Performance Constraints Aware Voltage Islands Generation in SoC Floorplan Design

Ming-Ching Lu
SpringSoft, Inc.
Science-Based Industrial Park
Hsinchu 300, Taiwan

Meng-Chen Wu, Hung-Ming Chen,
Hui-Ru Jiang
Department of Electronics Engineering
National Chiao Tung University, Hsinchu, Taiwan

Abstract— 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 post-placement 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 paper, 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 the approach is effective in reducing power consumption and routing cost and the assignment of supply voltage in modules.

I. INTRODUCTION

To cope with the increasing System-on-a-Chip (SoC) design complexity, hierarchical design and reusable IP (Intellectual Property) modules are widely used [3], [11]. 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 [7], [9]. 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 reduce the design cycle.

The dynamic power and static power dissipation in CMOS digital circuits both have direct relationship with supply voltage $V_{dd}$. Dynamic power is proportional to $V_{dd}^2$ and static power is proportional to $V_{dd}$. Applying lower $V_{dd}$ under the performance requirements is obviously the most effective way to reduce power consumption. One of the techniques to reduce power consumption is Voltage Island methodology, proposed from IBM [8]. 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 [1], [5], [8], [12], [6]. In [5], [6], iterative voltage island partitioning and floorplanning approach is used, but the exploration of solution space is somewhat restricted. In [12], a post-placement approach to generating voltage islands is proposed. However, chip floorplanning level has more flexibility. Moreover, since timing convergence is an important issue in deep submicron (DSM) design, the critical delay should be bounded. Therefore floorplanning with performance constraints is a necessity [10].

In this paper, 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 [2] as our floorplan representation and underlying implementation since B*-tree has provided very good quality of non-slicing 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 [2], and to solve the critical delay problems by performance constraint consideration [13]. Our main contributions include:

- We generate voltage islands in chip floorplanning stage instead of post-placement one, in order to have more flexibility 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 are on different voltage islands. It is illustrated in Fig. 1.

This work was partially supported by the National Science Council of Taiwan ROC under grant No. NSC 94-2220-E-009-020 and -042.

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 is142mW). Blocks 5, 6 and 7 are under performance constraints and they are placed on different voltage islands.

- Instead of two-stage iterative approaches in [5], [6], we have one-stage floorplanning packing methodology, which can explore solution space.
- Experimental results based on some MCNC benchmarks with the fabricated 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 paper is organized as follows. In Section II, we discuss chip level floorplanning strategy considering voltage islands architecture and performance constraint blocks. In Section III, we present our floorplanning algorithm for simultaneously dealing with voltage islands and performance constraints. Our experimental results are shown in Section IV and we conclude the paper in Section V.

II. 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.

A. Review of 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 B*-tree, we can also obtain an admissible placement by packing the blocks in linear time with a contour structure [4]. The packing based on 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. We adopt B*-tree [2] as our floorplan representation and underlying implementation due to its good quality of non-slicing floorplans in area and wirelength costs, plus some properties for voltage islands generation.

B. Voltage Islands Methodology

The combination of increasing active power density and leakage currents has created a power management problem in the semiconductor industry. Mostly performance-critical element of the design requires the highest voltage level to maximize performance,
while other coexisting functional cores may not need this voltage level, hence they can be run at lower voltages to save significant active power. This idea enables the concept of voltage island architecture[8].

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 source 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 translation and communication between different islands, which requires level converters. We also need to consider power routing complexity [5] for design cost. Hence the overhead for applying voltage islands methodology with respect to area and delay is inevitable.

C. Performance Constraints Consideration in Floorplanning

Performance is a concern 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 by binding them together to optimize performance or to meet the delay constraints by placing them close enough to each other. The constraint requires designated nets (blocks/cores) to be placed within a pre-defined bounding box nets. In [10], the maximum delay of performance constraint 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 constraint blocks in voltage island architecture, especially for those which are not in the same voltage island.

