Net Balanced Floorplanning Based on Elastic Energy Model

Wei Liu and Alberto Nannarelli
Dept. of Informatics & Mathematical Modelling
Technical University of Denmark
Kongens Lyngby, Denmark

***Abstract*—Floorplanning is becoming more and more important in VLSI design flows, especially for System-on-Chip (SoC) designs where IP blocks dominate standard cells. Moreover, in deep sub-micron technologies, where process variations can introduce extra signal skew, it is desirable to have floorplans with balanced net delays to increase the safety margins of the design.**

In this paper, we investigate the properties of floorplanning based on the elastic energy model. The B*-tree, which is based on an ordered binary tree, is used for circuit representation and the elastic energy is used as the cost function. To evaluate how well a net is balanced, we introduced a new metric 'Unbalancing'. A more balanced net would have a smaller 'Unbalancing' value. Experimental results show that our approach can not only meet fixed-outline constraints, but also achieve significant improvements in net balance for all the circuits in the MCNC benchmark.

I. Introduction

The System-on-Chip (SoC) design methodology is being used more and more widely in recent years. With technology scaling, more functionalities can now be put onto a single chip. To reduce design complexities, these functionalities usually appear as Intellectual Property (IP) macros. According to the International Technology Roadmap for Semiconductors (ITRS), the number of IP macros in a typical design in 2006 was almost 600, and it is still rising.

The impact of the growth in macros is that we are rapidly moving from ICs with a sea of standard cells and a few macros, to ICs with a sea of macros and a few areas of standard cells that implement custom designed, or glue logic in between the macros.

Therefore, when we are dealing with a "sea of macros", the situation is quite different from traditional design. Because macros, instead of standard cells, become dominant, a bad floorplan will not be compensated for in later steps when macros take more than half of the circuit area.

For these reasons, floorplanning becomes very important in the design flow because it is no longer possible to bring design closure through optimization in place-and-route alone.

In addition, manufacturing processes also pose new challenges like fixed-outline constraints and process variations. As a result, more and more effort should go to floorplanning.

In this work, we investigate the floorplanning problem under these new requirements and constraints. In [2], the authors proposed an energy-based approach, which models relations between blocks as net energy and overlap energy. The optimization objective is to minimize the overall energy. However, having overlaps means a larger search space and extra computations to remove overlaps. We adopted the net energy model and incorporate it with the B*-tree representation proposed in [4], which guarantees overlap-free floorplanning. Then, we developed an algorithm based on simulated annealing, which perturbs the floorplan in a more random manner than the original method used in [2]. In addition, our algorithm can handle both hard and soft blocks within a multi-dimensional constraint space.

The elastic energy model has the potential of finding better balanced net structures. A more balanced net has less differences in length between wires and thus less differences in delay. We observe that a more balanced net could improve performance and robustness. An example is shown in Fig. 1. The original net structure is shown in the left. $Source$ and $Sinks$ are terminals of blocks that communicate with each other. The $source$ type terminal sends out signal and the $sink$ type terminal receives signal. As can be seen in the original net structure, the path from $Source$ to $Sink2$ is the longest and defines the largest wire delay. A more balanced structure of the same net is shown in the right. $Sink2$ is pulled towards $Source$ in order to reduce the elastic energy. Consequently, $Sink3$ and $Sink4$ are pushed away slightly due to the movement of blocks. Although the total wirelength might stay the same or increase, differences in signal delay among wires become smaller. The largest wire delay is reduced in length, and as a result the clock cycle could be shorter. In process technologies where designers need to cope with process variations, a system with balanced nets would have less signal skew and be more robust to process variations. In the above example, we could maintain the original cycle time and any variations in wire delay not exceeding the safety margin would be tolerated in the optimized design. To evaluate how balanced a net is, we introduce a new metric 'unbalancing' to measure the difference in length between each wire within a net.

