# Low Power Floorplanning/Placement Methodology Considering Performance Constraints and Voltage Island Generation

Prepared by Ming-Ching Lu

Directed by Prof. Hung-Ming Chen

In Partial Fulfillment of the Requirements

for the Degree of
Master of Science

Department of Electronics Engineering

National Chiao Tung University

Hsinchu, Taiwan 300, R.O.C.

E-mail: benjamin.ee88@nctu.edu.tw

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Due to the growth in enormous design complexity with continuous scaling-down of technology, circuit sizes are getting much larger than before. To cope with the increasing design complexity, hierarchical design and reuse IP (Intellectual Property) modules are widely used [3] [15]. An efficient and effective floorplanning/placement approach is needed in order to improve solution quality and reduce runtime for the circuit design. Since the early stage of design will determine the overall chip performance. (Floorplanning/placement is to decide the positions of circuit blocks or IP cores on a chip subject to various objectives.)

Meanwhile, increased circuit density and performance compel the need to reduce power consumption that increases significantly as designers strive to utilize the advancing silicon capabilities. The consumer product market further drives the need to minimize chip power consumption [7].

Moreover, the widening requirements of special design applications force we to consider many more things than ever in floorplanning/placement stage. Among those requirements we care in very deep sub micron era (DSM), power dissipation and performance demands are very critical. Energy consumption is recognized as one of the most important parameters in designing modern portable electronic systems and wireless electronic systems, low power consideration has become essential in very large scale integration (VLSI) circuit and system designs today [11]. In CMOS digital circuits, power dissipation mainly consists of dynamic and static components. Dynamic power and static power both have direct relationship with supply voltage $Vdd$ (dynamic power is proportional to $Vdd^2$ and static power is proportional to $Vdd$), applying as low $Vdd$ as possible under the performance requirements is obviously the most effective way to reduce power consumption. There are many methods discussed before to reduce leakage power or dynamic power, such as multithreshold-voltage CMOS (MTCMOS) circuit technology [8], body biasing dual-threshold method [16] and multiple supply and threshold voltages methodology [4] [12].

One of the techniques to reduce power consumption is "Voltage Island" methodology, as the scale of process technologies shrinks, more and more devices can be implemented on a single chip. This enables various applications to be realized as SoC designs by using pre-designed cores. A voltage island is a group of on-chip cores powered by the same voltage source, independently from the chip-level voltage supply. This is a concept that the use of voltage islands permits operating different portions of the design at different supply voltage levels. It can achieve power saving and this technique becomes more and more popular [1] [6] [10].

Until recently, still few floorplanning/placement are concentrating on voltage islands consideration. Even so, to form good voltage islands property to save power and solve the critical delay problems (by performance constraints [14] [17]) to maintain good connecting between cores and reduce area overhead penalty is really a important subject.

## 1.1 Our Contributions

In SoC design, many cores of different functions are integrated together. If those individual cores have power characteristics unique of its own (each core can be applied its supply voltage arbitrarily) from the rest parts of the design and can be optimized accordingly, we can therefore achieve the best power solution (the lowest power consumption), but instead of sacrificing the floorplanning/placement like cobwebs, even worse the power routing complexity will become very large. To solve this problem, we propose a method in this thesis which can group these cores (blocks) to form islands that are in the same power supply domain and save power by applying lowest available supply voltages possible, and simultaneously consider the trade-off between power and the area/wirelength costs. Our method can be extended to consider the hierarchical voltage islands application due to the reason that almost every voltage islands consideration designs need to cluster cores using the same power supply. Moreover, to address particular design requirements (there may be high-speed transitions demand or critical paths between some appointed or particular cores for example), we can impose corresponding position constraints for the circuit cores in a floorplanning/placement.

In this thesis, we handle the floorplanning/placement with voltage islands and performance constraints using the B*-Tree representation. The experimental results show that our method performs well. According to the circuit power table information, our method utilizes the idea of location constraint (LC relation)[1] to constrain the nodes relationship between each pair of nodes which exist the parent-child relationship and group a set of cores using the same supply voltage by clustering their corresponding nodes in the B*-Tree to form an island-like distribution (we call it voltage island in the remainder of this thesis for short) to save more power and

[1]Location constraint is used to ensure that one module abuts against another one which is proposed in [2].

reduce power routing cost. We first explore a *feasible* floorplan/placement[2], and then apply our algorithm that can cluster blocks in the same supply voltage and place blocks applying performance constraints during each iteration. Finally we can generate a nice floorplan/placement for voltage islands and performance constraints. Experimental results based on the MCNC benchmark with the corresponding power tables and constraints show that our method definitely can meet this kind of requirements.

The remainder of this thesis is organized as follows. In Chapter 2, we briefly review B*-tree representation [2]. We also discuss the methodology of voltage islands [10], performance constraints [17] and give the problem formulation. In Chapter 3, we discuss our method for dealing with voltage islands and performance constraints, the detailed placement procedure and our algorithm present the detail design flow. Our experimental results are presented in chapter 4. Finally we give the conclusion of this thesis and future work in Chapter 5.

[2]A feasible floorplan/placement is a floorplan/placement that no any two modules overlap each other.

# Chapter 2

# Preliminaries

In this chapter, we briefly review the B*-Tree representation, concepts of voltage islands, and performance constraints in floorplanning. Then we formulate the lower power floorplanning/placement methodology considering performance constraints and voltage island generation.

## 2.1 Review of the B*-Tree Representation

A B*-Tree [2] is an ordered binary tree whose root corresponds to the module on the bottom-left corner for modeling a nonslicing floorplan. Given a compacted placement $P$ that can neither move down nor move left (called an *admissible placement*), we can construct a unique B*-tree in linear time. we can represent it by a unique B*-Tree $T$. Further, given a B*-tree, we can also obtain an admissible placement by packing the blocks in linear time with a contour structure [5].

In a B*-Tree $T$, the root of $T$ represents the block on the bottom-left corner, the $x$- and $y$-coordinates of the block associated with the root $(x_{root},y_{root})$ = (0,0). If node $n_j$ is the left child of node $n_i$, block $b_j$ is placed on the right-hand side and adjacent to block $b_i$ in the admissible placement; i.e., $x_j$=$x_i$+$w_i$, $w_i$ is the width of $b_i$. Otherwise, if node $n_j$ is the right child of $n_i$, block $b_j$ is placed above block $b_i$, with the $x$-coordinate of $b_j$ equal to that of $b_i$; i.e., $x_j$=$x_i$. With the contour

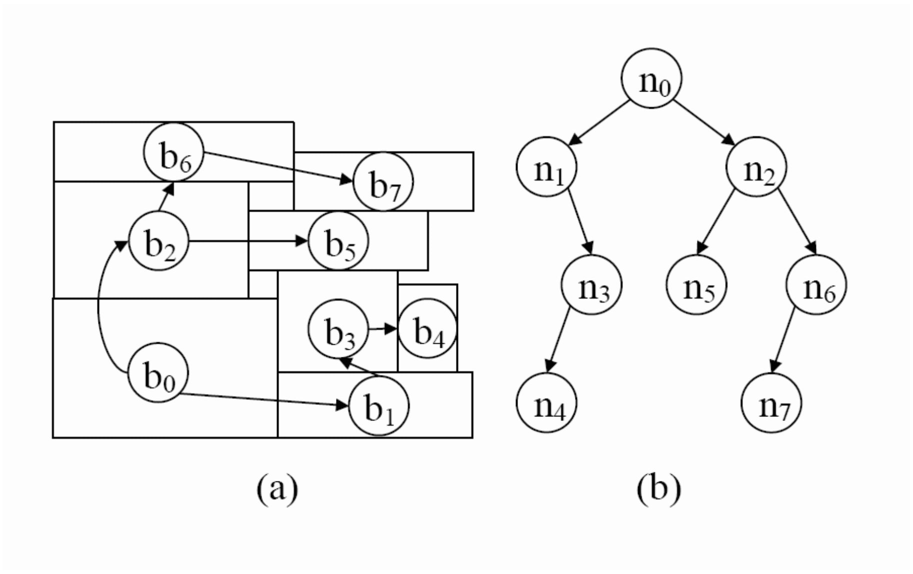


Figure 2.1: (a) An admissible placement. (b) The B*-tree representing the placement [17].

structure, we can compute the $y$-coordinate of a block in constant time.

Figure 2.1 illustrates an admissible placement (a) and its corresponding B*-Tree (b). Using the depth-first search (DFS) procedure, the B*-Tree $T$ for an admissible placement $P$ can be constructed in a recursive fashion. We first pick $n_0$, the root of $T$, and place $b_0$ on the bottom-left corner. Then we traverse the left child of $n_0$, $n_1$. Block $b_1$ is placed on the right of $b_0$. Therefore, since $n_1$ does not have a left child, we traverse $n_3$, the right child of $n_1$. The process continues until all nodes are traversed, and finally we will have an admissible placement. Inheriting from the nice properties of ordered binary trees, the B*-Trees is simple, efficient, effective, and flexible and particularly suitable for representing a nonslicing floorplan with various types of modules and for creating or incrementally updating a floorplan.

## 2.2 Voltage Islands Generation Methodology

Voltage Island is a system architecture and chip implementation methodology which can be used to reduce power consumption for SoC designs [10]. Active (dynamic) power is the power consumed by the intended work of the circuit to switch states and thus execute logic functions. We use the following equation to evaluate dynamic power: $P_{active} = C * \mathrm{f} * Vdd^2$, where $C$ is the charging/discharging of the capacitance of the switching nodes. Even if we assume perfect scaling of the capacitance per unit area and supply voltage $Vdd$, frequency of operation has increased at a faster rate than the scaling of the silicon process technology, and led to an increase in power density. In addition to active power, smaller geometries make leakage power become as important as active power in many applications. Device structures are improved for lower transistor oxide thickness ($T_{ox}$) to better transistor performance. To maintain the reliability, $Vdd$ must be lowered as $T_{ox}$ is reduced; and transistor threshold voltage ($V_t$) must be reduced as well in order to maintain performance. The decrease in $V_t$ and $T_{ox}$ drives significant increases in leakage power.

