Performance-Constrained Voltage Assignment in Multiple Supply Voltage SoC Floorplanning

MENG-CHEN WU
National Chiao Tung University
MING-CHING LU
SpringSoft, Inc.
and
HUNG-MING CHEN and JING-YANG JOU
National Chiao Tung University

3

Using voltage island methodology to reduce power consumption for System-on-a-Chip (SoC) designs has become more and more popular recently. Currently this approach has been considered either in system-level architecture or postplacement stage. Since hierarchical design and reusable intellectual property (IP) are widely used, it is necessary to optimize floorplanning/placement methodology considering voltage islands generation to solve power and critical path delay problems. In this article, we propose a floorplanning methodology considering voltage islands generation and performance constraints. Our method is flexible and can be extended to hierarchical design. The experimental results on some MCNC benchmarks show that our method is effective in meeting performance constraints and can simultaneously consider the tradeoff between power routing cost and total power dissipation.

Categories and Subject Descriptors: B.7.2. [**Integrated Circuits**]: Design Aids—*Layout*

General Terms: Algorithms, Design

ACM Reference Format:

Wu, M.-C., Lu, M.-C., Chen, H.-M., and Jou, J.-Y. 2009. Performance-constrained voltage assignment in multiple supply voltage SoC floorplanning. ACM Trans. Des. Autom. Electron. Syst. 15, 1, Article 3 (December 2009), 17 pages.
DOI = 10.1145/1640457.1640460 http://doi.acm.org/10.1145/1640457.1640460

Authors' addresses: M.-C. Wu, H.-M. Chen, and J.-Y. Jou, Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan; M.-C. Lu, SpringSoft, Inc. Science-Based Industrial Park, Hsinchu, Taiwan; email: hmchen@mail.nctu.edu.tw.

DOI 10.1145/1640457.1640460 http://doi.acm.org/10.1145/1640457.1640460

1. INTRODUCTION

To cope with the increasing design complexity of System-on-a-Chip (SoC), hierarchical design and reusable IP (Intellectual Property) modules are widely used [Coussy et al. 2002; Vorg et al. 2004]. 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 [Hwang 2003; Meindl 1995]. Since the early stage of design will determine the overall chip performance, an efficient and effective power-aware floorplanning/placement approach is needed to improve the quality and shorten the design cycle.

The dynamic and static power dissipation in CMOS digital circuits both have direct relationship with supply voltage *Vdd*: dynamic power is proportional to Vdd^2 and static power is proportional to *Vdd*. Applying lower *Vdd* under the performance requirements is obviously one of the effective ways to reduce power consumption. One of the techniques to reduce power consumption is voltage island methodology, which is proposed by IBM [Lackey et al. 2002]. A voltage island is a group of on-chip cores powered by the same voltage source, independently from the chip-level voltage supply. This concept (in use of voltage islands) permits operating different portions of the design at different supply voltage levels.

Voltage island architecture can achieve power saving and has become more and more popular [Carballo et al. 2003; Hu et al. 2004; Lackey et al. 2002; Wu et al. 2005; Hung et al. 2005; Lee et al. 2006; Ching et al. 2006]. Hu et al. [2004] and Hung et al. [2005] partition IP cores into several subvoltage islands, floorplan each subvoltage island independently, and floorplan all voltage islands to form the final result. This approach somewhat restricts the exploration of solution space. In Wu et al. [2005], a postplacement approach of generating voltage islands is proposed. However, chip floorplanning level has more flexibilities. Moreover, since timing convergence is an important issue in deep submicron (DSM) and nanometer design, the critical delay should be bounded. Therefore floorplanning with performance constraints is a necessity [Tang and Wong 2002; Wu and Chang 2004].

In this article, we propose a methodology to preserve good voltage islands property, which can be viewed as the clustering of modules with same operating supply voltage in achieving lower power consumption. We adopt B*-tree [Chang et al. 2000] as our floorplan representation and underlying implementation since B*-tree can provide very good quality of nonslicing floorplans in area and wirelength costs, plus some properties for voltage islands generation. Our methodology can save power consumption and routing cost by location constraint [Chang et al. 2000], and solve the critical delay problems by performance constraint consideration [Wu and Chang 2004]. We generate voltage islands in the chip floorplanning stage instead of the postplacement one, in order to have more flexibilities in design. We have added heuristics to adapt the B*-tree algorithm to obtain voltage islands more easily and efficiently. Our approach can simultaneously consider voltage islands generation and performance constraints imposed by designers, even when the constrained modules

