2], which have randomly generated the voltage levels for each block from the set {1.0V, 1.1V, 1.2V, 1.3V, 1.5V} and 1.5V is assumed to be the chip-level voltage. Our algorithm is implemented in the C++ programming language and all experiments were performed on a Linux machine with Intel Core 2 Duo 2 GHz CPU and 2GB memory. The wire length is calculated by using half perimeter bounding box. We compare our algorithm with [2] and the number of voltage islands generated range is from zero to four.

In the benchmark of small size circuits such as n10, and n30 in Table 4.1 and Table 4.2, it is not obvious to see the difference between these two approaches in power, run time, and area. But the wire length is averaged about 16% losing to [2] in our approach. This is because the connections between blocks could initially been cut in the partitioning step in order to group blocks with the same working levels together. But the speed up here is averaged about 4.8.

Table 4.1 Experimental result on GSRC benchmark n10

Data	# of islands	Power			Run time(s)			HPWL			Area		
		Ours	[2]	Improve	Ours	[2]	Speed up	Ours	[2]	Improve	Ours	[2]	Improve
N10	Vi = 0	498778	498778	0%	0.047	0.78	16.60	13074	13250	1.30%	223309	222555	0%
	Vi = 1	400655	393660	-1.70%	0.505	1.17	2.32	17858	13865	-28.70%	225009	223918	0%
	Vi = 2	359287	352637	-1.80%	0.505	1.41	2.79	15582	13147	-19%	224292	222530	-0.70%
	Vi = 3	301464	309498	2.60%	0.505	1.72	3.41	16742	13107	-27%	223617	224842	0.50%
	Vi = 4	285083	295682	3.60%	0.505	1.8	3.56	17707	13903	-27%	224928	222977	-0.80%
Average Improvement				0.50%			5.73			-20%			-0.40%

Table 4.2 Experimental result on GSRC benchmark n30

Data	# of islands	Power			Run time(s)			HPWL			Area		
		Ours	[2]	Improve	Ours	[2]	Speed up	Ours	[2]	Improve	Ours	[2]	Improve
N30	Vi = 0	469330	469330	0	1.17	7.64	6.53	37948	34090	-11%	214555	211720	-1%
	Vi = 1	346487	370179	6.40%	3.22	10.08	3.13	43080	35907	-19.90%	214067	215475	-0.38%
	Vi = 2	329564	338939	2.76%	3.648	11.17	3.06	38530	38371	-0.40%	217652	213974	-1.70%
	Vi = 3	303276	318792	4.86%	3.569	11.42	3.20	41750	37687	-10.80%	214508	217452	1.30%
	Vi = 4	247134	297259	16.80%	3.479	15.13	4.35	45139	36469	-23%	214420	210206	-2%
Average Improvement				7.70%			4.05			-13%			0%

Table 4.3 Experimental result on GSRC benchmark n50

Data	# of islands	Power			Run time(s)			HPWL			Area		
		Ours	[2]	Improve	Ours	[2]	Speed up	Ours	[2]	Improve	Ours	[2]	Improve
N50	Vi = 0	446802	446802	0	2.13	25.17	11.82	89432	79448	-12.50%	203803	200465	-1.60%
	Vi = 1	305227	310796	2%	7.54	25.82	3.42	98856	86723	-13%	204860	212706	3.70%
	Vi = 2	283445	292137	3%	7.54	28.42	3.77	95110	90074	-5.50%	208107	221209	5.90%
	Vi = 3	262264	279442	6%	7.61	26.4	3.47	98681	93934	-5%	204466	230740	11.30%
	Vi = 4	238193	245203	6%	7.813	35.08	4.49	97143	92425	-5%	204884	223999	8.50%
Average Improvement				2.70%			5.39			-8.4%			5.6%

Table 4.4 Experimental result on GSRC benchmark n100

Data	# of islands	Power			Run time(s)			HPWL			Area		
N100		Ours	[2]	Improve	Ours	[2]	Speed up	Ours	[2]	Improve	Ours	[2]	Improve
	Vi = 0	403877	403877	0	6.97	87.52	12.56	130030	144325	10%	185890	180674	-2.88%
	Vi = 1	338986	354566	4%	16.15	93.85	5.81	156137	133389	-17%	186833	187139	0%
	Vi = 2	284050	319961	11%	17.078	106.22	6.22	162128	133912	-21%	187039	187726	0%
	Vi = 3	267786	286636	7%	17.113	108.27	6.33	156222	141916	-10%	187079	189106	1%
	Vi = 4	259479	265173	2%	17.078	105.12	6.16	148151	140141	-5%	187212	190363	1.60%
Average Improvement				6%			7.41			-9%			0%