The combination of increasing active power density and leakage currents has created a power management problem in the semiconductor industry. One scenario shown in Figure 2.2 is an example of "Voltage Islands" that optimizes the individual voltage to reduce active power, achieving required performance. Mostly performance-critical element of the design requires the highest voltage level supported to maximize the performance, while other function cores coexist on the SoC may not need this level of voltage, so they can be run at lower voltages to save significant active power. Furthermore, some functions cores, like embedded analog cores, are specified at specific voltages, and can be easily accommodated in mixed voltage systems.

Introducing voltage islands concept makes the chip design process even more complicated with respect to static timing and power routing. In particular, the complexity grows significantly with the number of islands. Designs using voltage islands needs to group the cores powered by the same voltage source and not to violate other design metrics such as timing and wire congestion. Meanwhile, the

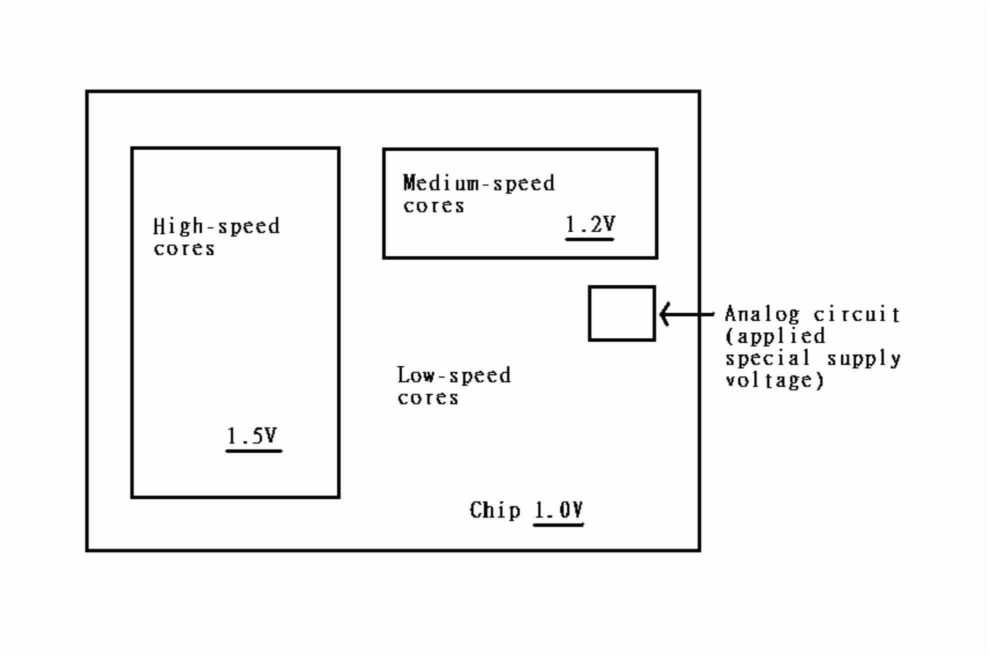


Figure 2.2: Timing critical Voltage Island methodology. Each of the elements are identified by their minimum required voltage, and the most performance-critical elements are placed together and supplied by highest voltage to maximize its performance.

number of voltage islands should be appropriate (not too many) considering signal translation and communication between different islands, which requires level converters. We also need to consider power routing complexity [6]. Therefore the overhead for applying voltage islands methodology with respect to area and delay is unavoidable.

Figure 2.3 is a voltage island powering and switching control graph from [10]. The whole voltage island are powered from the island voltage, $VDDI$ (VDD-inside), while the circuits are powered from a supply voltage called $VDDO$ (VDD-outside). When there are signals traveling between inside and outside of the island, we need level converters to translate signals' level in different supply voltages. It is the price we need to pay when using different supply voltages in a single chip. If we apply the corresponding lowest supply voltage to each of the functional blocks in traditional floorplanning/placement strategy and not using voltage island methodology,

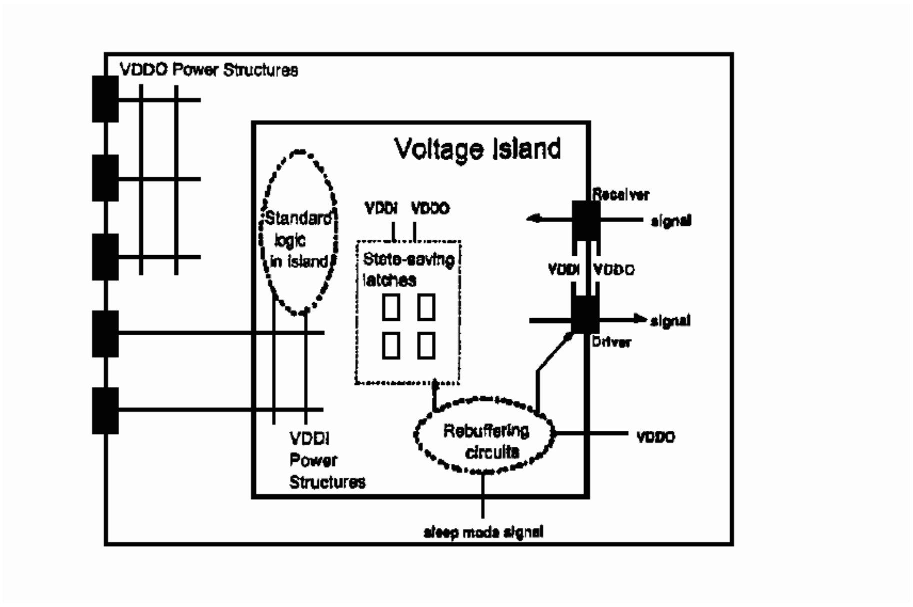


Figure 2.3: Voltage island powering and switching control graph from [10]. The voltage island is powered from the island voltage ($VDDI$), while the whole chip is powered from a supply voltage ($VDDO$). There are receivers and drivers to handle converting the voltage level of the signal between inside and outside of the island.

we will get a expensive cost in power routing complexity and area overhead in level converters.

## 2.3 During Floorplanning/Placement Performance Constraints Consideration

Performance constraints is a consideration for the interconnect delay domination in the circuit performance for DSM VLSI design. Minimizing total wire length, as traditional floorplanners/placers did, can not guarantee bounded delay for critical nets. It is desirable to minimize the critical net delay by tightening them together to optimize performance or to meet the delay constraints by placing them close enough to each other. To be exact, the constraint requires designated nets (blocks/cores) to be placed within a pre-defined bounding box nets. Imposing the performance constraint, we can optimize not only the circuit delay, but also the total wire length.

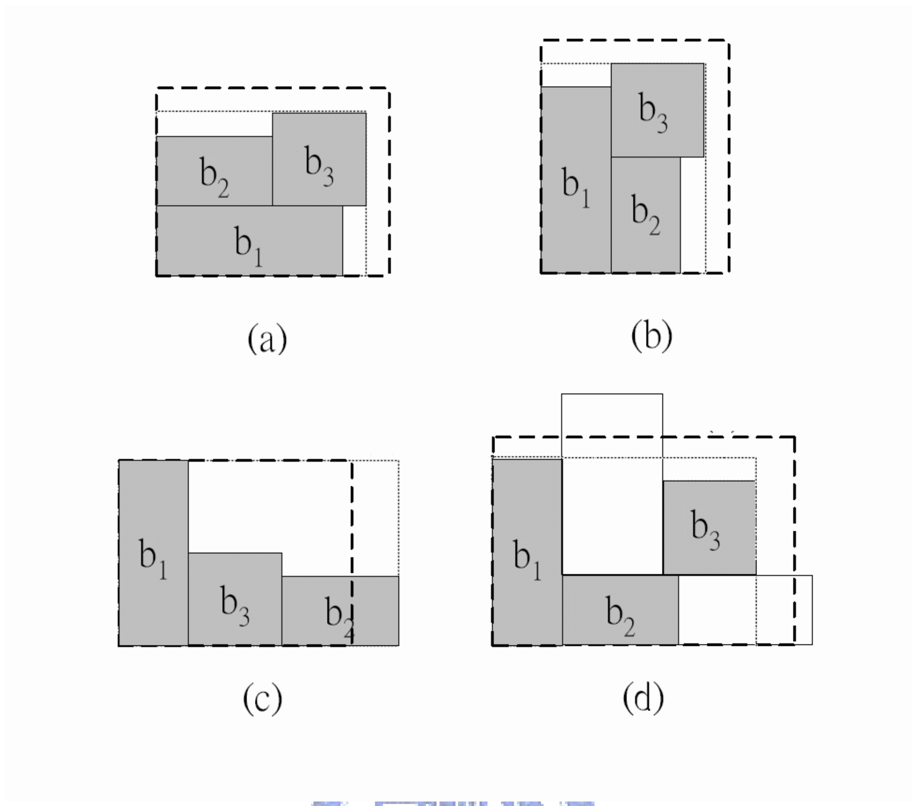


Figure 2.4: (a)(b) are both feasible placements of performance blocks b1, b2 and b3. The rectangle of the dotted line is the bounding box of the blocks. The rectangle of the dash line is the bounding box with the maximum delay. (c) An infeasible placement with the blocks falling out of the bounding box associated with the delay constraint. Note that (a)(b)(c) are applying the same performance constraints. If the performance constraints are not so tight, (d) is a feasible placement as well.

Figure 2.4 is an example to explain the performance bounding box. We constrain the blocks' distance between each other by using a bounding box that the summation of its' height and width is bounded, therefore their maximum delay is bounded.

## 2.4 Problem Formulation

The problem we concern about is described as follow: Let $B$=$\{b_1$,$b_2$,...,$b_n\}$ be a set of $n$ rectangular modules whose width, height, and area are denoted by $W_i$, $H_i$, and $A_i$, 1≦$i$≦$n$. Let ($x_i$,$y_i$) denote the coordinates of the bottom-left corner of module $b_i$, 1≦$i$≦$n$, on a chip. Each module is associated with a power table, a list of matching pair that store (supply voltage, power dissipaiton). A floorplan/placement $P$ considering the performance constraint and voltage islands generation is that according to the given power table information, an assignment of $(x_i,y_i)$ for each $b_i$, $1 \leqq i \leqq n$, such that cores clustered using the same voltage to form appropriate number of islands achieving low power consumption, while no two modules overlap and the given performance constraints are satisfied. The goal of floorplanning/placement is to optimize a specified cost metric, including the area $A_{tot}$ induced by the assignment of $b_i$s on the chip or the arbitrary combination of the area, wirelength, power dissipation and the number of voltage islands.