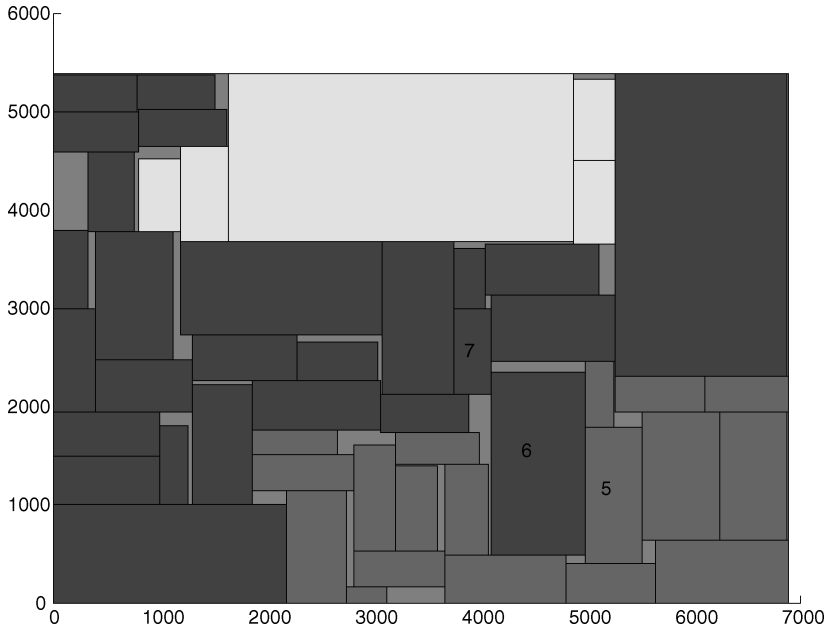

Fig. 1. A resultant floorplan $ami49$ from our approach which generates voltage islands with performance constraint consideration (dead space = 4.53%, power = $146.6mW$ while the lowest possible power = $142mW$). Blocks 5, 6 and 7 are under performance constraints and they are placed on different voltage islands.

are on different voltage islands. It is illustrated in Figure 1. Instead of two-stage iterative approaches in [Hu et al. 2004; Hung et al. 2005], we have one-stage floorplan packing methodology, which can explore more solution space. Experimental results based on some MCNC benchmarks with the additional power tables and constraints show that our method can meet the performance requirements while reducing the cost of power routing complexity.

The remainder of this article is organized as follows. In Section 2, we discuss chip level floorplanning strategy considering voltage islands architecture and performance-constraint blocks. In Section 3, we present our floorplanning algorithm for simultaneously dealing with voltage islands and performance constraints. Our experimental results are shown in Section 4, and we conclude the article in Section 5.

2. VOLTAGE ISLANDS ARCHITECTURE AND PERFORMANCE CONSTRAINTS IN CHIP LEVEL FLOORPLANNING

In this section, we briefly review the B*-tree representation, concepts of voltage islands, and performance constraints in floorplanning. The problem is then formulated.

2.1 Review of B*-Tree Representation

A B*-tree [Chang et al. 2000] is an ordered binary tree whose root corresponds to the module on the bottom-left corner for modeling a nonslicing floorplan.

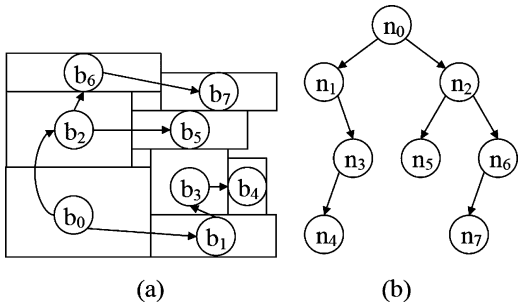

Fig. 2. A B*-tree placement. (a) An admissible placement. (b) The B*-tree representing the corresponding placement.

Given a B*-tree, we can also obtain an *admissible* placement by packing the blocks in linear time with a contour structure [Guo et al. 1999]. An admissible placement is a compacted placement, and no block can move down nor left in this placement. The packing based on the B*-tree representation uses simulated annealing algorithm with module moves (rotate, move to another place, swap, and remove-insert-best), including deletion and insertion in trees. In a B*-Tree T, the root of T represents the block on the bottom-left corner, the x-coordinate and y-coordinate 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; that is, $x_j = x_i + w_i$, 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 structure, we can compute the y-coordinate of a block in constant time.

Figure 2 illustrates (a) an admissible placement and (b) its corresponding B*-tree. Using the depth-first search (DFS) procedure, the B*-tree T for an admissible placement P can be constructed in a recursive fashion. In Figure 2, 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*-tree is simple, efficient, effective, and flexible and particularly suitable for representing a nonslicing floorplan with various types of modules (such as preplaced and rectilinear) and for creating or incrementally updating a floorplan. We use the B*-tree representation and simulated annealing algorithm as our underlying engine and embed the voltage islands generation concept to save power dissipation.

2.2 Voltage Island Methodology

The combination of increasing active power density and leakage currents has created a power management problem in the semiconductor industry. Performance-critical elements of the design require the highest voltage level

to maximize performance, while other coexisting functional cores may not need this voltage level and can work at lower voltages to save significant active power. This idea enables the concept of voltage island architecture [Lackey et al. 2002].

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. The cores powered by the same voltage should be grouped together without violating design metrics such as timing and wire congestion. Meanwhile, the number of voltage islands should be appropriate (not too many) considering signal level maintenance and communication between different islands, which requires level converters. We also need to consider power routing complexity [Hu et al. 2004] for design cost. Hence the overhead for applying voltage islands methodology with respect to area and delay is inevitable.

2.3 Performance Constraints Consideration in Floorplanning

Performance is one of the major concerns since the interconnect delay dominates 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 to optimize performance or to meet the delay constraints by placing these blocks/cores with critical nets close enough to each other. In Tang and Wong [2002], the maximum delay of performance constrained blocks is bounded by the summation of its height and width of the bounding box enclosing those blocks. However, it is not trivial to bound the maximum delay for those performance constrained blocks in voltage island architecture, especially for those which are not in the same voltage island.

2.4 Problem Formulation

For voltage island planning, we use a simplified model for modules/IPs, based on the setup in [Hu et al. 2004]. Since the power consumption of an IP varies with different supply voltage, we use a power table for every IP. The power table is a list of matching pairs [*supply voltage, power dissipation*] and specifies the legal voltage levels to work functionally and the corresponding average power dissipation values as illustrated in Figure 3 . We set this power dissipation based on IP's timing constraint and circuit size.