D. Problem Formulation

For voltage island planning, we use a simplified model for modules/IPs, based on the setup in [5]. Since the power consumption of an IP varies with different supply voltage, we use a power table, which is a list of matching pairs, (supply voltage, power dissipation), specifying the legal voltage levels to work functionally and the corresponding average power dissipation values, for every IP. We set this power dissipation based on IP’s timing constraint and circuit size.

The problem concerned in this paper is as follows. 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 $A_i$, $1 \leq i \leq n$. Let $(x_i, y_i)$ denote the coordinates of the bottom-left corner of module $b_i$, $1 \leq i \leq n$, on a chip. Each module is associated with a power table. A floorplan/placement $P$ considering the performance constraint and voltage islands generation is an assignment of $(x_i, y_i)$ for each $b_i$, $1 \leq i \leq n$, such that cores are clustered using the same voltage to form appropriate number of islands and achieving low power consumption, while no two modules overlap and the given performance constraints are satisfied. The goal is to simultaneously minimize the packing area, power routing cost and total power dissipation, while meeting performance constraints.

III. PERFORMANCE CONSTRAINTS AWARE VOLTAG EISLANDS GENERATION IN FLOORPLAN DESIGN

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

A. Floorplanning with Voltage Islands Generation

We first give an example to show the setup in creating voltage islands in SoCs using B*-tree and one intuitive strategy. In Fig.2 each core is followed by a number which identifies the number of its usable voltages, then associated with a power table. 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. One obvious way to maximize power saving in floorplanning is to operate each block at its lowest available voltage. We set this power dissipation based on IP’s timing constraint and circuit size. The problem concerned in this paper is as follows. 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 $A_i$, $1 \leq i \leq n$. Let $(x_i, y_i)$ denote the coordinates of the bottom-left corner of module $b_i$, $1 \leq i \leq n$, on a chip. Each module is associated with a power table. A floorplan/placement $P$ considering the performance constraint and voltage islands generation is an assignment of $(x_i, y_i)$ for each $b_i$, $1 \leq i \leq n$, such that cores are clustered using the same voltage to form appropriate number of islands and achieving low power consumption, while no two modules overlap and the given performance constraints are satisfied. The goal is to simultaneously minimize the packing area, power routing cost and total power dissipation, while meeting performance constraints.

Fig. 2. 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 by 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.

Fig. 3. An example to explain the condition that two nodes do not abut does not represent the situation that the corresponding two blocks abut; and the condition that nodes are not in the same subtree does not mean they do not abut physically. In (a), node $c_7$ is not in the samesubtree with $c_2, c_4, c_5$; but in the floorplan, block $c_2, c_5$ and $c_7$ are connected. In (b), nodes $\{c_1, c_3, c_6\}$ form a subtree in the B*-tree; but in the floorplan, block $c_3$ is not connected with block $c_6$. There are extra area overhead in this voltage island.

blocks which support two or more legal supply voltages to one of its higher legal voltages to alleviate the problems.

One key observation 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, which means to cluster the blocks with the same supply voltage (say compatible), grouping them to be a subtree in corresponding B*-tree. However, the condition that two nodes do not abut in the tig does not always mean that the corresponding two blocks abut. Similarly, the condition that nodes are not in the same subtree does not mean they do not abut physically. We give an example in Fig.3. We observe that not all the blocks constructed in a subtree will be put together to form a voltage island, so it is not necessary to place a B*-tree with perfect subtrees. We believe that it is more practical to increase the probability of those nodes to be clustered together, then apply a simple checking method to inspect if they really form a favorable island.

From above observation, we know that the area cost is getting lower and the dead space of the total area is becoming smaller due to B*-tree module packing, 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 $b$. The probability a node adds to a compatible subtree and the subtree grows and maps to a favorable voltage island shape will be improved. To implement this idea, we define 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 change the positions of these two nodes:

- $V(p) = V(n)$: Node $p$ and node $n$ are compatible. No need to worry about the property will be changed.
- $V(p, parent) = V(n)$: Node $p$'s parent and node $n$ are compatible, let $n$ be the leaf of the subtree or connect two compatible subtrees to a larger subtree.
- $V(p, leftchild) = V(n)$ and $V(p, rightchild) = V(n)$: Node $p$'s left child and right child both have the same voltage with node $n$, let $n$ be the root of the subtree or connect two compatible subtrees as well.