# Chapter 3

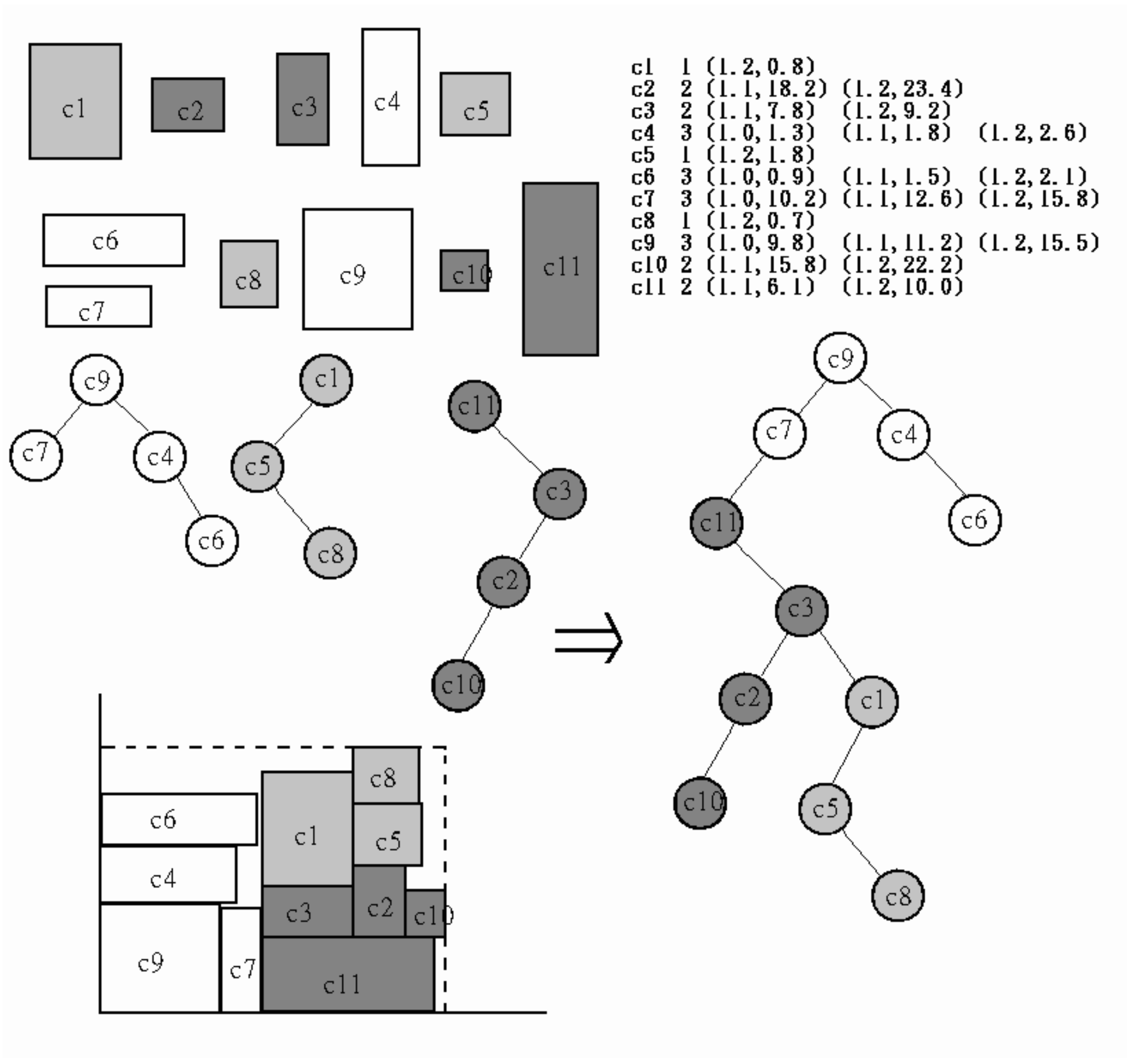
# Algorithm for Simultaneously Considering Voltage Islands Generation and Performance Constraints

In this chapter, we first present the solutions for voltage islands generation with B*-Tree representation. Then, we propose the feasible conditions for B*-Tree with the performance constraints.

## 3.1 Floorplanning/Placement for Voltage Islands

We first give an example to show the setup in creating voltage islands in SoCs using B*-Tree. In Figure 3.1, each core is followed by a number identified the number of its usable voltages, then associated with a *power table*; it is a list of matching pairs that store (supply voltage, power dissipation) which specifies the legal voltage levels that can be used to work functionally and the corresponding average power dissipation values. For instance, the block $c_4$ can operate at 1.0, 1.1 or $1.2V$, and its corresponding power consumption are $1.3mW$, $1.8mW$ and $2.6mW$.

Because $1.2V$ is the highest supply voltage of all the eleven blocks, the chip-level voltage is assumed to be $1.2V$ (the highest voltage). If we do not need to create



|  |  |  |  |  |
|---|---|---|---|---|
| c1 | 1 | (1.2, 0.8) |  |  |
| c2 | 2 | (1.1, 18.2) | (1.2, 23.4) |  |
| c3 | 2 | (1.1, 7.8) | (1.2, 9.2) |  |
| c4 | 3 | (1.0, 1.3) | (1.1, 1.8) | (1.2, 2.6) |
| c5 | 1 | (1.2, 1.8) |  |  |
| c6 | 3 | (1.0, 0.9) | (1.1, 1.5) | (1.2, 2.1) |
| c7 | 3 | (1.0, 10.2) | (1.1, 12.6) | (1.2, 15.8) |
| c8 | 1 | (1.2, 0.7) |  |  |
| c9 | 3 | (1.0, 9.8) | (1.1, 11.2) | (1.2, 15.5) |
| c10 | 2 | (1.1, 15.8) | (1.2, 22.2) |  |
| c11 | 2 | (1.1, 6.1) | (1.2, 10.0) |  |

Figure 3.1: A illustration of a naive approach to generate voltage islands in chip-level design. If we want the maximized power saving, one obvious way is to operate each block at its lowest available voltage. Firstly, we partition the blocks by their lowest supply voltage, we construct the subtrees of those compatible blocks, then we build the B*-Tree like the final tree shown in the graph, and the corresponding placement is shown beside the B*-Tree.

any voltage islands, all the blocks will use their highest voltage as the traditional floorplanning/placement did. However if we want to minimize the power consumption for whole chip, one obvious way is to operate each block at its lowest voltage, which means that we need at least 3 voltage islands: one for $\{c1, c5, c8\}$, one for $\{c2, c3, c10, c11\}$, and one for $\{c4, c6, c7, c9\}$. It is obviously not a perfect solution because even the power saving is maximized, the price of area/wirelength overhead may be very high. Sometimes we may be forced to use more islands, or to switch the supply voltages of some blocks which support two or more legal supply voltages to one of its higher legal voltages to alleviate the problems.

Our method to create the voltage islands is to constrain the nodes relationship between each pair of nodes which exist the parent-child relationship in the B*-Tree representation. In other words, to attain voltage islands requirement, we hope the blocks which use the same supply voltage (say *compatible*) can be clustered together, so we group those blocks to be a subtree in corresponding B*-Tree.

However due to the disadvantage of the B*-Tree representation, not any two nodes abut in the tree always means that the corresponding two blocks abut. Similarly, nodes are not in the same subtree do not mean they do not abut physically. We give an example in Figure 3.2. Hence we observe that not all the blocks constructed in a subtree will be put together to form a voltage island, so we have no necessary to place a B*-Tree with perfect subtrees. What we want to do is to increase the probability those nodes can be clustered together, then apply a simple checking method to inspect if they really form a favorable island.

During each iteration in the annealing process, area optimizing makes the modules getting closer to each other (the dead space is getting smaller), character of mapping a subtree to a voltage island will be more apparent. The reason is that because small dead space means that blocks packing is thick, therefore the possibility of contact between neighboring blocks is increasing. Then, the conditions like

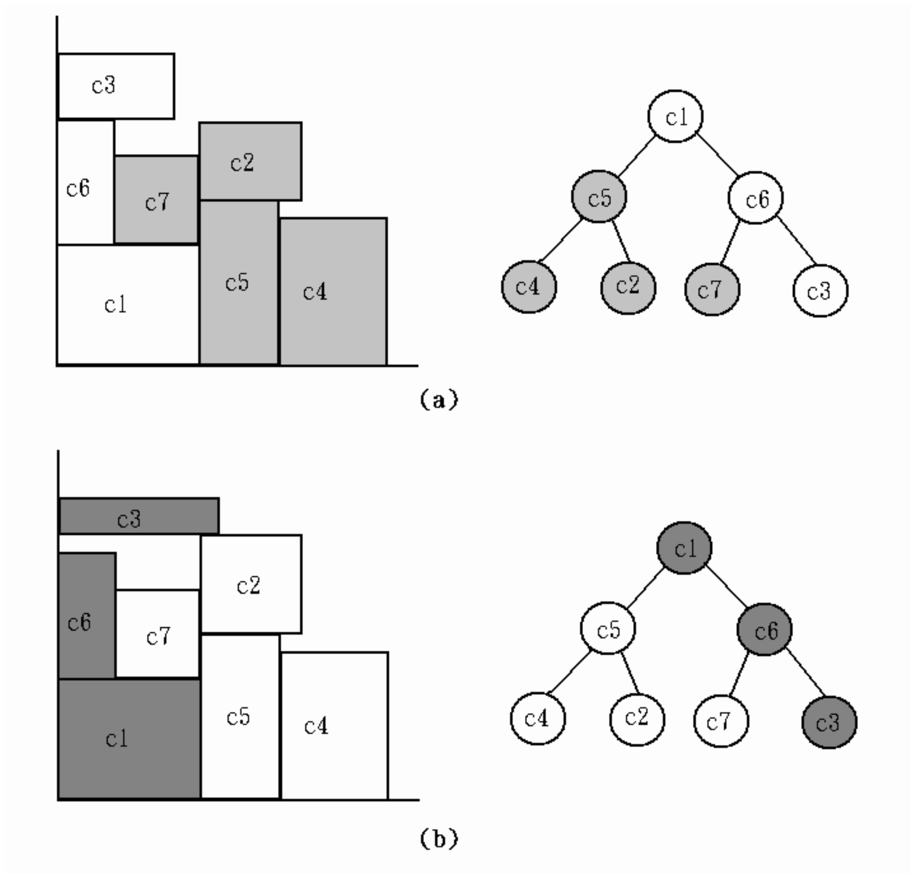


Figure 3.2: Example to explain the condition that not any two nodes abut always represent the situation that the corresponding two blocks abut; and nodes are not in the same subtree do not mean they do not abut physically. In(a), node $c7$ is not in the same subtree with $\{c2, c4, c5\}$; but in the floorplanning/placement, block $c2$, $c5$ and $c7$ are connected. In(b), nodes $\{c1, c3, c6\}$ form a subtree in the B*-Tree; but in floorplanning/placement, block $c3$ is not connected with block $c6$. There are extra overhead in this voltage island.

