行政院國家科學委員會專題研究計畫　期中進度報告

子計畫一：極大型混合尺寸模組的平面規劃與擺置(1/3)

計畫類別：整合型計畫
計畫編號：NSC92-2220-E-002-013-
執行期間：92 年 11 月 01 日至 93 年 07 月 31 日
執行單位：國立臺灣大學資訊工程學系暨研究所

計畫主持人：楊佳玲
共同主持人：張耀文

報告類型：完整報告

處理方式：本計畫可公開查詢

中　華　民　國 93 年 5 月 28 日

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※　　　　極大型混和尺寸模組的平面規劃與擺置　　　　※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：□個別型計畫　　x 整合型計畫

計畫編號：NSC　93-2220-E-002 –001 -

執行期間：　　92 年　11 月　1　日至　93 年　7　月 31 日

計畫主持人：楊佳玲

共同主持人：張耀文

計畫參與人員：

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：台灣大學資訊工程學系

中　華　民　國　93　年 5　月 30 日

行政院國家科學委員會專題研究計畫成果報告
先進電子設計自動化技術研發
子計畫一：極大型混和尺寸模組的平面規劃與擺置（1/3）

計畫編號：NSC 93-2220-E-002 –001 -
執行期限：92 年 11 月 1 日至 93 年 7 月 31 日
計畫主持人：楊佳玲　　台灣大學資訊工程系
共同主持人：張耀文　　台灣大學電子工程研究所
e-mail: yangc@csie.ntu.edu.tw
ywchang@cc.ee.ntu.edu.tw
http://www.csie.ntu.edu.tw/~yangc
http://cc.ee.ntu.edu.tw/~ywchang/

一、中文摘要

由於奈米 IC 技術的持續進步，設計的複雜度以驚人的速度在成長，IP 模組被廣泛的重複利用，且大量的緩衝區塊(buffer block)被用來最佳化晶片的整體延遲。除此之外，競爭激烈的 IC 市場需要迅速及有效的工具來處理超大型的設計和對不同平面規劃的限制做最佳化。這些趨勢都使得快速、有效、可處理超大型混和大小模組的平面規劃工具成為必要。為因應超大型混和大小模組的平面規劃，在今年的計畫中，我們提出快速退火模擬法(fast simulated annealing)來處理大型的電路。我們也利用快速退火模擬法來處理各種平面規劃的限制，顯示出快速退火模擬法的優越性。

關鍵詞：平面規劃，擺置，多層架構，奈米技術 模擬退火法

Abstract

Design complexities are growing at a breathtaking speed with the continued improvement of the nanometer IC technologies. On one hand, designs with one billion transistors are even expected within this decade, IP modules are widely reused, and a large number of buffer blocks are used for delay optimization as well as noise reduction in nanometer interconnect-driven floorplanning/placement, which all drive the need of a tool to handle ultra large-scale mixed-size modules/cells. On the other hand, the highly competitive IC market requires faster design convergence, faster incremental design turnaround, and better silicon area utilization. Efficient and effective hierarchical design methodology and tools capable of placing and optimizing ultra large-scale mixed modules and cells are essential for such large designs. In this year's report, we proposed a fast simulated annealing (Fast-SA) to handle the ever growing circuit size. We also use the Fast-SA to handle different modern floorplanning constraints to demonstrate the effectiveness and the efficiency of our Fast-SA method.

Keyword: fixed-outline floorplanning, placement, multilevel framework, nanometer technology, simulated annealing

二、Introduction & Objective

In 1983, Kirpatrick, Gelatt, and Vecchi observed the analogy between a combinatorial optimization problem and the problem to determine the lowest-energy ground state of a physical system with many interacting atoms [10]. They generalized the basic approach by introducing a multi-temperature approach in which the temperature is lowered slowly in stages. At each temperature, the system is simulated by Metropolis' procedure until the system reaches equilibrium. This is so-called simulated annealing. Since then, the simulated annealing technique has been very successfully applied to many optimization problems. For VLSI design automation, simulated annealing is widely used in floorplanning, placement, routing, etc. Due to its popularity and usefulness, several variants of simulated annealing have been proposed in the literature. For example, Otten and van Ginneken in [14,15] and Wong, Leong, and Liu in [17] studied the mechanism of simulated annealing in great depth by applying annealing to floorplan design. However, the excessive running time is a significant drawback of the simulated annealing process. Some adaptive temperature schedule techniques are also introduced to simulated annealing process to save iterations [1,16]. Other possibility is using a better perturbation method to perform a better local search [4]. This kind of methods is confined by the given problem and is thus not general. Recently, Hentschke mixed random perturbations and greedy perturbations to improve simulated annealing based placement [7], but the greedy perturbation scheme is not popular for those applications. In this report, we propose a fast simulated annealing scheme, called Fast-SA, to improve the annealing schedule for better solution convergence speed and stability. It consists of three stages of temperature modification. The fist stage is like a random