The problem discussed in this article is as follows. Let $B = \{b_1, b_2, \ldots, b_n\}$ be a set of n rectangular modules whose width, height, and area are denoted by W_i, H_i, and A_i, $1 \leqq i \leqq n$. Let (x_i, y_i) denote the coordinates of the bottom-left corner of module b_i, $1 \leqq i \leqq n$, on a chip. Each module is associated with a power table. A floorplan/placement P considering the performance constraints and voltage islands generation is an assignment of (x_i, y_i) for each b_i, $1 \leqq i \leqq n$, such that cores are clustered using the same voltage to form appropriate number of voltage islands and achieving low power consumption, while no two modules overlap and the given performance constraints are satisfied. For different voltage islands, there are level converters to modify the voltages of signals. If there are more level converters, we have to pay more power routing cost. Since we need

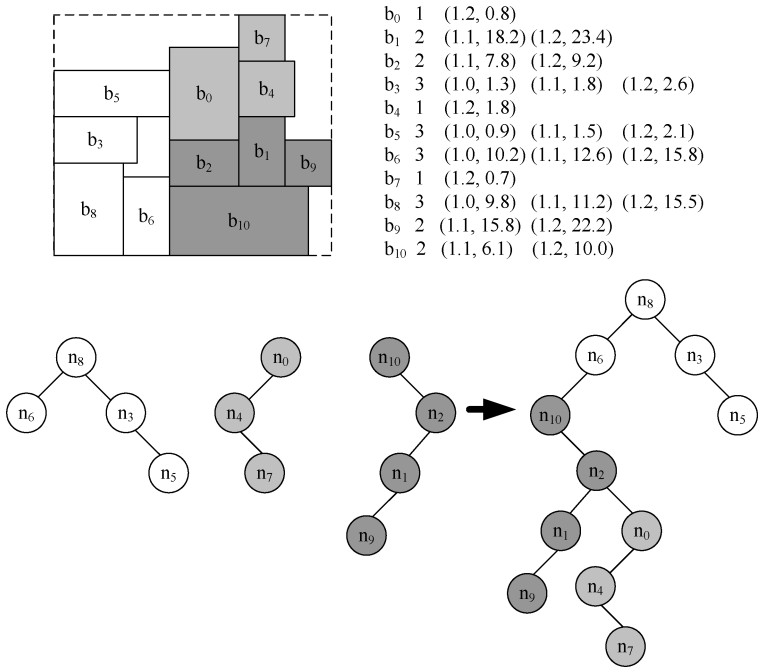

Fig. 3. An illustration of an intuitive approach to generate voltage islands in chip-level design. One obvious way to maximize power saving in floorplanning is to operate each block at its lowest available voltage. We partition the blocks according to their lowest supply voltage, then construct the subtrees of those compatible blocks. We then build the B*-tree and the corresponding floorplan. This approach will seriously limit the exploration of the solution space.

to reduce the power routing cost during the floorplan stage as well, the goal of our algorithm is to simultaneously minimize the packing area, power routing cost, and total power dissipation, while meeting performance constraints.

3. PERFORMANCE-CONSTRAINED VOLTAGE ASSIGNMENT IN MULTIPLE SUPPLY VOLTAGE FLOORPLAN DESIGN

In this section, we propose the heuristics for voltage islands generation with B*-tree representation, discuss the strategy to consider performance constrained blocks during floorplanning under voltage island architecture, and present our algoritm.

3.1 Floorplanning with Voltage Islands Generation

We first give an example to show the setup in creating voltage islands in SoCs using B*-trees and one intuitive strategy. In Figure 3, each core is followed by a number identifying its operable voltages, which are associated with a power table. For instance, the block b_3 can operate at $1.0V$, $1.1V$, or $1.2V$, and its corresponding power consumption are $1.3mW$, $1.8mW$, and $2.6mW$. One obvious way to maximize power saving in floorplanning is to operate each block at its lowest voltage, which means that we need at least 3 voltage islands: one for

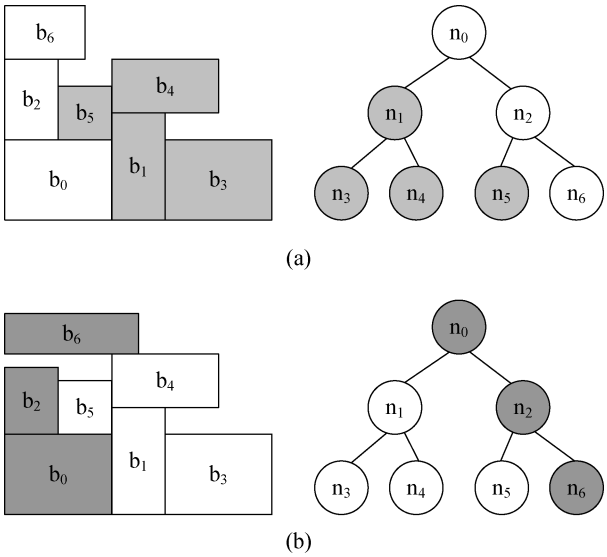

Fig. 4. Examples to explain the condition that nodes are not in the same subtree does not mean they do not abut physically; and the condition that two nodes abut does not represent the situation that the corresponding two blocks abut directly. In (a), node n_5 is not in the same subtree with $\{n_1, n_3, n_4\}$; but in the floorplan, block b_1, b_4 and b_5 are adjacent. In (b), nodes $\{n_0, n_2, n_6\}$ abut one by one and are in one subtree; but in the floorplan, block b_2 is not adjacent to block b_6. There are extra area overheads in this voltage island.