Figure 3.2(a) will increase, and the conditions like Figure 3.2(b) will decrease.

We develop an idea to generate voltage islands in B*-Tree: When the area cost is getting lower and the dead space of the total area is becoming smaller, there will be a visible mapping relationship that if $n_j$ is the left child (or right child) of $n_i$ in the B*-Tree representation, then the block $b_j$ right (or left) abuts to block $b_i$. So much so that the probability a node adds to a compatible subtree[1] and the subtree grows and mapping to a favorable voltage island shape will be increased. As the usage of the concept, our approach to attain to the intent is to randomly choose two nodes $n$, $p$ in the tree, $V(n)$ and $V(p)$ denote the adopted voltages of node $n$ and node $p$. If the following three situations occur, we change the positions of these two nodes.

- $V(p) = V(n)$: Node $p$ and node $n$ are compatible . This operation let compatible blocks in the subtree or between subtrees can exchange their positions.
- $V(p.parent) = V(n)$: Node $p$'s parent and node $n$ are compatible. This operation allows exchanging two nodes to form a good subtree easily.
- $V(p.leftchild) = V(n)$ and $V(p.rightchild) = V(n)$: Node $p$'s left child and right child both has the same voltage with node $n$. This operation allows exchanging two nodes to form a good subtree easily as well.

Except the three operations considered to add for perturbation during simulated annealing, we modify two operations when constructing a B*-Tree.

- Delete_Node: If we want to delete node $n$, we raise the child node ($n.leftchild$ or $n.rightchild$) which is compatible with node $n$'s parent. If the children are in the same situation (both compatible or both not compatible), then we randomly choose one of them.

[1]A compatible subtree is a subtree which nodes inside are all in the same voltage with the node.

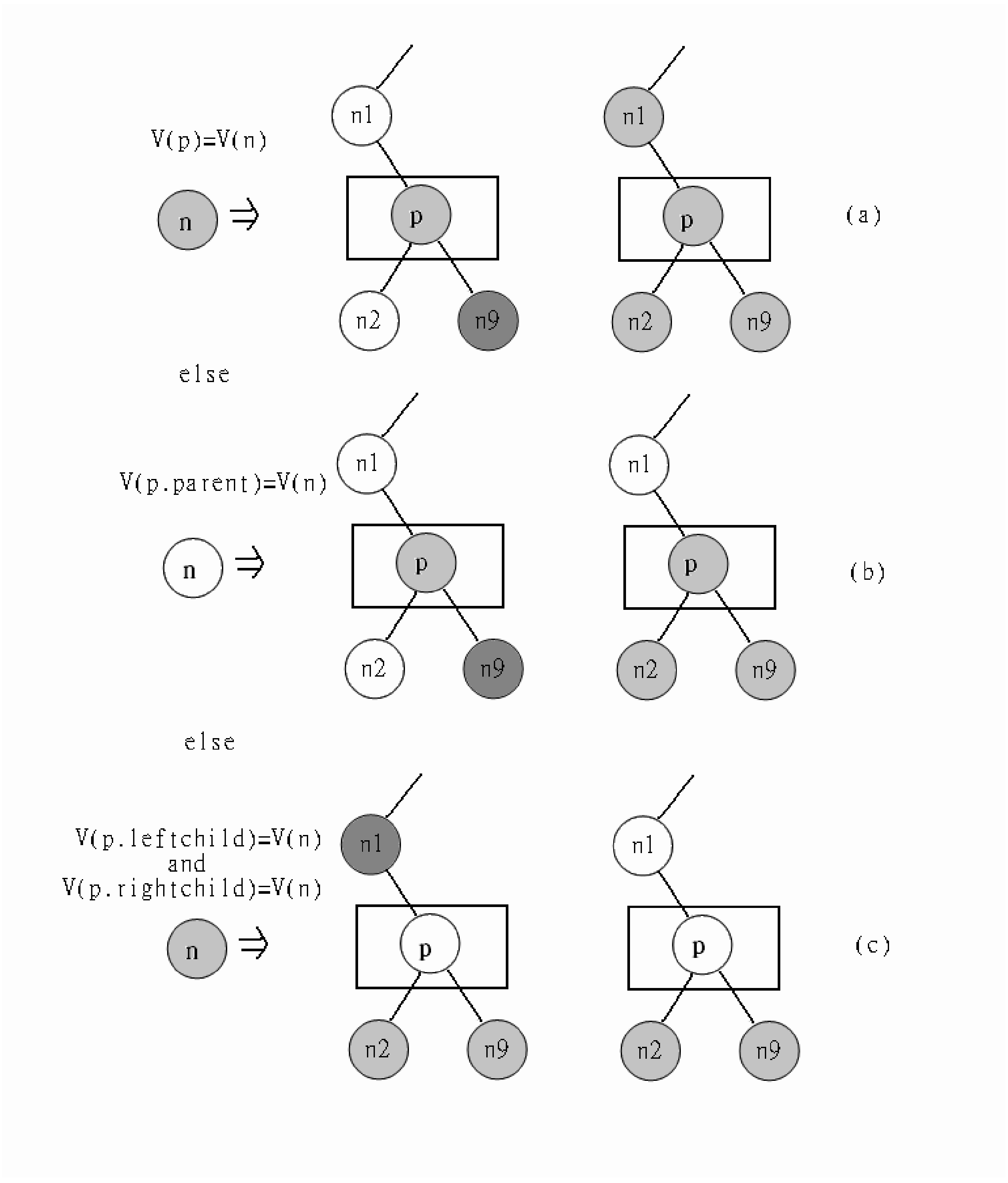


Figure 3.3: Three added operations to increase the probability of clustering. (a) Supply voltages of node $n$ and node $p$ are compatible, $V(p) = V(n)$. We can maintain the subtree property whose root is node $p$, even if there is not a subtree, we just exchange the two nodes in the same supply voltage. No good voltage island property will be ravaged. (b) Allow the two nodes $p$, $n$, $V(p.parent) = V(n)$, exchanging to let $n$ be the leaf of the subtree or to connect two compatible subtrees to a larger subtree. (c) Allow the two nodes $p$, $n$, $V(p.leftchild) = V(n)$ and $V(p.rightchild) = V(n)$, exchanging to let $n$ be the root of the subtree or to connect two compatible subtrees as well.

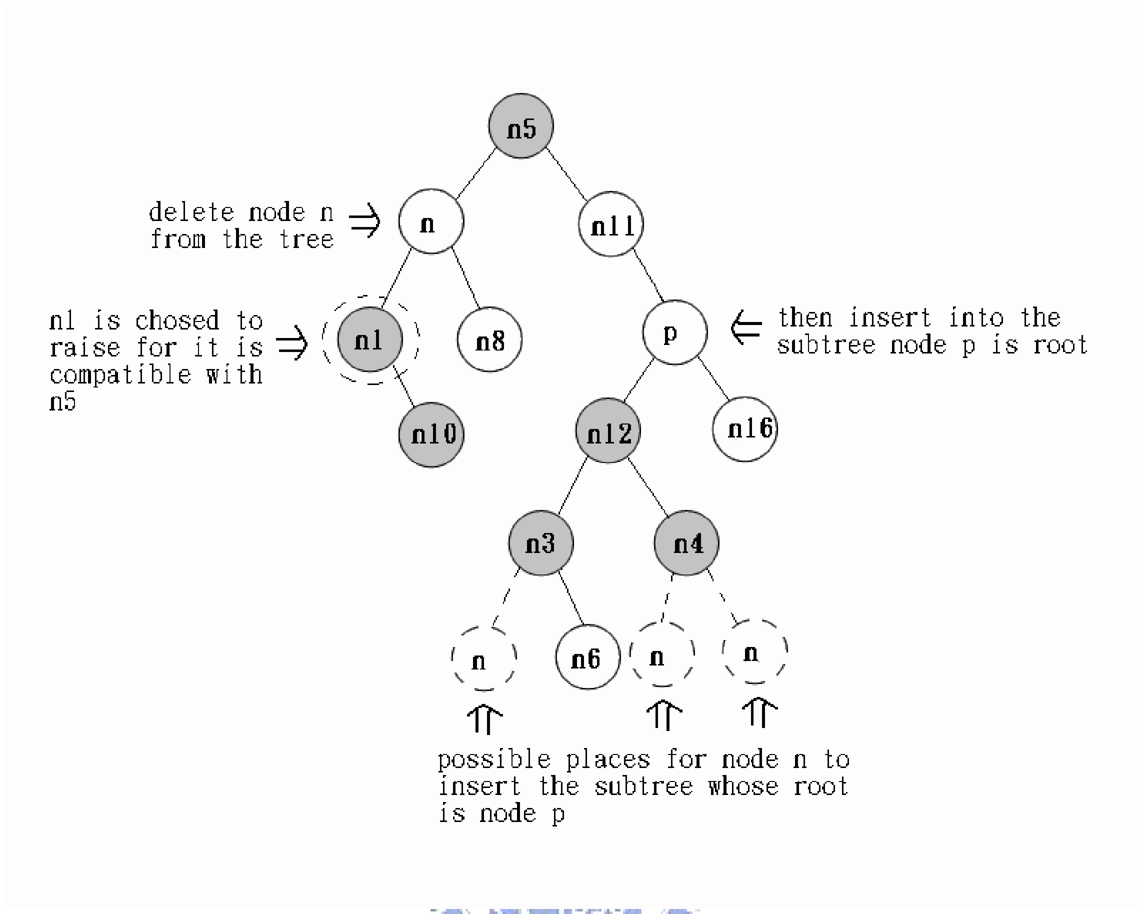

n5
delete node n from the tree
n
n11
n1 is chosed to raise for it is compatible with n5
n1
n8
p
then insert into the subtree node p is root
n10
n12
n16
n3
n4
n
n6
n
n
possible places for node n to insert the subtree whose root is node p

Figure 3.4: An illustration of the two modified constructing operations. Node $n$ is to be deleted from B*-Tree and inserted into the subtree which the root is node $p$. Because node $n1$ and $n5$ is compatible, we raise $n1$ to replace the place of node $n$. In the subtree whose root is node $p$, and $n3$, $n4$ and $n12$ are compatible nodes, we can see from the graph, there are three possible places for node $n$ to join the cluster of subtree in the same voltage.