Experimental results show that our algorithm can generate floorplans with more balanced net structures (improvements from 10% to 30% in balancing) with a small increase in the total wirelength (less than 10%) over traditional algorithms.

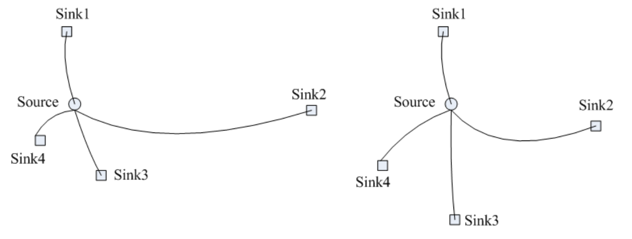


Fig. 1. Optimization results in more balanced net

II. Related Work

The floorplanning problem is NP-hard and many algorithms and heuristics have been proposed in the past decades. Early algorithms [3] use binary trees to represent the relation between modules positions and can only handle slicing floorplans.

In [5], the authors proposed an ordered pair of module name sequences called sequence-pair (SP), which was a breakthrough for representing non-slicing structures. However, the method still has a huge search space of size $(n!)^2 8^n$ and it cannot handle soft and rectilinear modules directly.

In [6], the authors proposed another non-slicing structure representation based on the ordered tree, the O-tree. The O-tree has a smaller search space than the sequence pair and it only requires linear time to transform to its constraint graph. The disadvantage of the O-tree is that it has an irregular tree structure and this makes primitive operations such as insert-node and delete-node, inefficient.

In [4], the authors proposed a B*-tree structure, which is based on ordered binary tree. For any B*-tree there is a unique corresponding placement, and, according to the B*-tree definition, the corresponding placement is overlap-free. These features reduce its search space. The B*-tree minimized the gap between representations for slicing and non-slicing floorplans, therefore, making it easier to implement and operate. The B*-tree representation will be reviewed in Section 3.

Many other representations [8],[9],[10],[11] also exist, some of them are extended to improve the runtime or solution quality. In [13], a method based on floorplan slack is proposed to meet fixed-outline constraints. Based on this, the authors in [14] proposed a method that can manage large numbers of mixed-size blocks through unifying partitioning and floorplanning. In [12] a similar multilevel approach was proposed based on the B*-tree representation.

Performance-driven floorplanning has also gained much attention. In [16], the authors proposed a postfloorplanning step to reduce the interconnect cost. In [15], the authors try to reduce leakage consumption since block positions have great impact on lateral heat conduction and thus affects temperature. Similarly, in [18] and [17] the authors try to reduce peak temperature while minimizing the performance loss.

Traditionally, floorplanning algorithms are block-oriented and use a combination of area and wirelength to evaluate a given solution. The half-perimeter wirelength (HPWL) method is widely used to estimate wirelengths. It computes the smallest rectangle that encompasses all terminals of a net and takes the sum of the width and the height of the rectangle as an estimation. A formal definition of the HPWL method is given below,

$$HPWL_x = \sum_{j=1}^{n}(max_i(x_{i,j}) - min_i(x_{i,j})) \tag{1}$$

$$HPWL_y = \sum_{i=1}^{n}(max_j(y_{i,j}) - min_j(y_{i,j})) \tag{2}$$

$$HPWL = HPWL_x + HPWL_y \tag{3}$$

where $x_{i,j}$ and $y_{i,j}$ are the coordinates of each terminal.

Some researchers proposed net-oriented methods as an alternative to block-oriented methods. In [7], the authors proposed a force-directed placement algorithm, which use a star model for the nets. In [1], the authors proposed a methodology based on net-clustering and the force-directed method. An energy-based model superior to the force-directed model is demonstrated in [2].