Except these conditions are considered for perturbation during simulated annealing, we modify two following operations in the B*-tree algorithm.

- DeleteNode: If we want to delete node $n$, we adjust the supply voltage of the node child ($n.leftchild$ or $n.rightchild$) so that it is compatible with node $n$'s parent. If the children are in the same situation (both compatible or both not compatible), we randomly choose one of them.
- InsertNode: If node $n$ is to be inserted into the subtree which exits compatible nodes, it will be placed to join the longest side of the compatible nodes. We insert node $n$ to abut a node in the subtree that has the same voltage, clustering as many as possible. If there does not exist any node compatible, we randomly choose one place to insert.

Since the subtree construction is just a method to increase the possibility in forming a good voltage island property, we need a property checking function to check if there exists a favorable voltage island shape. We do it after the contour updated to make sure the voltage island property is acceptable. Our checking is efficient since we only check blocks on the segment of contour line replaced by the new added block, not all of the blocks in one voltage island.

B. Floorplanning with Performance Constraints Blocks in Voltage Island Architecture

Traditional floorplanners/placers minimize total wirelength but they cannot 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 [10], [13] 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 $(x_t, y_t)$, their locations are the corner points as far as possible, and the delay of the net $D_{s,t} = |x_t - x_s| + |y_t - y_s|$, where $d$ is a constant to scale the distance.

When the bounding box has a maximum bounded distance $D_{max}$, the general maximum delay for the distance from the source to the sink is:

$$\min_{(x, y) \in B_{max}} |x - x_s| + |y - y_s|$$

By applying the above idea to a set of performance constraint blocks (whose areas are $A_i, i=1,2,..,k$), they can get some rectilinear super blocks that the width $W_{perf}$ and height $H_{perf}$ satisfy the performance constraint: $W_{perf} + H_{perf} = B \leq D_{max}$, where $D_{max}$ is the maximum bounded distance. Among the placements (rectilinear super blocks) meeting the performance constraint, they pick the one with the minimum dead space $S_{perf} = W_{perf} \times H_{perf} - \sum A_i$ and fix the rectilinear block (and thus fix the delay) for further processing with other blocks. Using the pre-clustered shape-fixed appropriate rectilinear block, they guarantee that the performance constraint will be satisfied throughout the remaining processing. In [10], a method to dynamically adjust $W_{perf}$ and $H_{perf}$ into the rectilinear super blocks is used.

We then use a B*-tree algorithm using voltage island architecture. The performance of each block does not vary with supply voltage. The legal supply voltage has big impact on the driving strength, thus the bounding box size. If signals are communicated by high supply voltage, the bounding box for performance constraint blocks will be the largest. In addition, allowable box size should be a function of the supply voltage. In this paper we use the conservative modeling by using largest bounding box without performance blocks.

Based on the above discussion in the setup of enclosing bounding box of performance constraint blocks, our approach combines the advantages of the two methods in [13], [10], keeping 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 delay and wirelength can be optimized. This is further verified in the condition that supply voltages of the performance constraint blocks are possibly different. If we tighten the shape of performance constraint blocks at the beginning, we may be forced to raise the supply voltages of some of them to the higher one to meet voltage island property; or we will get a disorder B*-tree structure that the voltage property is withered. When the temperature becomes lower in annealing process, we do not allow a solution that violates performance constraints even if it has a better cost, the best solution will be kept until next feasible solution with better cost.

C. The Algorithm

Our floorplanning/placement design algorithm is based on the simulated annealing method and we only consider hard modules in this paper. 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 situations we discussed increase probability to be allowed to do swap, while other situations that wither the subtree property have lower probability.)
- $Op_5$: Move a block to another place. (new DeleteNode and InsertNode)

The first 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 for performance constraint blocks. In $Op3$, we flip a block. It can be applied to any blocks including performance constraints blocks, since we do not bind them together at the beginning. $Op1$ and $Op5$ change the relations of blocks to get a different placement and B*-tree structure based on our heuristics.