- Insert_Node: If node $n$ is to be inserted into the subtree whose root is node $p$ and there exists compatible nodes, it will be placed to join the cluster of the compatible nodes. We insert node $n$ to abut a node in the subtree that has the same voltage to cluster them as possible. If there does not exist any node compatible, then we randomly choose one place to insert into as before.

The reason why we need to remodel the basic procedures in constructing the tree from before is that former floorplanners/placers only handle "Position Constraints".

Position constraints are concerned on the changes of blocks position, either precise position, such as pre-placed problems, or the relative position, i.e. alignment or performance constraints problems. However, for preserving voltage island property, each node in different supply voltage is in different condition. Nodes in different supply voltages are ranked to different orders when needed to connect with other nodes or delete from the B*-Tree.

The subtree construction is just a method to increase the trend that has higher possibility constructing a good voltage island property, so we need a property checking function to check if there is a favorable voltage island shape. We do it after the contour updated to make sure the voltage island property is acceptable.

Here we introduce how we perform the voltage island property checking.

A horizontal contour is used in implementing a horizontal B*-Tree representation to construct corresponding placement, it can be used to reduce the runtime for finding the $y$-coordinate of a newly inserted module. Without the contour, the runtime for placing a new module is linear to the number of blocks. By preserving the advantage of runtime, we check voltage island property right after the updating of the horizontal contour when inserting a new block. When a new block is inserted, a new contour will replace the current contour, and there must be at least one block contact with the new block. For the example in Figure 3.5, $b_5$ is a newly added block, it induces a new contour by replacing a segment of the odd contour which was contributed by blocks $b_3$, $b_4$ and $b_0$, and because of character of B*-Tree to represent a placement (blocks can not be moved down nor left), so block $b_5$ must has connection with one of the three blocks. In Figure 3.5, block $b_5$ has contact with block $b_3$, then we can observe the voltage island property by checking if they are compatible to each other. They will be in the same voltage island if they are compatible, or they will be in different voltage islands. With the addition of the contour record, we can further observe the shape of the island, and preserve a

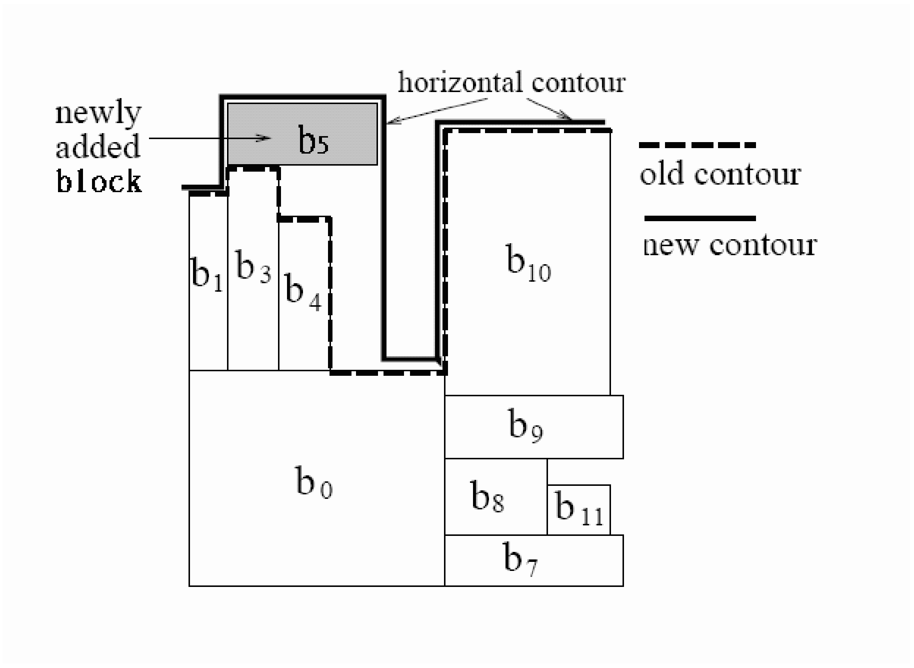


Figure 3.5: An illustration of voltage island property checking method. $b_5$ is a newly added block which replaces a segment of contour contributed from $b_3$, $b_4$ and $b_0$, so we check the connection between the 3 blocks with block $b_5$, not all of the blocks inside the placement.

favorable shape in each iteration of simulated annealing if we want. Runtime will be reduced a lot for we only check blocks on the segment of contour line replaced by the new added block, not all of the blocks the voltage island property one by one. Once we find a block contacts with new added block, we do not check other blocks anymore, because their voltage island property had be constructed already.

## 3.2 Floorplanning/Placement with Performance Constraints Blocks

Traditional floorplanners/placers minimize total wirelength but they can not guarantee critical nets to meet bounded delay. This becomes more important because timing convergence is a big issue in DSM design. In order to make critical net delay meets delay constraint, there is a method proposed in [14] [17]. Because actual interconnect delay after appropriate buffer insertions will be close to linear in terms of distance. The concept of the method is to use the linear function in terms of distance to estimate delay. They assume there are a source at ($x_s$,$y_s$) and a sink at ($x_t$,$y_t$), their locations are the corner points as far as possible, and the delay of the net $D_{s,t}$,

$$D_{s,t} = \delta(\mid x_t - x_s \mid + \mid y_t - y_s \mid)$$

where $\delta$ is a constant to scale the distance to timing.

$D_{s,t}$ is the maximum distance between source and sink equal to the half perimeter of the bounding box of the two blocks. In [17], they use the delay model to do sub-placement (to place a set of feasible sub-placements for the performance blocks) by restricting those performance blocks the longest distance

$$(\mid x_t - x_s \mid + \mid y_t - y_s \mid) = D_{s,t}/\delta \leq D_{max}/\delta$$

$D_{max}$ is the given maximum delay for the distance from the source to the sink.

So given $k$ performance blocks whose areas are $A_i$, $i$=1,2,...,$k$, they can get some rectilinear super blocks that their width $W_{perf}$ and height $H_{perf}$ satisfy the performance constraint:

$$W_{perf} + H_{perf} = B \leq B_{max}$$

$B_{max}$ is the maximum bounded distance

Among the placements (rectilinear super blocks) meeting the performance constraint, they pick the one with the minimum dead space $S_{perf}$= $u$*$v$-$\sum A_i$ and fix the rectilinear block (and thus fix the delay) for further processing with other blocks. By using the pre-clustered shape-fixed appropriate rectilinear block, they guarantee that the performance constraint will be satisfied throughout the remaining processing.

There is a little difference in choosing an appropriate rectilinear super block in [14]. In [14], they use a method that adjust the $W_{perf}$, $H_{perf}$ dynamically into the rectilinear super blocks. At higher temperature in annealing process, let $W_{perf}$ and $H_{perf}$ to be half-half

$$W_{perf} = B_{max}/2$$

and

$$H_{perf} = B_{max}/2$$

Simulated annealing is characterized as chaotic process where a square range-box is appropriate to use for approximate guidance. And at lower temperature, a specific range box is almost fixed and can't be changed easily to exactly capture the meaning of delay bound.

Our method combines the advantages of the two methods above, using the B*-Tree representation to handle the floorplanning/placement problems and we keep the flexibility of the sub-placement for the performance constraint. We do not pick the minimum dead space sub-placement and fix the shape (or the relational position) of the performance blocks before processing with other blocks at the beginning. Instead, we let the performance blocks process with other blocks as if they are not under restriction, the total area and wirelength can be optimized better (this phenomenon will be obvious if much more sets of performance constraint blocks occur), the reason is that we do not tighten but just meet the constraints (so maybe the distance between the blocks can be not so close to each other), so there is more space flexibility to handle the overall placed area or wirelength.

When the temperature becomes low, we do not allow a solution that has infeasible performance constraints even if it has a better cost value, the best solution will be kept until next feasible solution with better cost.

## 3.3 Simultaneous Voltage Island Generation and Performance Constraints

More than that, the most important voltage islands property can be constructed better by it, for that supply voltages of the performance blocks are possibly different, if we tighten the shape at the beginning, we may be forced to arise the supply voltages of some of them to the higher one to meet voltage island property and we will lose power benefit; or we will get a disorder B*-Tree structure that the voltage property is withered.