$\{b_0, b_4, b_7\}$, one for $\{b_1, b_2, b_9, b_{10}\}$, and one for $\{b_3, b_5, b_6, b_8\}$. This arrangement is obviously not optimal since the exploration of solution space is limited and the cost 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.

An important observation to create the voltage islands is to constrain the relationship between each pair of nodes which exist the parent-child relationships in the B*-tree, which means to cluster the blocks with the same supply voltage (say *compatible*), grouping them to be a subtree in the corresponding B*-tree. However, the condition that nodes are not in the same subtree does not mean they do not abut physically. Similarly, the condition that two nodes abut in the tree does not always mean that the corresponding two blocks abut, as shown in Figure 4. We observe that not all the blocks constructed in one subtree will be put together to form one voltage island. It is therefore more practical to increase the probability of those nodes to be clustered together, then apply a simple validation to inspect if they really form a favorable island with better power routing cost.

From the B*-tree representation and observations mentioned previously, 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 abuts to the block b_i (or is visible and above to the block b_i). Thus the probability that a

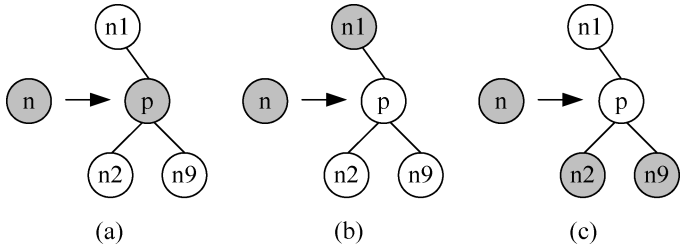

Fig. 5. Three swapping conditions to increase the probability of clustering the same voltage islands. (a) $V(p) = V(n)$. Node p and node n have the same voltage. (b) $V(p.parent) = V(n)$. Node n and the parent of node p have the same voltage. (c) $V(p.leftchild) = V(n)$ and $V(p.rightchild) = V(n)$. Node n has the same voltage with both children of node p.

node adds to a compatible subtree and the subtree grows and maps to the same voltage island shape will be increased. To implement this idea, we first randomly choose two nodes n, p in the tree ($V(n)$ and $V(p)$ denote the adopted voltages of node n and node p), and if the following conditions appear, we swap the positions of these two nodes in a B*-tree.

—$V(p) = V(n)$. Node p and node n are compatible, no new voltage islands will be created.

—$V(p.parent) = V(n)$. Node p's parent and node n are compatible. We swap node p and node n, and node n becomes the leaf of the subtree, or connects two or three compatible subtrees to form a larger subtree (if one or both p's children have the same voltage with node p's parent).

—$V(p.leftchild) = V(n)$ and $V(p.rightchild) = V(n)$. Node p's children both have the same voltage with node n. We swap node n and node p, and node n becomes the root of the subtree, and connects two compatible subtrees to form a larger subtree.

The three swapping conditions are explained in Figure 5. Besides, we modify two following operations in the B*-tree algorithm.

—Delete_Node. If we want to delete node n, we have to keep the connection of nodes with the same voltage. If the supply voltage of one child nodes ($n.leftchild$ or $n.rightchild$) is compatible with node n's parent, we choose it to be new child of node n's parent. If the children are in the same situation (both compatible or both not compatible), we randomly choose one of them.

—Insert_Node. If node n has to be inserted into one subtree and the subtree exists compatible nodes, the node n will be placed to join the cluster of the compatible nodes. If there does not exist any node compatible, we randomly choose one place to insert.

The reason why we need to remodel the basic procedures in constructing the B*-tree [Chang et al. 2000] is that it (or other floorplanners) only handles position constraints. Position constraints are concerned on the changes of blocks position, either precise position (such as pre-placed problems) or the relative position, that is, alignment or performance constraints problems. However, considering to maintain voltage island, the nodes in different supply voltages

should be ranked to different orders when needed to abut with other nodes or delete from the B*-tree.

Since the subtree construction is just a method to increase the possibility to place blocks/cores together, we need a property checking function to check if there exists a suitable voltage island. We do it after the contour updated to make sure the number of voltage islands is acceptable. Our checking is efficient since the block is only checked one time while it is placed at the segment of contour line, instead of checking all placed blocks.

3.2 Floorplanning with Performance Constrained Blocks in Voltage Island Architecture

Traditional floorplanners/placers minimize total wirelength but they can not guarantee critical nets to meet bounded delay. This problem becomes more important because timing convergence is a big issue in DSM design. In order to meet critical delay constraint, there are methods proposed in Tang and Wong [2002] and Wu and Chang [2004] during floorplanning. Since actual interconnect delay after appropriate buffer insertions will be close to linear in terms of distance, linear function in terms of distance to estimate delay is used. Assume there are a source at (x_s, y_s) and a sink at (x_t, y_t), and the delay of the net $D_{s,t} = \delta Dist_{s,t} = \delta(|x_t - x_s| + |y_t - y_s|)$, where δ is a constant to scale the distance to timing, and $Dist_{s,t}$ is the maximum distance between source and sink, equal to the half perimeter of the bounding box of the two points. Wu and Chang [2004] use the delay model to do subplacement (to place a set of feasible subplacements for the performance constrained blocks) by restricting the longest distance of performance constrained blocks: $Dist_{s,t} = (|x_t - x_s| + |y_t - y_s|) = D_{s,t}/\delta \leq D_{max}/\delta$, where D_{max} is the given maximum delay bound.