In [2], the wirelength minimization problem is addressed as a minimum (potential) energy problem. The energy associated with a block is the sum of the net energy (elastic) and the overlap energy. The net energy is analogous to the elastic potential energy in physics. Consequently in the floorplan, placing coupled blocks close to each other reduces the net energy. However, a natural consequence of the net energy-based placement approach is cell overlap [2]. Therefore, overlap energy is introduced to model how many modules cover each bin on the floorplan grid. A solution with uniformly distributed overlap energy would be overlap free.

In our algorithm, we adopt the net energy model and eliminate the overlap as discussed next.

III. Review of B*-trees and Energy-Based Model

In this section, we first give definitions of B*-trees and show examples on how to construct and operate on a B*-tree. Then, we review the elastic energy model proposed in [2]. After that, we introduce a new metric to measure how balanced a floorplan is.

A. Definition of B*-trees

For any module b_i, the left child in its horizontal B*-tree is the lowest unvisited module located on the right hand side and adjacent to b_i. The right child is the module located above and adjacent to b_i with its x-coordinate equal to that of b_i, and its y-coordinate less than that of the top boundary of the module on the left hand side and adjacent to b_i. To compact along the y direction, a vertical B*-tree can be constructed similarly with the requirement on left and right child exchanged. Both horizontal and vertical B*-tree can be constructed in linear time.

The root of the tree corresponds to the module on the left bottom corner. To construct a B*-tree for a given floorplan, we can use a Depth First Search (DFS) procedure recursively.

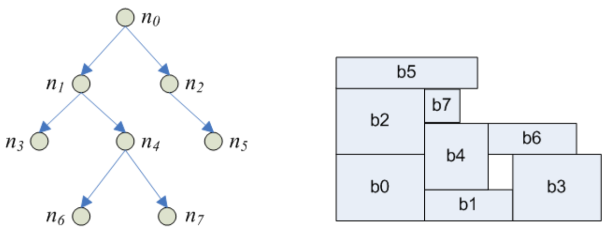


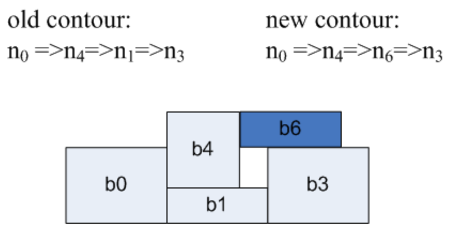
Fig. 2. A floorplan and its corresponding B*-tree

Fig. 2 gives a floorplan and its corresponding B*-tree. Module b0 is made the root since it's on the left bottom corner. Then we construct the left subtree for b0. Module b1 is made b0's left child. Likewise, module b3 is made module b1's left child. Since b3 does not have modules located to its left nor does it have modules located above with equal x-coordinate, the procedure goes back to construct the right subtree for b1. Thus, module b4 is made module b1's right child. The procedure continues until all modules are visited.

However, in a floorplanning algorithm we usually have a tree structure first and then find coordinates for all the modules. Similarly, starting from the root the process could proceed in a recursive fashion. According to the definition, we can have the following geometric relationship between a module and its left and right child.

$$node->left.x = node.x + node.width$$
$$node->right.x = node.x$$

To find the y-coordinate, a contour structure can be used to keep track of the current contour curve along with the traversal of all nodes. As shown in Fig. 3, the current contour is composed of n0,n4,n1 and n3 and node n6's y-coordinate needs to be determined. To place module b6 without overlap, its y-coordinate needs to be at least equal to module n3's y-coordinate. The contour is updated as well to reflect the new contour curve.



old contour:
$n_0 \Rightarrow n_4 \Rightarrow n_1 \Rightarrow n_3$
new contour:
$n_0 \Rightarrow n_4 \Rightarrow n_6 \Rightarrow n_3$
b4
b6
b0
b3
b1

Fig. 3. Updating y-coordinate for block b6

B. Operations on B*-trees

Primitive tree operations include insertion and deletion. Since we do not consider rectilinear modules in this work, insertion is made straightforward. A node can be inserted to any position in the tree and the node being replaced is made the newly inserted node's child. Deleting a node is a little more complex. Three scenarios could occur, namely, deleting a leaf node, deleting a node with one child and deleting a node with two children.