Table 4.5 Experimental result on GSRC benchmark n200

Data	# of islands	Power			Run time(s)			HPWL			Area		
N200		Ours	[2]	Improve	Ours	[2]	Speed up	Ours	[2]	Improve	Ours	[2]	Improve
	Vi = 0	395316	395316	0%	12.37	287.01	23.20	251832	293015	14%	176238	177028	0.40%
	Vi = 1	319205	323634	1.30%	24.711	304.72	12.33	311501	325514	4.30%	184348	180936	-1.80%
	Vi = 2	285591	302733	5.70%	24.04	425.23	17.69	327764	299397	-9.40%	184395	182840	0%
	Vi = 3	274482	290853	5.60%	24.508	441.16	18.00	314456	330801	4.90%	184189	186078	1%
	Vi = 4	260458	269593	3.30%	30.436	455.37	14.96	321492	327422	1.80%	184213	184680	0%
Average Improvement				4%			17.24			3.9%			0%

Table 4.6 Experimental result on GSRC benchmark n300

Data	# of islands	Power			Run time(s)			HPWL			Area		
N300		Ours	[2]	Improve	Ours	[2]	Speed up	Ours	[2]	Improve	Ours	[2]	Improve
	Vi = 0	612608	612608	0%	26.67	623.34	23.37	474291	506844	6.40%	284358	290003	1.94%
	Vi = 1	519318	546558	5%	46.348	688.76	14.86	527666	518432	-1.70%	285625	296435	3.60%
	Vi = 2	430085	455192	5.50%	46.426	791.2	17.04	556681	582631	4.40%	285553	297003	3.80%
	Vi = 3	416473	427239	2.50%	46.659	838.89	17.98	560348	570106	1.70%	285533	302007	5.40%
	Vi = 4	397910	403606	1.40%	46.568	902.38	19.38	573125	577876	0.80%	295131	290944	-1.40%
Average Improvement				1%			18.53			2.3%			2.7%

In Table 4.3 and Table 4.4, we see the difference of wire length between our approach and [2] is decreasing to about 8.5%. This is because as the circuit size growing larger, nets of this circuit would be much more, and the net cuts caused by partitioning would fewer while in the partitioning step we still take the net formed by voltage level into consideration. As a result, we are able to anticipate that our approach could generate a floorplan with less wire length and better power saving in the larger circuit size test case. We would show the experimental results on circuit with 200 modules and 300 modules in Table 4.5 and Table 4.6.

In Table 4.5 and Table 4.6, the run time of annealing based floorplanner is averaged about 18 times longer than the run time of our approach. In the mean time, the power saving of our approach is mostly more than [2], and the wire length is averaged better than [2] as we expected. This proofed that our method is more suit for larger scale designs.