By applying this idea to a set of performance constrained blocks (whose areas are $A_i, i = 1, 2, \ldots, k$), they can get some rectilinear super blocks that the width W_{perf} and height H_{perf} satisfy the performance constraint: $W_{perf} + H_{perf} = B \leq B_{max}$, where B_{max} is the maximum bounded distance. Among the placements (rectilinear super blocks) meeting the performance constraints, they pick the one with the minimum dead space $S_{perf} = W_{perf} * H_{perf} - \sum A_i$ and fix the rectilinear block (and thus fix the delay) for further processing with other blocks. By using the preclustered shape-fixed appropriate rectilinear block, they guarantee that the performance constraints will be satisfied throughout the remaining processing. In Tang and Wong [2002], it is different in choosing an appropriate rectilinear super block, where they use a method that adjusts the W_{perf} and H_{perf} dynamically into the rectilinear super blocks. At higher temperature in annealing process, let W_{perf} and H_{perf} to be half-half: $W_{perf} = H_{perf} = B_{max}/2$. 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 cannot be changed easily to exactly capture delay bound.

There is a major problem in this performance model using voltage island architecture. The performance of each block does not vary with supply voltage. The legal supply voltage has a big impact on the driving strength, thus the

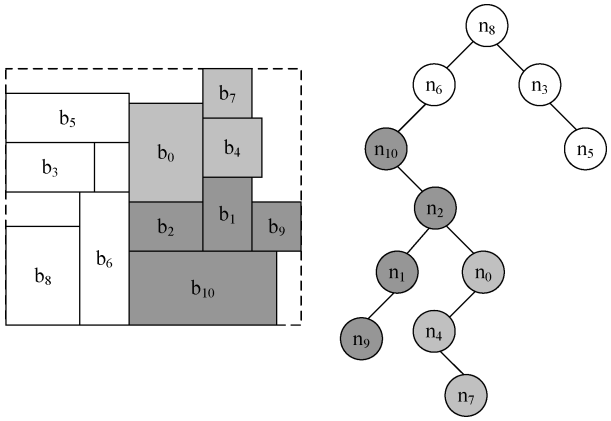

Fig. 6. The *diameter* is the number of nodes between two nodes. Each pair of blocks whose corresponding nodes have 0 diameter is almost abutting to each other, except blocks b_3 and b_8. Two nodes whose diameter is less than or equal to 2 are near to each other. Blocks b_0 and b_5 are adjacent to each other though the diameter of nodes n_0 and n_5 is larger than 2.

bounding box size. If signals are communicated by high supply voltage, the bounding box for performance constrained blocks will be larger than one with lower voltage. In addition, the allowable box size should be a function of the supply voltage. In Wu et al. [2003], the *location constraint* (LC for short) was proposed to maintain the relation of blocks after packing of B*-trees. The blocks with LC relation will be placed with the desired shape (L-shape or T-shape) in the final floorplan. Wu and Chang [2004] extend the LC relation to *alignment shape* to guarantee the alignment constraints will be satisfied at all times. From these two works and the observation in Section 3.1, we know that if the nodes in the B*-tree are closer to each other, the chance that blocks are next to each other will be higher in the final floorplan. We then define a *diameter* of two nodes in the B*-tree: the number of nodes between those two nodes. Two nodes with 0 diameter are in the direct parent-child relationship. We use Figure 6 (from Figure 3) to illustrate this idea. In Figure 6, we can see that the blocks whose corresponding nodes with diameter less than or equal to 2 are almost adjacent or near to each other. There exists an exception that blocks b_0 and b_5 are placed next to each other but their corresponding nodes have the diameter larger than 2. Though the *diameter* does not guarantee the final relation of blocks, we could use it to guide the desired blocks being placed near each other without restricting their shapes (as in Wu and Chang [2004]).

Our approach combines the advantages of these methods [Wu et al. 2003; Wu and Chang 2004; Tang and Wong 2002], keeping the flexibility of the subplacement for the performance constraints. We do not pick the minimum dead space subplacement and fix the shape (or the relational position) of the performance constrained blocks before processing with other blocks at the beginning. Instead, in a B*-tree, we set the diameter of performance constrained nodes and let the performance constrained nodes be handled with other nodes as if they are not under restriction. And then, after floorplanning, we check whether these performance constrained blocks are placed in the desired bounding box

or not. If these performance constrained blocks are in the desired bounding box, we accept this floorplan. The bounding box will be shrunk at every time we accept a new floorplan until meeting the performance constraints, like mentioned in Tang and Wong [2002]. Thus, the total area and wirelength can be better optimized. This is further verified in the condition that supply voltages of the performance constrained blocks are possibly different.

In experiments, the initial bounding box is the bounding box of first floorplan we got after first packing. We use its half perimeter to calculate first D_{bound}, where D_{bound} presents the delay of the shrinking bounding box as we mentioned, and we can get $D_{s,t}$ of performance constrained blocks. There is no doubt we have a first valid floorplan whose $D_{s,t} < D_{bound}$. Every time we accept a valid floorplan, we decrease the D_{bound} and repeat it until $D_{bound} <= D_{max}$. Once the condition $D_{s,t} <= D_{bound} <= D_{max}$ is achieved, we get a floorplan meeting performance constraints and can use D_{max} to constrain performance blocks in further process. By reducing D_{bound} step by step, we can get more solution space than Wu and Chang [2004] and guarantee the final floorplan meets the performance constraints by $D_{bound} <= D_{max}$ as Tang and Wong [2002].