1) A leaf node can be deleted directly and it introduces no overhead.
2) If the node being deleted has only one child, the child inherits its parent's position in the tree. If the child is a left child and has its own children the procedure needs to be done recursively for the child as well.
3) If the node being deleted has two children, its left child replace its position and the procedure is again done recursively.

An example is shown in Fig. 4 where root node n0 is deleted. As the figure shows, in the worst case the time complexity for a deletion operation is the height of the subtree of which the root is the deleted node.

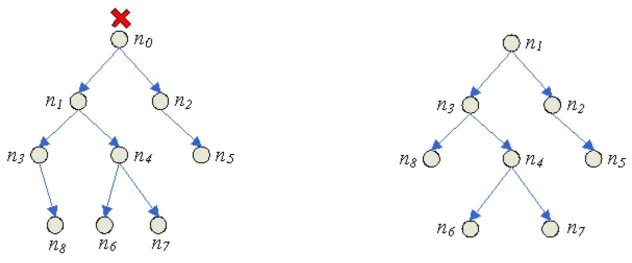


Fig. 4. Deleting root node n0

C. Energy-Based Model

As mentioned in Section 1, the energy-based model approach uses the net (elastic) energy to keep closely coupled blocks together and uses the overlap energy to remove possible overlaps between blocks. In our work, we borrow the idea of net (elastic) energy and discard overlap energy since the B*-tree structure can guarantee an overlap-free floorplan. Therefore, we will only give a description of the elastic energy model in this section.

In physics, the elastic force is conservative and it has a corresponding potential energy defined as $E = -kx^2$ where x is the deformation of the spring and k is the elastic constant of the spring. In [2], the authors similarly defined the net elastic energy as the quadratic distance between the net terminals. Therefore, the elastic energy of a block is the total elastic energy of all its terminals. The elastic energy of a floorplan is the total energy of all blocks.

Terminals within a net can be classified as source and sink. A net containing one source and four sinks is shown in Fig. 5. In the figure, the center of mass is the center of all sink type terminals. Source is attracted to the center of mass so that the signal delay to all sinks can be balanced. Therefore, the energy of a source is defined as the quadratic distance between the source and the center of mass (x_{med},y_{med}).

$$E(s) = (x_s - x_{med})^2 + (y_s - y_{med})^2 \tag{4}$$

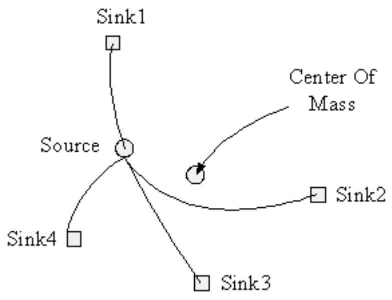


Fig. 5. A net containing one source and four sinks

For sinks, if the net contains only two terminals (one source and one sink) the energy is defined as the quadratic distance from the sink to the source.

$$E(t) = (x_t - x_s)^2 + (y_t - y_s)^2 \tag{5}$$

In all other cases, the model considers various contributions to the sink's energy and the following equation is used:

$$\begin{aligned} E(t) \quad = \quad & w_t[(x_t - x_{med})^2 + (y_t - y_{med})^2 + d^2(t, bbox) \\ & + 0.5(|x_t - x_s| + |y_t - y_s|)\sqrt{Abbox}] \end{aligned} \tag{6}$$

w_t is the net's weight. So net with a higher priority can be given a higher weight.

$d(t, bbox)$ is the distance between the sink and the bounding box of all other sinks. It is zero if the sink is within the bounding box. This term in the equation adds extra values to sinks that are not within the bounding box of all other sinks.

$Abbox$ is the area of the net bounding box. The last term is the Manhattan distance between the sink and the source multiplied by half the length of a square with the same area as the net bounding box. The value is linear with respect to the distance to the source.