## 3.4 Our Methodology

Our floorplanning/placement design algorithm is based on the simulated annealing method [9]. We only consider hard modules here. The flow of our algorithm is summarized in Figure 3.6. We perturb a B*-Tree to another by the following operations:

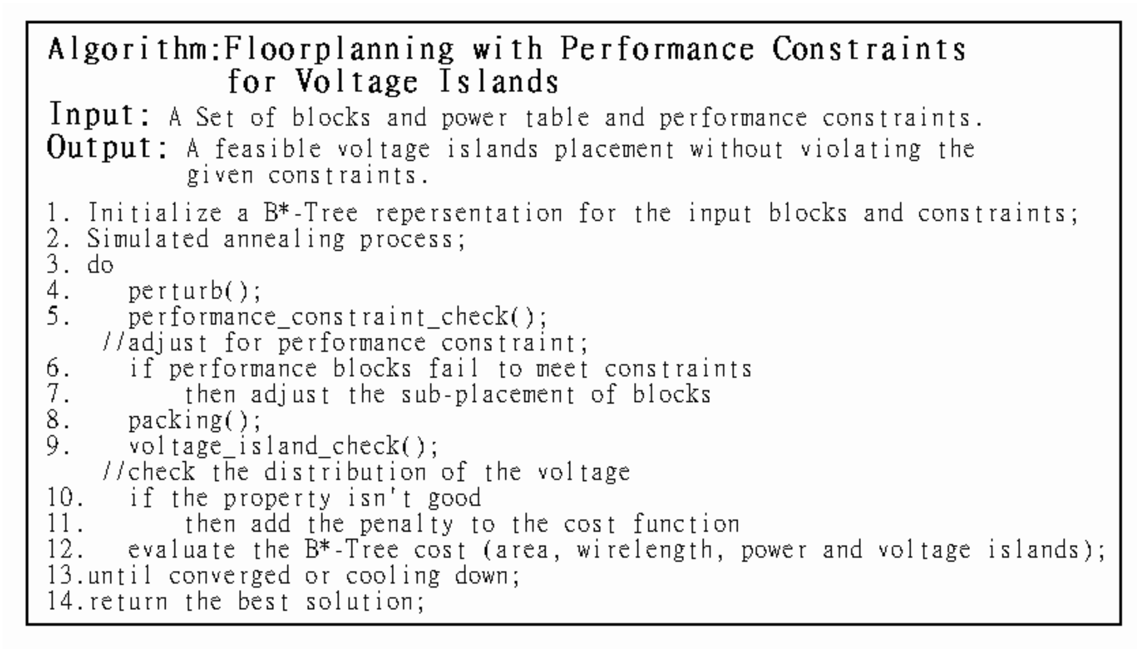
- *Op*1: Change the supply voltage of a block. (Except that only one supply voltage is available.)

- *Op*2: Rotate a block.

- *Op*3: Flip a block.

- *Op*4: Swap two blocks. (The three situations we discussed in Figure 3.3 will have higher probability to be allowed to do swap, while other situations that will wither the subtree property will have lower probability.)

- *Op*5: Move a block to another place. (It is pertinent with the two operations in Figure 3.4, because moving a block to another place need to delete it from the tree and then reinsert it into the tree.)



```
Algorithm:Floorplanning with Performance Constraints
          for Voltage Islands
Input: A Set of blocks and power table and performance constraints.
Output: A feasible voltage islands placement without violating the
        given constraints.
1. Initialize a B*-Tree repersentation for the input blocks and constraints;
2. Simulated annealing process;
3. do
4.    perturb();
5.    performance_constraint_check();
   //adjust for performance constraint;
6.    if performance blocks fail to meet constraints
7.        then adjust the sub-placement of blocks
8.    packing();
9.    voltage_island_check();
   //check the distribution of the voltage
10.   if the property isn't good
11.       then add the penalty to the cost function
12.   evaluate the B*-Tree cost (area, wirelength, power and voltage islands);
13.until converged or cooling down;
14.return the best solution;
```

Figure 3.6: The voltage islands and performance driven design flow.

The first one operation $Op1$ is designed for blocks' voltage perturbation. In $Op2$, we rotate a block. This action can be applied to any node without changing the relations between any two nodes except performance constraints blocks. Because each performance bounding box has a maximum bounded distance $B_{max}$, we should make sure not to violate the constraints. In $Op3$ , we flip a block. It can be applied to any blocks including performance constraints blocks, for we do not tight them together at the beginning. $Op4$ and $Op5$ change the relations of blocks to get a different placement and B*-Tree structure. The detail procedures of our method is discussed in Section 3.1.

# Chapter 4

# The Experimental Results

Our method can handle circuits that have two or three kinds of supply voltages, circuits with more than three supply voltages is also processible. If there is only one supply voltage overall the circuit, then our program will run like original B*-Tree simulated annealing method, and will not spend extra runtime to handle voltage island property. We implemented our algorithm in the C++ programming language on a PC with P4-2.4GHz cpu and 440MB memory. We experiment with our approach on MCNC circuit benchmark, and compare with [2], the original B*-Tree with simulated annealing method.

To compare the power routing complexity, and the overhead area due to level converters, we adopt a simple method to estimate the cost of it. Wire connections between two blocks in different islands always need level shifters to change its signal levels, we assume that all level converters are placed on the periphery of voltage islands (the boundary of the two islands), except for the boundary of the chip. The main reasons are that, firstly, we preserve a thin layered unit level converter area on the boundary between two different islands, and it is enough for the required area of all level converters for the wire connections; secondly, the power pins located on the outmost periphery (the boundary of the chip), so we do not place level converters there.

Moreover, although different types of level converters may need different sizes of area to implement, we simply assume all level converters are the same size. Based on above assumptions, we count the boundary length except the outmost side to be the power routing/overhead area cost. Table 4.1 show the experiment results, columns 1 and 2 give the name of the circuit (the number of blocks) and the power tables we use, and the power consumption in column 5 is lowest since we use the lowest available voltages for every block in it, and we compare the power routing/overhead converter area cost from [2], which are normalized to ours.

We experiment our method by testing with different power tables, which are comprised of (supply voltage, power dissipation) pairs. They are randomly assigned in order to simulate the fact that different functional cores may be different in their power density or supply voltage. Pt2 is the power table that contains 2 different supply voltages, while pt3, pt3_1, pt3_2 are power tables contains 3 folds. Figure 4.1 is a feasible placement[1] result graph of our voltage island generation method. Note that even we use pt3 (or pt3_1, pt3_2) to construct voltage island, if the area or wirelength requirement is tight, we may get the placement solution raising the lower voltage blocks to a higher supply voltage to meet the requirement (we may finally get a floorplanning/placement that are only 2 folds of voltage supplies even we use pt3 series power tables to be the power information, an illustration examples are shown in the Figure 4.2). Figure 4.3 is an example of a placement without voltage island generation methodology, it is a scrambled placement even the power consumption is the lowest.

From Table 4.1, we can see that our runtime is about 3 times to an original B*-Tree method. Although our power consumption is a little more than the lowest power listed in column 5 (because we assume they use lowest power), but our routing/level

[1]A feasible placement in B*-Tree representation with performance constraints is a placement that no any two blocks overlap each other while each block can not be move left or down and requirement of performance constraints is meet.

Table 4.1: The comparison between [2] and our approach on power consumption and power routing cost. The results are obtained from some MCNC benchmarks with different power tables. Area in $mm^2$, dead space(Dead) in%, power consumption(P) in $mW$ and runtime(T) in *seconds*. C is level converters area and routing cost.

| Circuit | Table | Original B*-Tree [2] | | | | | Ours | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Area | Dead | P | C | T | Area | Dead | P | C | T |
| hp | pt2 | 8.95 | 1.4% | 83.7 | 1.81 | 4 | 9.11 | 3.10% | 86.4 | 1 | 15 |
| | pt3 | | | 73.4 | 2.38 | | 9.10 | 2.98% | 78.3 | | 18 |
| ami33 | pt3 | 1.27 | 8.94% | 113.6 | 4.52 | 26 | 1.181 | 2.07% | 123.2 | | 89 |
| | pt3-1 | | | 131.1 | 4.76 | | 1.183 | 2.23% | 136.3 | | 89 |
| ami49 | pt2 | 36.8 | 3.68% | 147.1 | 4.18 | 53 | 36.67 | 3.34% | 151.5 | | 243 |
| | pt3 | | | 142 | 5.43 | | 36.68 | 3.38% | 156.2 | | 234 |
| | pt3-1 | | | 183.1 | 6.11 | | 36.75 | 3.52% | 196.4 | | 234 |
| | pt3-2 | | | 208 | 5.97 | | 36.78 | 3.64% | 222.9 | | 240 |

converters cost is about 16.4% - 55.2% compared with [2].

In order to compare our results with [2] and [17] in similar number of voltage islands and especially power routing cost, we come up with a heuristic that adjusts supply voltage of the original B*-Tree results: For a floorplan/placement, we raise firstly the block that is surrounded by the blocks applying the highest voltage (because a block that applies highest voltage means that its voltage can not be changed to other voltage levels) until all the blocks applying highest voltage are connected together to be a island. We fix the blocks that applying highest voltages, then take care of the blocks applying second high voltage, the same method is used until they form a island, too. Voltage islands generation is completed until all voltage level are considered.

Figure 4.4 is an initial floorplan/placement without voltage island generation methodology applying the lowest power, and Figure 4.5 is the placement result after applying the heuristic to raise voltages of some blocks, and 3 - 4 voltage islands with 3 kinds supply voltages is acceptable. Figure 4.6 is an example of raising all lowest voltage to form a two voltage islands floorplan/placement with 2 kinds of supply

Table 4.2: The comparisons between power amount that need to be raised to form a voltage island floorplan/placement. The number of voltage islands and power routing/level converters area cost are nearly the same for both approaches.

| Circuit | Table | Lowest | Ours | | Original B*-Tree [2] | |
|---|---|---|---|---|---|---|
| | | | Power | Percentage | Power | Percentage |
| hp | pt2 | 83.7 | 86.4 | 3.2% | 97.9 | 16.7% |
| | pt3 | 73.4 | 78.3 | 6.7% | 91.6 | 24.8% |
| ami33 | pt3 | 113.6 | 123.2 | 8.5% | 136.8 | 20.4% |
| | pt3-1 | 131.1 | 136.3 | 4% | 161.7 | 23.3% |
| ami49 | pt2 | 147.1 | 151.5 | 3% | 171.4 | 16.5% |
| | pt3 | 142 | 156.2 | 10% | 169.6 | 19.4% |
| | pt3-1 | 183.1 | 196.4 | 9.7% | 239.6 | 30.1% |
| | pt3-2 | 208 | 222.9 | 7.2% | 254.3 | 22.3% |

voltages.

