A Fixed-outline Floorplanning Method Based on 2.5D

Sheqin Dong Qi Xie

Department of Computer Science and Technology

Tsinghua University, Beijing, 100084, P.R. China

Abstract

In this paper, a fixed-outline floorplanning algorithm based on 2.5D is proposed. By using constraints of area and number of pins to divide the modules into 4 layers, it confines the variations of the widths of the floorplans to a small region through common subsequence of sequence pair representation. The goal of minimizing the area is consistent with that of satisfying the given outline. A set of benchmarks is used for test. Experiments show that the proposed algorithm can achieve high successful probabilities in short run time, even with tight outlines and large aspect ratios are imposed.

Keywords: Fixed-outline floorplan, Floorplanning based on 2.5D , Common Subsequence

1. Intruduction

Floorplanning is a fundamental problem in the design process of complex chips and has a profound impact on the area, delay, power, and many other design parameters. Many representations for floorplans had been proposed in the last two decades, such as sequence pair [3], O-tree [4], CBL [5], B*tree [6] and BSG [7] etc.

*This work is supported by NSFC 90307005 and NSFC 60473126

Classical floorplanning is formulated as a free-die problem. It roughly determines the layout of a given set of modules subject to various objectives, such as area and estimated total wirelength. In reality, the use of floorplanning during the chip synthesis process almost always comes after the die size and package have been chosen. As pointed in [1], modern floorplanning formulation should be cast as a fixed-outline problem, and the packing must simultaneously achieve zero whitespace and zero overlap for the given outline.

It is also showed in [2] that, empirically, fixed-outline floorplan instances are significantly harder than those without fixed outline. Several works have been done addressed to it. [2] proposed an algorithm based on simulated annealing and sequence pair. It tries to achieve fixed-outline floorplanning by better local search, but the successful probabilities are quite low. [8] presented a genetic algorithm, which can achieve high successful probabilities with loose outlines. But no results of tighter outlines were reported.

On the other hand, interconnect optimization has become a major concern in modern floorplanning. Interconnect delay has dominated over gate delay and timing constraints have become more and more difficult to be met with this huge number of interconnects involved. 2.5D chip is a feasible solution to these problems. It has been shown that interconnect lengths can be greatly reduced in 2.5D ICs. A 2.5D

chip is actually a chip with more than one silicon layers to place modules.

In this paper, a fixed-outline floorplanning algorithm based on 2.5D is proposed. During simulated annealing, the 2.5D algorithm divides the input modules into 4 layers by adding the constraints of area and number of pins. Then the fixed-outline algorithm is called in each layer. It uses common subsequence and penalty function to bind the variations of the widths of the floorplans to a narrow range. Thus, a simple cost function which has no items about the heights of the floorplans is presented. Experiments show that the proposed algorithm can achieve high successful probabilities in short run time, even with tight outlines and large aspect ratios imposed. Also the proposed algorithm performs well both on 2.5D and fixed-outline.

2. Priliminaries

2.1. Sequence Pair and Common Subsequence

The concept of sequence pair was first introduced in [3]. A sequence pair is two sequences of all the module names. Horizontal and vertical constraint graphs can be derived from a given sequence pair by the following two rules:

1) (...*a*...*b*..., ...*a*...*b*...) => *a* is placed left to *b*;

2) (...*b*...*a*..., ...*a*...*b*...) => *a* is placed below to *b*;

Fig.1 gives an example of sequence pair.

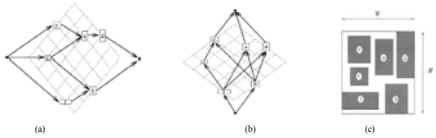

Figure 1. The (a) horizontal and (b) vertical constraint graphs of (ecadfb,fcbead), and (c) of its floorplan.

Definition 1. Given two sequences *X* and *Y*, a sequence *Z* is a ***common subsequence*** of *X* and *Y* if *Z* is a subsequence of both *X* and *Y*.

For example, assume a sequence pair P=(ecadfb,fcbead), "cad" is a common subsequence of P.