D. Degree of UnBalancing

In this work, we assume a net is composed of one source and one or more sinks. Wires connect each sink to its source. To evaluate how balanced a net is, we introduce the new metric *unbalancing* (UB). For a given net, its degree of unbalancing is the sum of the absolute difference between the length of each individual wire and the average length of all these wires.

A formal definition is given as follows. WL_{ij} denotes the wirelength between the j_{th} sink and its source in net i. Note that the estimation is based on the HPWL method

$$WL_{ij} = (x_{ij} - x_{source_i}) + (y_{ij} - y_{source_i}) \tag{7}$$

Therefore, we can obtain the average wirelength in net i WL_{avg_i}

$$WL_{avg_i} = \frac{1}{\#sink_i} \sum_{j \in sink_i} WL_{ij} \tag{8}$$

The unbalancing (UB) of net i is the sum of absolute difference between WL_{ij} and WL_{avg_i} for each j

$$UB_i = \sum_{j \in sink_i} |WL_{ij} - WL_{avg_i}| \tag{9}$$

And the unbalancing of a floorplan is the sum of the UB_i of all nets in the floorplan

$$UB = \sum_{i \in net} UB_i \tag{10}$$

If $UB_i = 0$ for a given net, then it is perfectly balanced, i.e. all wires have same length. If $UB = 0$ for all nets in a given floorplan, then the floorplan is perfectly balanced.

Fig. 6 shows an example of an improvement in the degree of Unbalancing. The original net has a total wirelength of 23 and the degree of UB is 8.5. The optimized net's total wirelength increases by 2 units but the degree of UB significantly drops to 3.5. The benefit is that the difference in wirelength among wires in the same net gets smaller. If the delay in the wire connecting Source and Sink2 is increased due to process variations, then the original net is more vulnerable to timing failures than the improved net if clock cycle time in the two designs is the same.

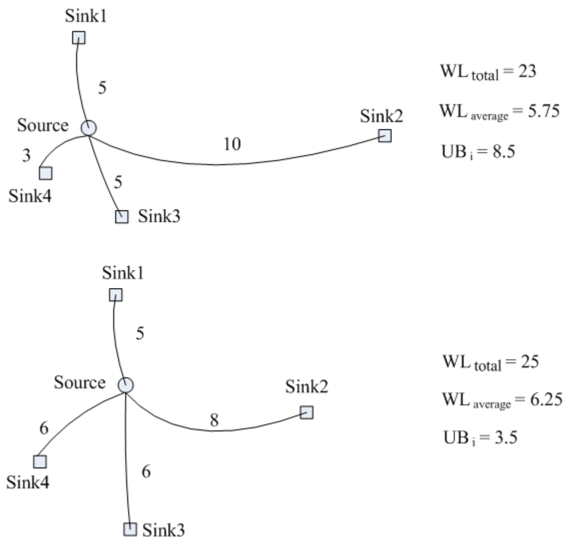


Fig. 6. Improvement in UB in a better balanced net

In Section 5, we use this metric to compare the results obtained from experiments using different cost functions and show that by using the elastic energy model a floorplan with better balanced nets can be obtained.

IV. Simulated Annealing

Our algorithm is based on simulated annealing. The basic process of simulated annealing is to transform a solution into a new one and to determine whether the new solution is accepted based on probability. The acceptance probability is related to a variable called $temperature$. Transformations are performed enough number of times at any given temperature and then it is decreased in a controlled process. In such a way, more solutions are accepted at high temperatures even if some of them are worse than the previous solution, while at low temperatures, the focus is more on local changes.