The power in column 4, 6 and the percentage in column 5, 7 are the power for nearly the same cost in number of voltage islands and the increased percentage according to the lowest power column 3 of Table 4.2. We can see that at least 10% - 20% power consumption can be saving by our method, not to mention the good shape of our generated voltage islands.

Table 4.3 show the comparisons of our results with [17] which considers only alignment and performance constraints. Our approach considers performance constraints and the generation of voltage islands.

Finally, we show Figure 4.7 and 4.8, two placement results that simultaneously consider voltage island generation and performance constraints. Figure 4.7 is a placement example applied tighter constraints, while Figure 4.8 is a looser one.

Table 4.3: Comparisons of the floorplanning/placement results including performance constraints with [17].

| Circuit | Table | Perf. | Perf. Constraint Only[17] | | | | Ours | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Area | Dead | P | C | Area | Dead | P | C |
| ami33 | pt3 | 3 | 1.181 | 2.2% | 113.6 | 4.34 | 1.18 | 2.02% | 121 | |
| | pt3-1 | | | | 131.1 | 4.93 | 1.181 | 2.2% | 145.1 | |
| ami49-2 | pt2 | 3 | 36.56 | 3.1% | 147.1 | 4.5 | 36.78 | 3.64% | 156 | |
| | pt3 | | | | 142 | 6.33 | 36.89 | 3.93% | 154.5 | |
| | pt3-1 | | | | 183.1 | 6.89 | 36.87 | 3.86% | 200.9 | |
| | pt3-2 | | | | 208 | 6.7 | 36.89 | 3.93% | 221.9 | 1 |
| ami49-3 | pt2 | 6 | 36.64 | 3.3% | 147.1 | 4.48 | 36.8 | 3.68% | 156.8 | |
| | pt3 | | | | 142 | 6.43 | 36.98 | 4.14% | 149.7 | |
| | pt3-1 | | | | 183.1 | 6.6 | 37.1 | 4.46% | 215.9 | |
| | pt3-2 | | | | 208 | 6.25 | 37.07 | 4.38% | 223.3 | |

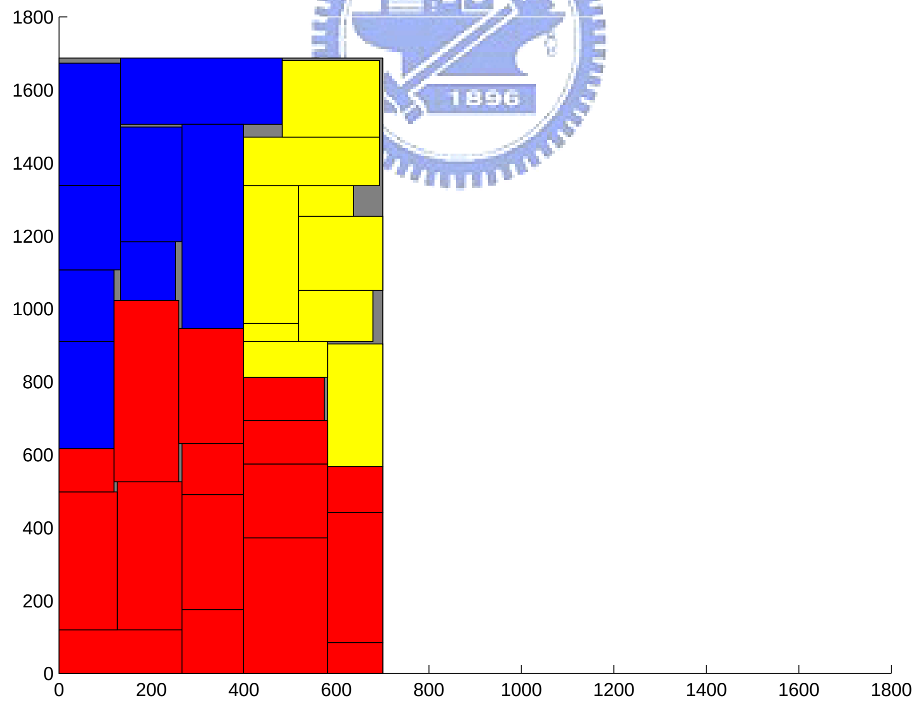


Figure 4.1: A feasible placement of circuit ami33 with 3 usable supply voltages. Dead space=2.07%, power=123.2$mW$ while the lowest power =113.6$mW$ (since we want smaller dead space, we must give away some power).

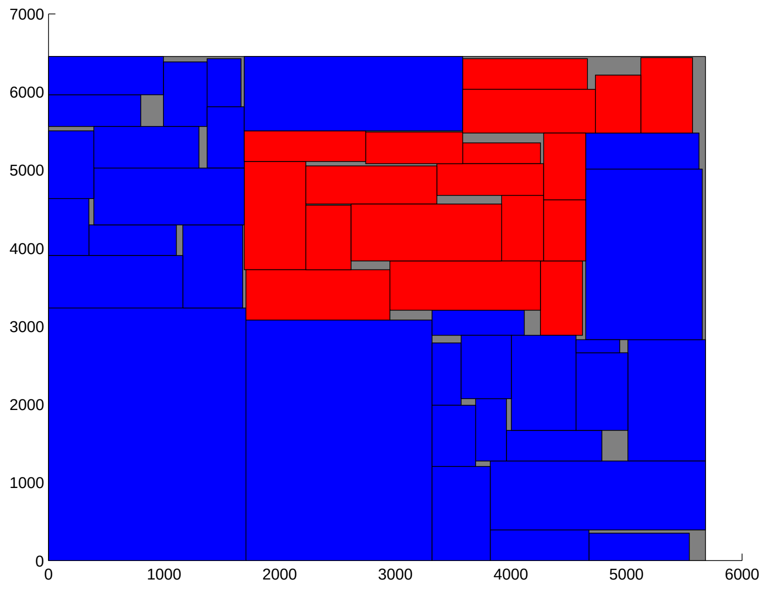


Figure 4.2: A feasible placement of circuit ami49 with 3 usable supply voltages power table information, but this placement result only use 2 kinds of supply voltages. Dead space=3.38%, power=156.2$mW$ while the lowest power =142$mW$.

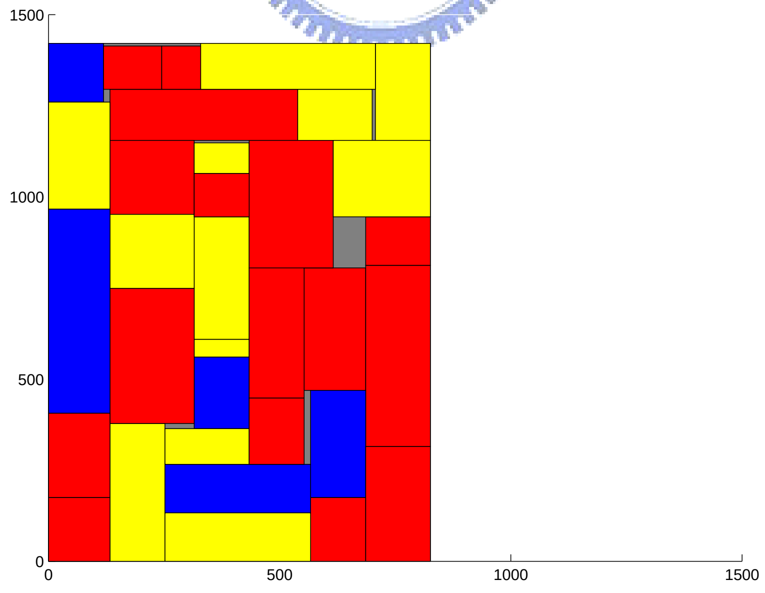


Figure 4.3: A traditional placement of circuit ami33 without voltage island generation methodology. Dead space=1.47%, and the power routing/level converter area cost=4.77.

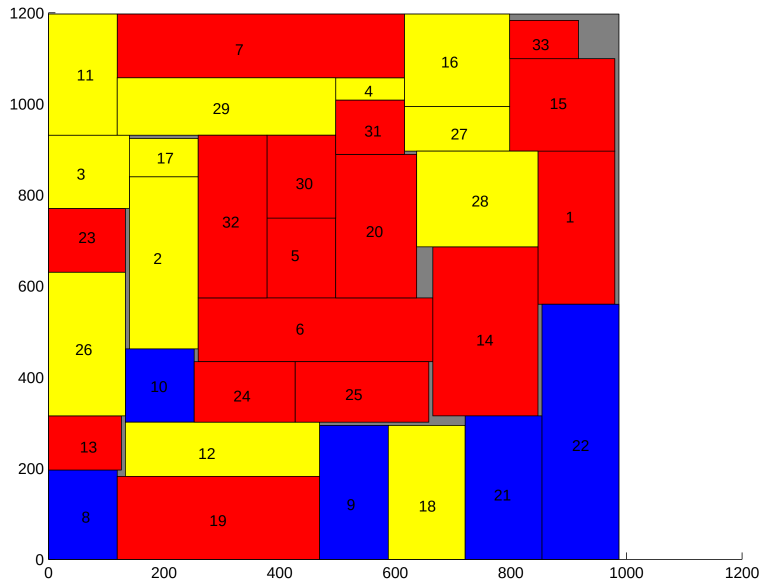


Figure 4.4: An illustration of circuit ami33 without voltage island methodology. Dead space=2.12%, power=113.6$mW$.