3.3 The Algorithm

Our floorplanning/placement design algorithm is based on the simulated annealing method and we only consider hard modules in this article. We perturb a B*-tree to another by the following operations.

—*Op*1: Change the supply voltage of a block.
—*Op*2: Rotate a block.
—*Op*3: Flip a block.
—*Op*4: Swap two blocks.
—*Op*5: Move a block to another place.

The first operation *Op*1 is designed for blocks' voltage perturbation. The block which has more than two voltages will be involved by the operation. In this operation, we change the voltage of the block to be the same with its parent or children if possible and expect to generate larger voltage island. In *Op*2, we rotate a block. This action can be applied to any node without changing the relations between any two nodes. In *Op*3, we flip a block. It can be applied to any blocks including performance constrained blocks, since we do not bind them together at the beginning. *Op*4 and *Op*5 change the relations of blocks to get a different placement and B*-tree structure based on our heuristics. Like the *Op*1, these two operations expect to extend the voltage islands. To get better results by *Op*4, we give the higher probability to the two nodes with the same voltage, and lower probability to the two nodes with different voltages. In the original B*-tree, it gives the random move for *Op*5. In our algorithm, we use the new *Delete_Node* and *Insert_Node* here to generate voltage islands as large as possible. Moreover, if we tighten the shape of performance constrained blocks at the beginning, we may be forced to raise the supply voltages of some of them to the higher one to match the voltage of the island which these blocks belong to; or we will get a malicious B*-tree structure that the voltage property

```
Algorithm: Floorplanning with Performance Constraints
           for Voltage Islands(blocks, power_table, constraints)
Input: A set of blocks, the power table and performance constraints.
Output: A feasible voltage islands placement
        without violating the given constraints.
1.  Initialize a B*-tree for the input blocks and constraints;
2.  Simulated annealing process;
3.  do
4.      perturb();
5.      performance constraint check();
6.      if diameters of performance-constrained blocks fail to meet constraints
7.          then adjust the corresponding nodes in the B*-tree
8.      packing();
9.      voltage island check();
10.     if the property is not good
11.         then add the penalty to the cost function
12.     evaluate the cost of this floorplanning;
13. until converged or cooling down;
14. return the best solution;
```

Fig. 7. The algorithm of floorplanning with performance constraints for voltage island generation.

is withered. When the temperature becomes lower in annealing process, we do not accept a solution that violates performance constraints even if it has better cost, the best solution will be kept until next feasible solution with better cost.

Figure 7 shows our algorithm in detail. First, we initialize the B*-tree and set up the power table and constraints (see line 1). We construct the initial B*-tree according to the order which the blocks are presented and set all blocks to their lowest voltage. Then, we start the simulated annealing (SA) process. After each perturbation (see line 4), we check the performance constrained blocks (see line 5). Because there are no exact coordinates of performance constrained blocks in this stage (before packing), we use the *diameter* to maintain their relation in the B*-tree (close to each other). If there exists any diameter between each pair of nodes of performance constrained blocks larger than 2, we will reorder their relations and make sure there is no diameter larger than 2 to expect the more adjacency of performance-constrained blocks (see line 7). The adjusting function consists two steps: remove the node with the largest diameter and then insert it into the original group with diameter less then 2. Then we do the packing, check the performance constrained blocks, accept a placement without violating performance constraints (see line 8), and pass it to the next step. If the performance constraints check fails, we treat it as a failing step in the simulated annealing process. We check the property of voltage islands for this placement (see line 9), if the property is not good for this placement (ex. too many voltage islands or too many level converters), we penalized the solution so that it will be rejected by SA. After checking the property of voltage island, the placement is evaluated by its area, wire length, the number of voltage islands, and the cost of level converters (see line 12). This cost function, $cost = \alpha\text{area} + \beta\text{power} + \gamma\text{power routing}$, where $\alpha + \beta + \gamma = 1$, takes care of area of the floorplan, the total power consumption and the power routing cost of voltage

islands. In our experimental setting, we set $\alpha = 0.9$, $\beta = 0.05$, and $\gamma = 0.05$. The iteration continues until the end of SA scheme (see lines 3–13) and the best solution is reported(see line 14). According to this flow, we can guarantee the performance constraints are met during the perturbation and we will try to get the placement with good property of voltage islands at each perturbation.

4. EXPERIMENTAL RESULTS

We implemented our algorithm in C++ on a PC with P4-2.4GHz CPU and 440MB memory. Our method can handle circuits that have two or three kinds of supply voltages, circuits with more than three supply voltages are applicable as well. If there is only one supply voltage for the circuit, our program will be applied like the original B*-tree simulated annealing method, and will not spend extra runtime to handle voltage island property. The good voltage island property means that the number of voltage island and power routing cost are acceptable. For verifying our observation in voltage islands generation on large number of blocks, we apply our approach on some of the MCNC benchmarks with more blocks, and compare with Chang et al. [2000], the original B*-tree with simulated annealing method. For adopting voltage island architecture, power routing cost and level converter issues should be addressed.

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, and 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 I shows the comparison between Chang et al. [2000] and our approach on power consumption and power routing cost, where columns 1 and 2 give the name of the circuit (the number of blocks) and the power tables we use, and the power consumptions in column 4 are lowest since we use the lowest available voltages for every block in these circuits, and we compare the power routing/overhead converter area cost from Chang et al. [2000], which are normalized to ours. We experiment our method by testing with different power tables, which are composed 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 means the power table that contains 2 different supply voltages, while pt3, pt3_1, pt3_2 are power tables containing 3 folds. For power consumption, we sum up every block's power consumption at its operated voltage. From Table I, we can see

Table I.
The comparisons between pure B*-tree [Chang et al. 2000], the results of [Hu et al. 2004] which are implemented based on the B*-tree representation, and our approach on power consumption and power routing cost. The results are obtained from some MCNC benchmarks with different power tables. *C* is level converters area and routing cost, normalized to our approach. With slightly more power consumption, we can obtain much lower power routing cost in voltage islands generation. The unit of *P* is *mV* and *CPU* is second.