In floorplanning problems, a solution can be perturbed in a number of ways. Blocks could be moved to a different location. Pads could be moved along the boundary of the parent block. With its area being fixed, the width and height of a soft block are free to be changed under its aspect ratio constraint [4]. Therefore, soft blocks could have a different aspect ratio thus a new shape. The algorithm flow is given in Fig. 7.

```
simulated_annealing()
{
    initialize floorplan f;
    initialize Temperature;

    do{
        do{
            generate new floorplan g;
            update energy for each block;
            if(accept(f,g))
                f=g;
            if(g is the best ever seen)
                report g;
        }while(!thermal_equilibrium());

        decrease Temperature;
    }while(!stop());
}
```

Fig. 7. Program flow in the annealing process

During each iteration of the annealing process, we randomly assign new values to a set of variables to generate a new floorplan. To move a block to a new position, we need to generate three new values.

1) The block to be moved.
2) Its new parent in the B*-tree
3) Which side (left or right) of its parent it is inserted to.

If the block to be moved is a soft block, a new aspect ratio is generated and its dimension recalculated. Possible pad locations are modeled as slots in our algorithm. Therefore, to move a pad is simply to swap the contents of two slots. Thermal equilibrium is reached until enough iterations have been performed. When $temperature$ drops below a predefined threshold, the annealing process finishes.

To meet fixed-outline constraints, the $accept$ function first checks whether the new solution has an improvement in geometry violation either along its x dimension or its y dimension. Therefore, the floorplan will gradually transform towards the fixed-outline and eventually meets the constraint.

V. Experimental Results

The experiments are carried out for the Microelectronics Center of North Carolina (MCNC) building block examples, which is a benchmark widely used for testing floorplanning and placement algorithms [4], [5] and [6]. The benchmark contains five circuits with varying number of hard blocks, terminals, nets and IO pins. No soft blocks are provided in the benchmark. Table 1 summarizes the circuits characteristics of the benchmark.

TABLE I
Circuit Characteristics in MCNC benchmark

Circuits	#Blocks	#Nets	#Terminals	#IO Pins
hp	11	83	264	45
apte	9	97	214	73
xerox	10	203	696	2
ami33	33	123	480	42
ami49	49	408	931	22

Since previous works did not report actual block positions, there is no way of calculating the degree of unbalancing from their results. Therefore we compare the results of total wirelength (WL) and degree of unbalancing (UB) obtained from experiments using the same program but with different cost functions.

In the wirelength based approach, the cost function is solely the total wirelength, which for a given net is the half perimeter of the net's bounding box. Calculation is performed based on equations (1), (2) and (3).

In the energy based approach, for each block we first compute its elastic energy by adding up the energy associated with all its terminals according to equations (4), (5) and (6). Then we obtain a floorplan's energy by adding up all blocks' energy. The cost function is a floorplan's elastic energy.

Experimental results are shown in Table 2. The results contain all nets including power and clock grids. Pads positions are limited on the parent block's boundary only. The parent block's dimension is specified in the circuit specification. This is different from results reported by other works [4], [5] and [6] where they put pads along the boundary of the final floorplan's minimum bounding box.

Note that in the original MCNC benchmark circuit specifications, no information about a terminal's type (source or sink) is available. Therefore terminals' types are assigned randomly.

The results show that by applying our energy based algorithm, we obtain more net balanced floorplans, than those based on the wirelength model. The increase in total wirelength is not too large. In the case of *ami49*, which is the largest circuit, the degree of balancing improved by almost 20% while the increase in wirelength is only 3%.

VI. Conclusions

We have successfully implemented a floorplanning algorithm based on B*-tree structure and the elastic energy model. The algorithm can handle both hard and soft blocks and it places blocks and pads simultaneously. Moreover, it supports fixed-outline constraints. The resulting floorplans have more balanced nets than floorplans based on the wirelength model. This is more favorable in deep sub-micron technologies because signal skews are reduced, and, more importantly, the resulting floorplan is more robust against process variations since it provides larger safety margins.