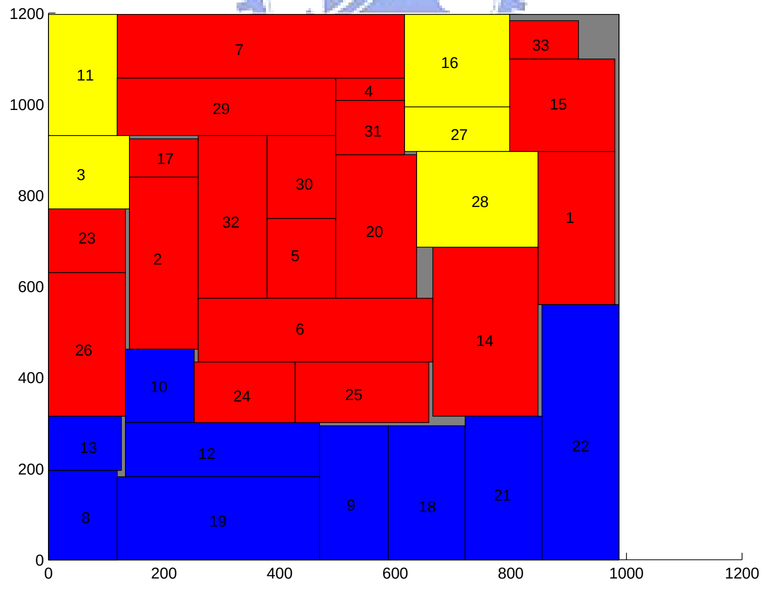


Figure 4.5: An illustration of circuit ami33 applying heuristic of supply voltage adjusting method for original B*-Tree result in Figure 4.4. We raise some blocks' supply voltages to reduce some power routing and area overhead due to level converters. Power=136.8$mW$, a 20.4% increase in power consumption compares to the lowest power dissipation.

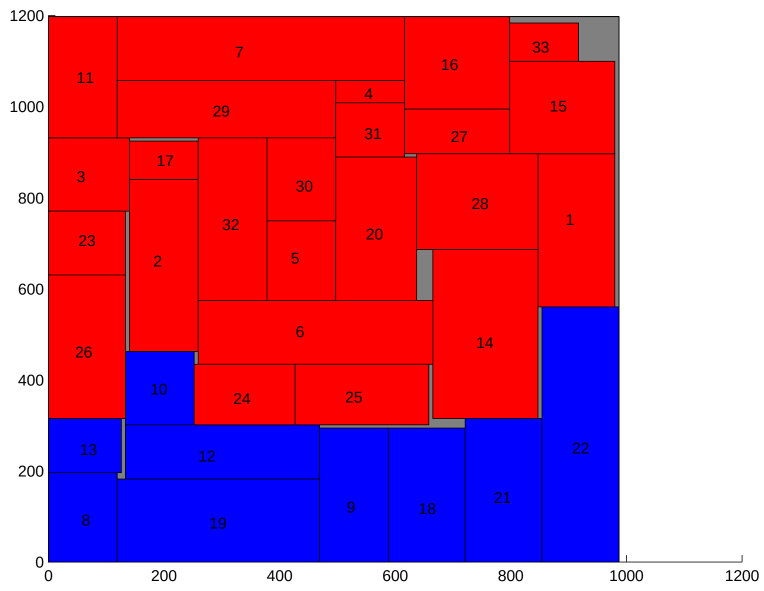


Figure 4.6: An example of circuit ami33 from Figure 4.4 raising all the lowest supply voltage and forming two voltage islands finally. Power=146.3$mW$, a 28.8% increase in power consumption.

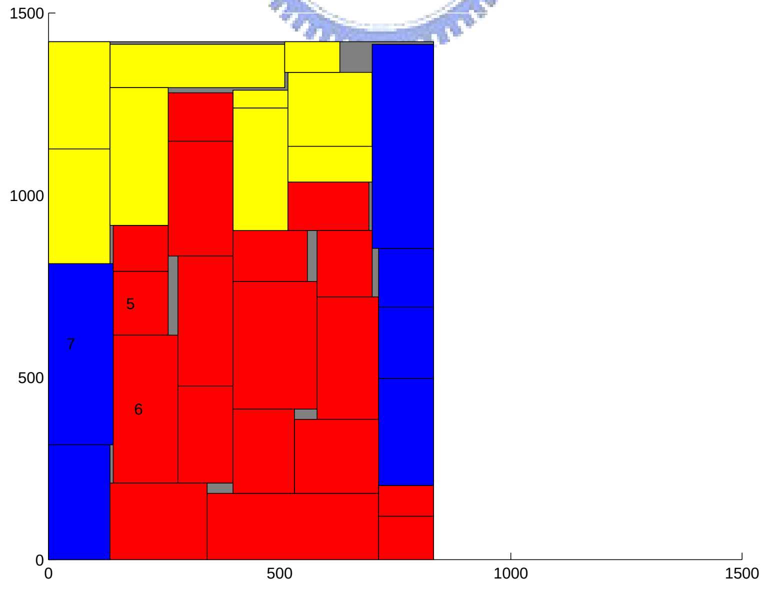


Figure 4.7: A feasible placement result with performance constraints consideration. Circuit ami33 with blocks 5, 6 and 7 under tighter performance constraints. Dead space=2.30%, power=116.9$mW$ while the lowest power =113.6$mW$.

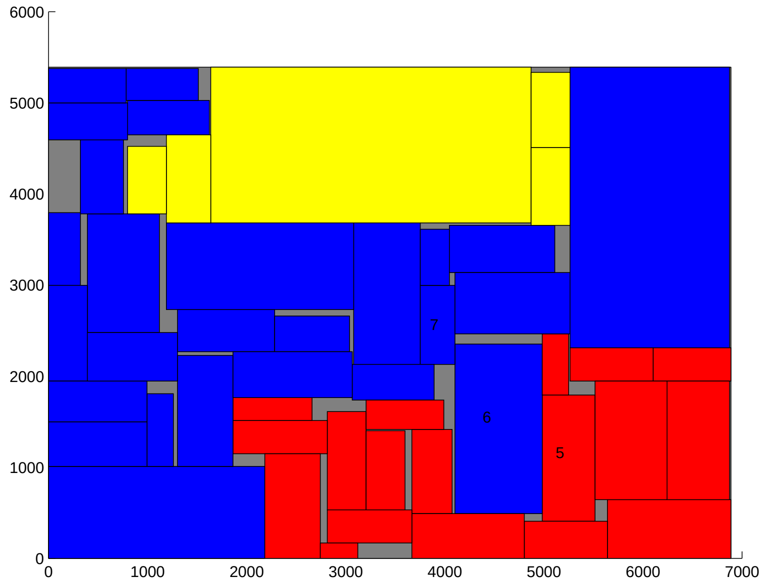


Figure 4.8: A feasible placement result with performance constraints consideration. Circuit ami49 with blocks 5, 6 and 7 under looser performance constraints. Dead space=4.53%, power=146.6$mW$ while the lowest power =142$mW$.

# Chapter 5

# Conclusion and Future Work

In this thesis, we have presented an effective algorithm to deal with the floor-planning/placement with voltage islands consideration and performance constraints. The algorithm is based on the B*-tree representation and the simulated annealing framework. According to the circuit power table information and the idea of location constraint (LC relation), we can group a set of cores using the same supply voltage and form a good shape of voltage island. The experimental results have shown the effectiveness and efficiency of our algorithm.

As the driving of the consumer product market and the growing use of portable and wireless electronic systems, low power design consideration has become essential today. There are many techniques to reduce power dissipation at each level of the design abstraction in [13], many things we need to take into consideration from other levels of design flow, even some are never been considered in the floor-planning/placement stage before. To study other low power design methodologies, and to combine and integrate these methods altogether to reach a whole low power system consideration will be a good orientation of future trend. We will focus on this as our future work.

# Bibliography

[1] J.-A. Carballo, J.L. Burns, Seung-Moon Yoo, I. Vo, and V.R. Norman. "A semi-custom voltage-island technique and its application to high-speed serial links [CMOS active power reduction]". In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 60–65, 2003.

[2] Y.-C. Chang, Y.-W. CHANG, G.-M. Wu, and S. W. Wu. "B*-Trees: A new representation for non-slicing floorplans". In *Proceedings IEEE/ACM Design Automation Conference*, pages 458–463, 2000.

[3] P. Coussy, A. Baganne, and E. Martin. "A design methodology for integrating IP into SOC systems". In *Proceedings of the IEEE*, pages 307–310, 2002.

[4] Y.S. Dhillon, A.U. Diril, A. Chatterjee, and Hsien-Hsin Sean Lee. "Algorithm for achieving minimum energy consumption in CMOS circuits using multiple supply and threshold voltages at the module level". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 693–700, 2003.

[5] P.-N. Guo, C.-K. Cheng, and T. Yoshimura. "An O-tree representation of non-slicing floorplan and its applications". In *Proceedings IEEE/ACM Design Automation Conference*, pages 268–273, 1999.

[6] Jingcao Hu, Youngsoo Shin, N. Dhanwada, and R. Marculescu. "Architecting voltage islands in core-based system-on-a-chip designs". In *Proceedings of the*

*International Symposium on Low Power Electronics and Design*, pages 180–185, 2004.

[7] Wei Hwang. "New trends in low power SoC design technologies". In *IEEE International SOC Conference*, page 422, 2003.

[8] J. Kao, A. Chandrakasan, and D Antoniadis. "Transistor sizing issues and tool for multi-threshold CMOS technology". In *Proceedings IEEE/ACM Design Automation Conference*, pages 409–414, 1997.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In *Science*, pages 671–680, 1983.

[10] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn. "Managing power and performance for system-on-chip designs using voltage islands". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 195–202, 2002.

[11] J. D. Meindl. "Low power microelectronics: Retrospect and prospect". In *Proceedings of the IEEE*, pages 619–635, 1995.

[12] K. Nose and T Sakurai. "Optimization of VDD and VTH for low-power and high-speed applications". In *Proceedings IEEE/ACM Design Automation Conference*, pages 25–28, 2000.

[13] M. Pedram and J. Rabaey. *"Power aware design methodologies"*. Kluwer Academic Publishers, 2002.

[14] X. Tang and D. F. Wong. "Floorplanning with alignment and performance constraints". In *Proceedings IEEE/ACM Design Automation Conference*, pages 848–853, 2002.

[15] A. Vorg, M. Radetzki, and W. Rosenstiel. “Measurement of IP qualification costs and benefits”. In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pages 996–1001, 2004.

[16] L. Wei, Z. Chen, K. Roy, M.C. Johnson, Y. Ye, and V.K. De. “Design and optimization of dual-threshold circuits for low-voltage low-power applications”. In *Very Large Scale Integration (VLSI) Systems, IEEE Transactions*, pages 409–414, 1997.

[17] Meng-Chen Wu. and Yao-Wen Chang. “Placement with alignment and performance constraints using the B*-tree representation”. In *Proceedings IEEE International Conference on Computer Design*, pages 568–571, 2004.