		Original B*-tree				Results of [Hu et al. 2004]				Ours			
Circuit	Table	Dead	P	C	CPU	Dead	P	C	CPU	Dead	P	C	CPU
hp	pt2	1.4%	83.7	1.81	4	3.10%	86.4	1	18	3.10%	86.4	1	15
	pt3		73.4	2.38		2.98%	78.30	1	22	2.98%	78.3		18
ami33	pt3	1.47%	113.6	4.52	26	4.25%	115.8	1.458	82	2.07%	123.2		89
	pt3-1		131.1	4.76		3.75%	136.6	2.461	86	2.23%	136.3		89
ami49	pt2	3.68%	147.1	4.18	53	4.08%	157.9	1.49	263	3.34%	151.5		243
	pt3		142	5.43		4.42%	151.1	1.53	246	3.38%	156.2		234
	pt3-1		183.1	6.11		5.24%	200.1	1.12	275	3.52%	196.4		234
	pt3-2		208	5.97		4.21%	223.7	1.20	255	3.64%	222.9		240

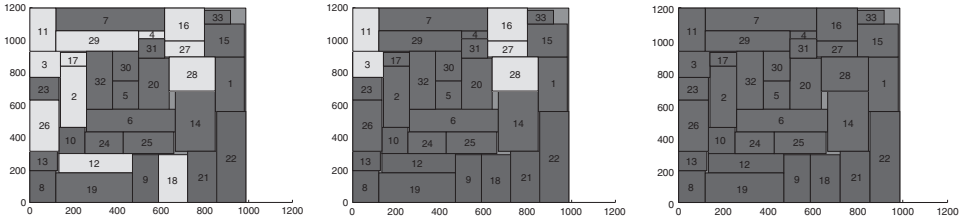

Fig. 8. The floorplans of circuit *ami*33 with 3 usable supply voltage. (a) An illustration of ami33 without voltage island methodology. Dead space = 2.12%, power = 113.6*mW*. (b) An illustration of applying heuristic of supply voltage adjusting method for original B*-tree resulting in Figure 8(a). We raise supply voltages of some blocks to reduce some power routing and area overhead due to level converters. Power = 136.8*mW*, and 20.4% increasing in power consumption compares to the lowest power dissipation. (c) An example from Figure 8(a) raising all the supply voltages and forming two voltage islands finally. Power = 146.3*mW*, and 28.8% increases in power consumption compared to the lowest power dissipation.

that our power consumption is a little more than the lowest power listed in column 4, but our routing/level converters cost is about 16.4%–55.2% less when compared with Chang et al. [2000].

There are many works discussing about voltage island issues. To demonstrate the effectiveness of our algorithm, we implement the work of Hu et al. [2004] based on the B*-tree representation, which is comparable in experimental setup and assumptions to our work, the corresponding results are also shown in Table I. For the benchmark of ami33 pt3, it obtains the result of dead space = 4.25%, power = 115.80, and level converter cost = 1.458. Because their method merges subvoltage islands first and floorplans these subvoltage islands, their method will get better result in power consumption. For dead space and level converter cost, our algorithm has better results on all benchmarks.

Here we use some figures to illustrate the differences between Chang et al. [2000] and the proposed algorithm. Figure 8(a) shows the final results from the

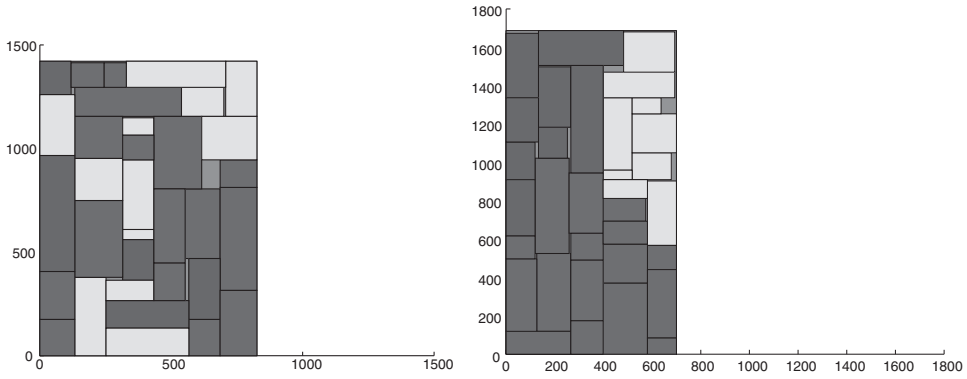

Fig. 9. Two floorplans of circuit *ami*33 with 3 usable supply voltage. (a) Floorplan with much higher power routing complexity since the number of voltage islands is large. Dead space = 1.47%, power = 113.6mW. (b) Floorplan with nice voltage island property (slightly more power dissipation). Dead space = 2.07%, power = 123.2mW.

Table II.
The comparison between two approaches on power consumption and power routing cost. With both meeting performance constraints, our approach obtains much lower power routing cost with slightly more power consumption.