2.2. Sequence Pair and Common Subsequence

Definition 2. Given a modules set *B* and an outline with width of W_d, let $B_p=\{m_{p1}, m_{p2}...m_{pk}\}$ be a subset of *B*. If it holds that $\sum_{i=1}^{k} w_{pi} \le W_d$, then B_p is a ***primary modules set*** (PMS for short) of *B*. Let $\sum_{i=1}^{k} w_{pi}$ be the length of B_p and denote it as $L(B_p)$. Let $|B_p|$ be the number of modules in B_p.

Definition 3. Given a sequence pair $<\Gamma_+, \Gamma_->$ and a PMS B_p, if there exists a common subsequence of Γ_+ and Γ_-, which has all the modules in B_p, we call that $<\Gamma_+, \Gamma_->$ ***covers*** B_p.

For example, assume $B_p=\{b, d, e\}$ is a PMS of $\{a, b, c, d, e\}$. Since (*bdeac*, *abdce*) contains a common subsequence <*bde*>, which includes all the modules in B_p, we call that it *covers* B_p.

Given above definitions, the following lemma can be drawn:

Lemma 1: If a sequence pair *covers* a primary modules set B_p, the width of its floorplan is no less than $L(B_p)$.

3. The Fixed-outline Algorithm

The fixed-outline algorithm first randomly creates a sequence pair as the initial solution, and then selects a PMS B_p from the given sequence pair. Then during the simulated annealing process, adaptations are performed after perturbations to make all the solutions obtained also *cover* B_p. After the adaptations and perturbations, an

algorithm is called trying to find a better PMS if the fixed-outline constraint has not been satisfied yet. Following are details of the fixed-outline algorithm.

Problem Definition: Each module m_i is rectangular and defined by a (w_i, h_i), where w_i and h_i are the width and height of the module respectively. Given a set of modules $B=\{m_1,m_2,...m_n\}$ and their interconnections, fixed-outline mode floorplanning determines the layout of each module subject to various objectives (such as estimate wirelength, etc.) while satisfying that no modules overlap and it holds that $W_F \leq W_d$ and $H_F \leq H_d$, where W_d and H_d are the width and height of given outline, and W_F and H_F are those of the floorplan.

For a given set of modules with total area A_T and given *maximum percent of dead-space* γ, we construct a fixed outline with aspect ratio α by following equations:

$$W_d = \sqrt{(1+\gamma)A_T\alpha} \quad H_d = \sqrt{(1+\gamma)A_T/\alpha}$$

Cost Function: The cost function adopted in the fixed-outline algorithm is:

Cost =W1*area/totalArea+ W2*wdiff

Where area is the area of the floorplan, and totalArea is the sum of all the modules' area. While wdiff is defined like this as the penalty function:

wdiff=(width-CurrentCPLength)/ CurrentCPLength

Here width is the width of the floorplan, and CurrenCPLength is the length of current common subseqence.

Unlike the cost functions adopted in other algorithms, neither the height of the floorplan nor the excess of the height appear in (1). And the excess of width is only introduced as a penalty. The reason lies in that: since all the solutions obtained *cover* B_p, their width must be no less than L(B_p) (from Lemma 1); by adding penalty function, the widths of solutions are driven exactly to L(Bp). The criteria of selecting B_p (in Section 3.3), guarantees that $L(B_p)$ is very close to W_d. Thus we regard the width of the solutions as constant, the goal of minimizing the area is consistent with that of minimizing the height. Therefore, it is reasonable to use this simple cost function, which may make the optimization easier.

The Selection of PMS: Firstly, we can get the set S of all the modules which lie on the right edge of the floorplan using the property of the sequence pair. Then we can find a set of modules from the right edge of the floorplan to the left edge for each module in S. This is a path in the horizontal constraint graphs, also a common sequence of the sequence pair. As we have all the common sequences of the floorplan, it's easy to choose the best one. We only need to compare the length of the common sequences and be sure that it's shorter than the width of the outline.

After the fixed-outline algorithm randomly creates a sequence pair as the initial solution, the algorithm above will be called firstly to generate the initial PMS. If the algorithm fails, we will randomly choose only one module whose length is less than the width of the outline as PMS. Absolutely, such a floorplan won't satisfy the fixed-outline constraint. As pointed before, after the adaptations and perturbations, this algorithm is called trying to find a better PMS if the fixed-outline constraint has not been satisfied yet. Thus, time and time again, we will get the perfect PMS whose length is exactly close to the width of the outline. As we have the reasonable cost function, the fixed-outline constraint will surely be satisfied at last.