IV. 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 run like original B*-tree simulated annealing method, and will not spend extra runtime to execute voltage island property. For testing the performance in voltage island generation on large number of blocks, we apply our approach on some of the MCNC benchmarks with more blocks, and compare with [2], the original B*-tree with simulated annealing method. For adopting voltage island architecture, power routing cost and level converter issues should be addressed. We simplified the cost of power routing/overhead area by using the scaled boundary length of voltage islands except for the boundary side of the chip. The scaling is based on the amount of level converters necessary in the chip.

Table I shows the comparison between [2] and our approach on power consumption and power routing cost, where the power consumption in column 5 is lowest since we use the lowest available voltages for every block in it. From Table I, we can see that our power consumption is a little more than the lowest power listed in column 5, but our routing/level converters cost is about 16.4% - 55.2% less when compared with [2]. Fig.4 shows the comparison between two floorplans of a4mi33, with and without voltage islands generation heuristic.

In order to compare our results with [2] and [13] in similar number of voltage islands and power routing cost, we apply an intuition heuristic that assigns supply voltage of the blocks from original B*-tree results. For a floorplan generated from [2], first we raise the supply voltage of the block that is surrounded by the blocks applying the highest voltage. The reason is because a block that is assigned the highest voltage means that its voltage can not be changed to other voltage levels, so we modify other blocks to fit

1Level converters are only needed when the signals are transmitted from low supply island to high supply island.
this kind of blocks. This process is continued until all the blocks applying highest voltage are connected together to be an island. We then take care of the blocks applying second higher voltage, until all voltage level are considered. In Table II, we can see that, with almost the same number of voltage islands, at least 10% - 20% power consumption can be saved by our method, not to mention the good shape of the generated voltage islands.

Table III shows the comparison of our results with [13] which considers only alignment and performance constraints. Both methods meet performance constraints but our approach could get much lower cost of level converters with slightly more power consumption. Fig.1 illustrates final floorplanning result of ami49 with performance constraints blocks 5, 6, and 7, and they are not on the same voltage island.

V. CONCLUSION

In this paper, 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.

TABLE I

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Table</th>
<th>Original B*-tree [2]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami33</td>
<td>pt3-1</td>
<td>1.174 1.47%</td>
<td>1.181 1.3%</td>
</tr>
<tr>
<td></td>
<td>pt3-2</td>
<td>36.5 36.8%</td>
<td>208 5.97%</td>
</tr>
</tbody>
</table>

TABLE II

The comparison between power amount that need to be raised to form a floorplan with voltage islands. This shows our approach can obtain lower power via voltage island methodology.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Table</th>
<th>Lowest</th>
<th>Ours</th>
<th>Original [2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>hp</td>
<td>pt2</td>
<td>8.95 1.4%</td>
<td>8.37 1.3%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pt3</td>
<td>75.4 78.3%</td>
<td>97.8 23.8%</td>
<td></td>
</tr>
<tr>
<td>ami33</td>
<td>pt3-1</td>
<td>113.6 123.2%</td>
<td>136.8 20.4%</td>
<td></td>
</tr>
<tr>
<td>pt3-2</td>
<td>131.1 136.3%</td>
<td>161.7 23.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ami49</td>
<td>pt3</td>
<td>147.1 151.5%</td>
<td>171.4 16.5%</td>
<td></td>
</tr>
<tr>
<td>pt3-1</td>
<td>183.1 196.4%</td>
<td>239.6 30.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>208</td>
<td>222.9 7.2%</td>
<td>254.3 22.3%</td>
<td></td>
</tr>
</tbody>
</table>

Table I shows the comparison between power consumption and power routing cost. With both meeting performance constraints, our approach obtains much lower power routing cost with slightly more power consumption.

Fig. 4. Two floorplans of circuit ami33 with 3 usable supply voltage. (a) Floorplan with much higher power routing complexity since the number of voltage islands is large. (b) Floorplan with nice voltage island property (slightly more power dissipation).

REFERENCES