Circuit	Table	Perf.	Perf. Const. Only [Wu and Chang 2004]				Ours			
			Area	Dead	P(mw)	C	Area	Dead	P(mw)	C
ami33	pt3	3	1.181	2.2%	113.6	4.34	1.18	2.02%	121	1
	pt3-1				131.1	4.93	1.181	2.2%	145.1	1
ami49-2	pt2	3	36.56	3.1%	147.1	4.5	36.78	3.64%	156	1
	pt3				142	6.33	36.89	4.53%	146.6	1
	pt3-1				183.1	6.89	36.87	3.86%	200.9	1
	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	1
	pt3				142	6.43	36.98	4.14%	149.7	1
	pt3-1				183.1	6.6	37.1	4.46%	215.9	1
	pt3-2				208	6.25	37.07	4.38%	223.3	1

original B*-tree. In this experiment, we fix each block at its lowest power and try to minimize the area. In Figure 8(b), we add the penalty in the cost function of the original B*-tree to minimize the boundary of different voltage islands. Finally, Figure 8(c) further reduces the number of voltage islands to two, which is to mimic the result applying our voltage island generation strategy, this result is therefore not in Table I. Figure 9 shows our experimental results and demonstrates the comparison between two floorplans of *ami*33, with and without voltage islands generation heuristic. From the experiments and these figures, we can observe that, for power saving, we could use the lowest voltage of each block, but we have to pay much more cost to level converters and power routing. Our algorithm can achieve a very good result at generating voltage islands with the acceptable cost.

According to Table II, our B*-tree based algorithm is also very efficient (the results reported in [Wu and Chang 2004] ran on a 440 Sun Sparc Ultra 10

machine). Table II shows the comparison of our results with Wu and Chang [2004], which considers only performance constraints. Column 3 gives the number of performance constrained blocks, there are one group in ami33 and ami49-2, two groups in ami49-3, and each group has 3 blocks. Both methods meet performance constraints but our approach could get much lower cost of level converters with slightly more power consumption. Figure 1 illustrates final floorplanning result of *ami*49 with performance constrained blocks 5, 6, and 7, and they are not on the same voltage island.

5. CONCLUSION

In this article, we have presented an effective algorithm to deal with the floorplanning 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, obtain appropriate number of voltage islands, and form good shapes of voltage islands. We also take performance constraints into consideration while generating voltage islands.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for providing suggestions to greatly improve this article.

REFERENCES

CARBALLO, BURNS, J., YOO, S.-M., VO, I, AND NORMAN, V. 2003. A semi-custom voltage-island technique and its application to high-speed serial links. In *Proceedings of the IEEE International Symposium on Low Power Electronics and Design*. 60–65.

CHANG, Y., CHANG, Y., WU, G., AND WU, S. 2000. B*-trees: A new representation for non-slicing floorplans. In *Proceedings of the IEEE/ACM Design Automation Conference*. 458–463.

CHING, L., YOUNG, E. F., LEUNG, K. C., AND CHU, C. 2006. Post-placement voltage island generation. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 641–646.

COUSSY, P., BAGANNE, A., AND MARTIN, E. 2002. A design methodology for integrating IP into SoC systems. In *Proceedings of the IEEE International Custom Integrated Circuits Conference (CICC)*. 307–310.

GUO, P.-N., CHENG, C.-K., AND YOSHIMURA, T. 1999. An O-tree representation of non-slicing floorplans and its applications. In *Proceedings of the IEEE/ACM Design Automation Conference*. 268–273.

HU, J., SHIN, Y., DHANWADA, N., AND MARCULESCU, R. 2004. Architecting voltage islands in core-based system-on-a-chip designs. In *Proceedings of the IEEE International Symposium on Low Power Electronics and Design*. 180–185.

HUNG, W.-L., LINK, G. M., XIE, Y., VIJAYKRISHNAN, N., DHANWADA, N., AND CONNER, J. 2005. Temperature-aware voltage island architecting in system-on-chip design. In *Proceedings of the IEEE International Conference on Computer Design*. 689–694.

HWANG, W. 2003. New trends in low power SoC design technologies. In *Proceedings of the IEEE SOC Conference*. 422.

LACKEY, D., ZUCHOWSKI, P., BEDNAR, T., STOUT, D., GOULD, S., AND COHN, J. 2002. Managing power and performance for system-on-chip designs using voltage islands. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 195–202.

LEE, W.-P., LIU, H.-Y., AND CHANG, Y.-W. 2006. Voltage island aware floorplanning for power and timing optimization. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 398–394.

MEINDL, J. 1995. Low power microelectronics: Retrospect and prospect. In *Proc. IEEE*. 619–635.

TANG, X. AND WONG, D. 2002. Floorplanning with alignment and performance constraints. In *Proceedings of the IEEE/ACM Design Automation Conference*. 848–853.

VORG, A., RADETZKI, M., AND ROSENSTIEL, W. 2004. measurement of IP qualification costs and benefits. In *Proceedings of the Design, Automation and Test in Europe*. 996–1001.

WU, G.-M., CHANG, Y.-C., AND CHANG, Y,-W. 2003. Rectilinear block placement using B*-trees. *ACM Trans. Des. Autom. Electron. Syst. 8*, 2, 188–202.

WU, H., LIU, I.-M., WONG, M., AND WANG, Y. 2005. Post-placement voltage island generation under performance requirement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 309–316.

WU, M.-C. AND CHANG, Y.-W. 2004. Placement with alignment and performance constraints using the B*-tree representation. In *Proceedings of the IEEE International Conference on Computer Design*. 568–571.

Received September 2007; revised August 2008, January 2009; accepted September 2009