Perturbations and Adaptations: The perturbations adopted in the fixed-outline algorithm are the most general ones used in sequence pair representation, which are rotating a module, swapping two modules in the first sequence or in both sequences. Note that in order to maintain the length

of B_p, the modules in B_p are not allowed to be rotated in our algorithm.

In the following cases, the solution obtained might not *cover* B_p any longer: 1) to swap two modules in the first sequence and only one of them belongs to B_p; 2) to swap two modules in the first sequence and both of them belong to B_p; 3) to swap two modules in both sequences and only one of them belongs to B_p.

To make the solution *covers* B_p, adaptations should be taken after perturbations of the above cases. These adaptations can be done in O(n)-time. In the following, we will introduce the adaptation for the first case in detail. The other two kinds of adaptations are quite similar and omitted here for the sake of space.

Let $<\Gamma_+, \Gamma_->$ be the sequence pair which already have two modules, m_1 and m_2 ($m_1 \in B_p$, $m_2 \notin B_p$), swapped in the first sequence. Let $\Gamma(i)$ denote the i-th module in the sequence Γ.

1) Set sequence $BF=\varnothing$, scan Γ_- from left to right, and insert to BF the modules which belong to B_p. Let x denote the position of m_1 in BF and p_1 denote the position of m_1 in Γ_-;

2) Set $BF=\varnothing$, scan Γ_+ from left to right, and insert to BF the modules which belong to B_p. Let y denote the position of m_1 in BF and p_2 denote the position of m_1 in Γ_+;

3) If $x=y$, $<\Gamma_+, \Gamma_->$ already covers B_p, so the procedure terminates. Otherwise, if $p_1 < p_2$, goto 4; else goto 5;

4) For $i=p_1$ to p_2-1, set $\Gamma_-(i)= \Gamma_-(i+1)$. Set $\Gamma_-(p_2)= m_1$ and stop;

For $i=p_2+1$ to p_1, set $\Gamma_-(i)= \Gamma_-(i-1)$. Set $\Gamma_-(p_2)= m_1$ and stop;

4. The Algorithm Based on 2.5D

Before using the fixed-outline algorithm, we need an algorithm to group modules. We set the numbers of layers to 4. The details will be discussed next.

Constraint: We choose two constraints for our 2.5D algorithm. They are constraints of total area and total number of pins. As in the fixed-outline algorithm, we construct a fixed outline by using the total area of modules. So to be sure that each layer has the same fixed-outline, the total area of each layer should be more or less the same. We also choose the total numbers of pins. The 2.5D algorithm is firstly designed to decrease the interconnect lengths by adding interconnects of layers. In theory, the interconnects of layers will be bigger if the number of pins in each layer is more approximate.

Cost Function: We set our cost function as this:

Cost=|ADiff(0)|+…+|ADiff(3)|+|PDiff(0)|+…+|Diff(3)|。

Where ADiff(i) means the diff between total area of layer i and quarter of total area of all the modules in 4 layers. Similarly, PDiff(i) covers the diff of number of pins. And we treat the area constraint and the pins constraint equally. The cost function is simple but effective.

Perturbations: We generate the initial solution randomly. We choose the two layers whose total area diff most. Then randomly choose two modules for swap. If after the swap, the diff between two layers can be reduced by 10 percents at least, we will accept it. Otherwise, we have to choose another two.

5. Experimental Results

We implemented the proposed algorithm in C++ and run test data created from MCNC benchmarks on a PC with AMD Athlon™ 64 Processor 3000+ 1.81GHZ and 1.00GB RAM. The following experimental results are the averages

of 100 runs. Note that the modules handled in our experiments are all hard modules.