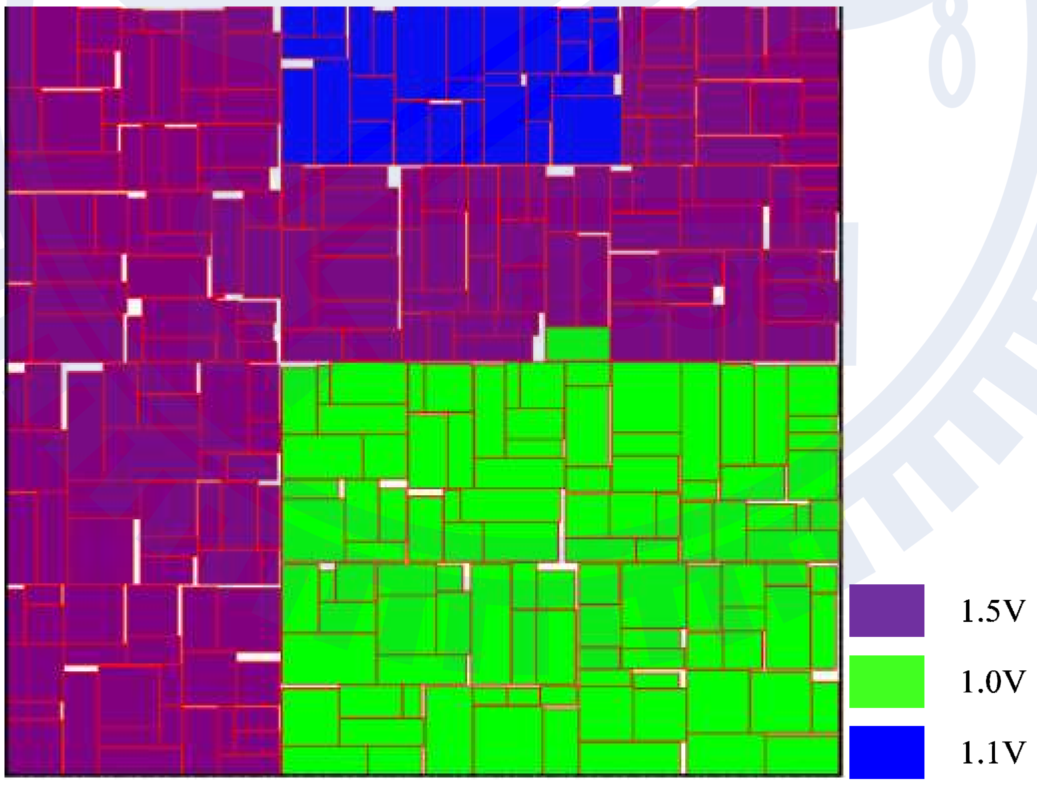


Fig. 4.1 The Resultant Floorplan for n300

Chapter 5

Conclusion and Future Work

In this thesis, we have proposed a fast, high-quality and non-stochastic approach for the floorplanning problem with simultaneous voltage assignment and island partitioning. The three factors area, wire length and power consumption of the resultant floorplan are taken into consideration while picking the resultant floorplan. The experiment results have shown that we are able to achieve a great power saving for the testing data sets while the number of voltage islands is restricted. Also, as the complexity of circuit is increasing, our approach is still able to handle the low power floorplanning with voltage islands problem efficiently. The decrease of execution time is significant while we still could keep other factors under a reasonable loss or even better than a simulated annealing-based approach for the same problem.

In the future, we will further improve the algorithm quality by refining the pruning strategies. The partition strategies need more constraints to balance the weight between wire length and power cost. The voltage islands should be placed close to the power pins in order to minimize the power routing complexity and the IR drop. Therefore, we are also considering the relationship among blocks, nets, and I/O pads to handle the IR drop problem.

Bibliography

[1] P. Coussy, A. Baganne, and E. Martin, "A Design Methodology For Integrating IP Into SOC Systems," Proceedings of the IEEE, 2002, pp. 307-310.

[2] Q. Ma and E. F. Y. Young, "Voltage Island-Driven Floorplanning," Proceedings of International Conference on Computer-Aided Design, 2007, pp. 644-648.

[3] R. H. J. M. Otten, "Automatic Floorplan Design," Proceedings of Design Automation Conference,1982, pp. 261-267.

[4] W. P. Lee, H. Y. Liu, and Y. W. Chang, "Voltage Island Aware Floorplanning for Power and Timing Optimization," Proceedings of International Conference on Computer-Aided Design, 2006, pp.389-394.

[5] Q. Ma and E. F. Y. Young, "Network Flow-based Power Optimization under Timing Constraints in MSV-driven Floorplanning," Proceedings of International Conference on Computer-Aided Design, 2008, pp1-8.

[6] W. K. Mak and J. W. Chen, "Voltage Island Generation under Performance Requirement for SoC Designs," Proceedings of Asia and South Pacific Design Automation Conference, 2007, pp.798-803.

[7] A. Vorg, M Radetzki, and W. Rosenstiel, "Measurement of IP qualification costs and benefits," Proceedings of Design Automation and Test in Europe Conference and Exhibition, 2004, pp. 996-1001.

[8] G. Karypis and V. Kumar, "Multilevel K-way Hypergraph Partitioning," Proceedings of Design Automation Conference, 1999, pp. 343–348.

[9] J. Z. Yan and C. Chu, "DeFer: Deferred Decision Making Enabled Fixed-Outline Floorplanner," Proceedings of Design Automation Conference, 2008, pp161-166.

[10] R. H. J. M. Otten, ”Efficient Floorplan Optimization, “ Proceedings of International Conference on Computer Design, 1983, pp. 499–502.

[11] L. Stockmeyer, ”Optimal orientations of cells in slicing floorplan designs, “ Information and Control , 1983, vol. 57, pp. 91–101.