TABLE II
EXPERIMENTAL RESULTS ON MCNC BENCHMARK

Circuits	Wirelength Based		Energy Based		Comparison	
	WL	UB	WL	UB	WL	UB
hp	253366	162992	269472	133374	9.56%	-32.86%
apte	614602	167922	637446	146465	3.72%	-12.78%
xerox	404278	359725	426468	324874	5.49%	-10.57%
ami33	96205	99286	105399	66664	6.36%	-18.17%
ami49	1070010	204111	1102538	163422	3.04%	-19.93%

REFERENCES

[1] S. Alupoaei and S. Katkoori. Net-based force-directed macrocell placement for wirelength optimization. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 10(6):824–835, December 2000.

[2] S. Alupoaei and S. Katkoori. Energy model based macrocell placement for wirelength minimization. In *VLSI Design, 2004. Proceedings. 17th International Conference on*, pages 713–716, 2004.

[3] Ralph H.J.M. Otten. Automatic Floorplan Design. In *Design Automation Conference, 1982. Proceedings 1982. 19th*, pages 261–267, 1982.

[4] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu. B*-trees: a new representation for non-slicing floorplans. In *Design Automation Conference, 2000. Proceedings 2000. 37th*, pages 458–463, June 2000.

[5] H. Murata et al. Rectangle-packing-based module placement. In *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*, pages 472–479, November 1995.

[6] P.-N. Guo, C.-K. Cheng, and T. Yoshimura. An o-tree representation of non-slicing floorplan and its applications. In *Design Automation Conference, 1999. Proceedings. 36th*, pages 268–273, June 1999.

[7] F. Mo, A. Tabbara, and R. Brayton. A force-directed macro-cell placer. In *Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on*, pages 177–180, November 2000.

[8] X. Hong. et al. Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan. In *ICCAD-2000.*, pages 8–13.

[9] J.-M. Lin, and Y.-W. Chang. TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans. In *DAC, 2001*, pages 764–769.

[10] H. Zhou, and J. Wang. ACG-Adjacent Constraint Graph for General Floorplans. In *ICCD, 2004.*, pages 572–575, October 2004.

[11] C. Zhuang, Y. Kajitani, K. Sakanushi, and L. Jin. An Enhanced Q-Sequence Augmented with Empty-Room-Insertion and Parenthesis Trees. In *DATE, 2002*, pages 61–68.

[12] H.-C. Lee, Y.-W. Chang, J.-M. Hsu, and Y,H.H. Multilevel floorplanning/placement for large-scale modules using B*-trees *Design Automation Conference, 2003. Proceedings*, pages 812–817, June 2003.

[13] S.-N. Adya, and I,-L. Markov Fixed-outline floorplanning through better local search. In *ICCD-2001.*, pages 228–334.

[14] J.-A. Roy, S.-N, Adya, and I.-L, Markov. Min-cut floorplacement. In *Computer Aided Design of Integrated Circuits and Systems, 2006. IEEE Transactions on*, vol.25, no.7, pages 1313–1326, July 2006.

[15] H.-D. Mogal, and K. Bazargan. Microarchitecture Floorplanning for Sub-threshold Leakage Reduction. In *DATE, 2007.*, pages 1–6, April 2007.

[16] C.-W. Sham, E.F.Y. Young, and H. Zhou. Optimizing Wirelength and Routability by Searching Alternative Packings in Floorplanning. In *ACM Transactions on Design Automation of Electronics Systems*, vol.13, no.1, article 21, January 2008.

[17] V. Nookala, D.J. Lilja, and S.S. Sapatnekar. Temperature-aware floorplanning of microarchitecture blocks with IPC-power dependence modeling and transient analysis In *ISLPED, 2006.*, pages 298–303, 2006.

[18] K. Sankaranarayanan, S. Velusamy, M. Stan and K. Skadron. A Case for Thermal-Aware Floorplanning at the Microarchitectural Level. In *The Journal of Instruction-Level Parallelism*, vol.7, October 2005.