Table 1 shows that the proposed algorithm are quite promising in both successful probabilities and run time, even when the aspect ratio is as large as 6. Another advantage of the proposed algorithm is that high successful probabilities can be achieved with tight outline imposed. When γ=7% and 6%, the successful probabilities is 100%. When γ=5%, the successful probabilities range from 70% to 90%, which are still acceptable. Fig.2, 3 and 4 shows several resultant floorplans of 198(created from ami33) with γ=7% and aspect ratios of 1, 2 and 6 respectively. In the above experiments, the algorithm terminates once a feasible floorplan. Otherwise, we regard the algorithm failed.

6. Conclusions

In this paper, a fixed-outline floorplanning algorithm based on 2.5D is proposed. The fixed-outline floorplanning algorithm uses common subsequence of sequence pair. It tries to confine the variations of the widths of the floorplans to a small region through common subsequence. Thus the goal of minimizing the area is consistent with that of satisfying the given outline. While the 2.5D algorithm uses the concept of layers. SA is used by adding constraints of area and number of pins to divide the modules into 4 layers. A lot of standard benchmark files are used for testing. The results are promising.

7. References

[1] A.B. Kahng, "Classical Floorplanning Harmful?" *Proceedings of ISPD*, San Diego, 2000, pp. 207-213.

[2] Saurabh N. Adya, Igor L. Markov, "Fixed-outline Floorplanning Through Better Local Search", *Proceedings of ICCD*, Austin, 2001, pp. 328-334.

[3] H. Murata, K. Fujiyoshi, S. Nakatake et.al, "VLSI module placement based on rectangle-packing by the sequence pair", *IEEE Trans. on CAD*, 1996, vol. 15(12), pp. 1518-1524.

[4] Y. Pang, C.-K. Cheng and T. Yoshimura, "An Enhanced Perturbing Algorithm for Floorplan Design Using the O-tree Representation", *Proceedings of ISPD*, 2000, pp. 168-173.

[5] Xianlong Hong et al., "Corner Block List: An Effective and Efficient Topological Representation of on-Slicing Floorplan", *Proceedings of ICCAD*, 2000, pp. 8-13.

[6] Y.-C. Chang, Y.-W. Chang, G.-M. Wu and S.-W. Wu, "B*-Trees: A New Representation for Non-Slicing Floorplans", *Proceedings of DAC*, 2000, pp. 458-463.

[7] S. Nakatake, K. Fujiyoushi, H. Murata and Y. Kajitani, "Module Placement on BSG-structure and IC Layout Applications," *Proceedings of ICCAD*, 2000, pp. 8-13.

[8] Chang-Tzu Lin, De-Sheng Chen et al, "Robust Fixed-outline Floorplanning Through Evolutionary Search," *Proceedings of ASP-DAC*, 2004, pp 42-44.

Table 1. The results of 198 with γ=7%

Aspect ratio	α=1	α=2	α=3	α=4	α=5	α=6
Run time	4.9s	3.1s	3.7s	3.3s	4.1s	4.9s
Successful	100%	100%	100%	100%	100%	100%

Figure 2. The resultant floorplans of 198 in 4 layers with γ=7% and aspect ratios of 1.

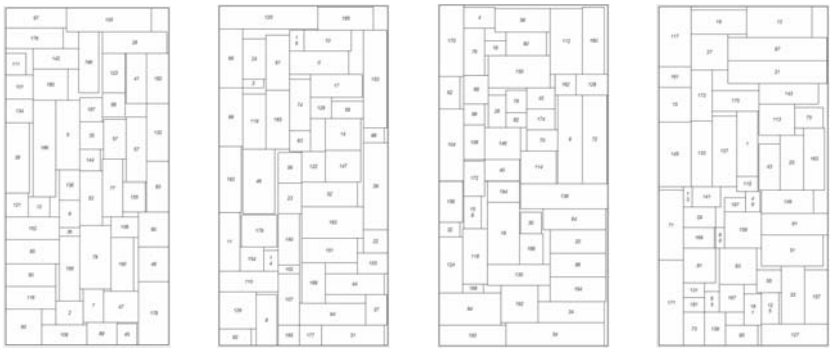

Figure 3. The resultant floorplans of 198 in 4 layers with γ=7% and aspect ratios of 2**.**

Figure 4. The resultant floorplans of 198 in 4 layers with γ=7% and aspect ratios of 6**.**