search. The second stage is a pseudo greedy local search. Finally, the third stage is a hill-climbing stage to avoid bogging in a local minimum in the second stage. Based on the B*-tree floorplan representation [5], three types of modern floorplanning problems are examined to demonstrate the effectiveness and efficiency of Fast-SA: (1) classical floorplan design, (2) fixed-outline floorplanning, and (3) position constrained (e.g., bus-driven) floorplanning. For classical floorplan design, simulations show that Fast-SA achieves an average speedup of 17 times over classical SA. To cope with fixed-outline floorplanning, we extend the Fast-SA to Adaptive Fast-SA to demonstrate the effectiveness of the new annealing scheme. The Adaptive Fast-SA controls the parameters of the desired aspect ratio, area, and wirelength dynamically according to the success rate of fitting into the given floorplan outline for the 500 most recent floorplan solutions. Experimental results show that our method achieves an average success rate of 100% (99.7%) for fixed-outline floorplanning with a dead space of 15% (10%) and various aspect ratios, compared to average success rates of 78% and 85% obtained by the recent works [2,3,4] and [11], respectively. In particular, even though fixed-outline floorplanning is considered much harder than classical outline-free floorplanning, our B*-tree based Fast-SA for fixed-outline floorplanning achieves floorplanning quality of as good as classical outline-free floorplanning. Floorplanning with position constraints are prevailing in modern floorplan design. There exist many types of position constraints in modern floorplanning, such as bus, range, alignment, symmetry constraints. To test Fast-SA for position constrained floorplanning, we consider the most recently addressed bus-driven floorplanning [18]. Compared with the most recent work by Xiang et al [18], our method can obtain bus-constrained floorplans of less 20% of dead space on average by using much less CPU time for hard modules. When coping with soft modules, our method can even reduce 55% of dead space on average. We note that the new simulated annealing schemes developed in this report are general and thus can be applied to other optimization problems.

三、Research Techniques

1. Simulated annealing

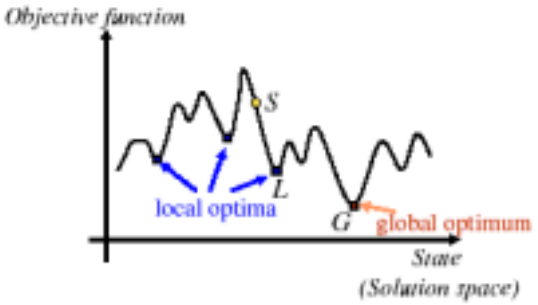

Figure 1. The analog of the solution space as a one-dimensional hills and valleys.

Simulated annealing is a widely used non-deterministic algorithm for solving combinatorial optimization problems. Every combinatorial optimization problem could be discussed in terms of a state space. A state is a configuration of the combinatorial objects involved. There are typically many configurations in a combinatorial optimization problem. Only some of these correspond to global optima. A greedy, iterative improvement method starts with a given state and examines a local neighborhood of the states for better solutions. The iterative method moves from the current state to one in the local neighborhood, if the latter has a better cost. If all the neighborhood solutions have inferior costs, the algorithm is said to have converged to a local optimum. In Figure 1, the states are shown along the x-axis, and it is assumed that two consecutive states are local neighbors. If the greedy, iterative method starts at an initial solution S, it could find a local minimum like L; however, it could not find the global minimum like G unless it ``climbs the hill." Simulated annealing is an algorithm with non-zero probability for ``up-hill" moves. The probability depends on the magnitude of the ``up-hill" movement and temperature (for modelling the search time---the lower the temperature, the longer the search time). The probability is defined as follows:

$$Prob(S \rightarrow S') = \begin{cases} 1 & \text{if } \Delta C \leq 0 \\ e^{-\Delta C/T} & \text{if } \Delta C > 0 \end{cases},$$

where $\triangle$C is the difference of the cost of the neighboring state and the cost of the current state, and T is the current temperature. Thus, at very high temperature, say T $\rightarrow\infty$, the above probability approaches 1. In contrast, when T $\rightarrow$ 0, the probability $e^{\{-\triangle C/T\}}$ approaches 0.

1.1. Floorplan Design Using Simulated Annealing

There are four basic ingredients for simulated annealing: solution space, neighborhood structure, cost function, and annealing schedule. In this report, we use the B*-tree representation to model a floorplan. Each B*-tree corresponds to a floorplan. Therefore, the solution space consists of all B*-trees with the given nodes (modules). To find a neighboring solution, we perturb a B*-tree to get another B*-tree by the following operations:

Op1: Rotate a block.

Op2: Move a node/block to another place.

Op3: Swap two nodes/blocks.

For Op1, we rotate a block orientation in a B*-tree node. It does not affect the B*-tree structure. For Op2, we delete a node and move it to another place in the B*-tree. For Op3, we swap two nodes in the B*-tree. After packing for the B*-tree, we obtain a new floorplan. Whether or not we take the new solution depends on the aforementioned probability which in turns depends on its cost function. The cost function is defined based on problem requirements. For example, we may adopt the following cost function to optimize the wirelength and area of a floorplan:

$$\Phi = \alpha \frac{A}{Anorm} + (1-\alpha)\frac{W}{Wnorm},$$ where

A is the current area, Anorm is the average area, W is the current wirelength, Wnorm is the average wirelength, and α controls the weight between area and wirelength.

1.2. Fast Annealing Schedule

The excessive running time is a significant drawback of the classical simulated annealing process. To reduce the running time of simulated annealing, we shall save the iterations. Choosing a smaller λ can definitely reduce the running time. However, if the temperature reduces too fast, then it is very likely to stop at a local optimum. To

integrate the random search and hill climbing more efficiently, we proposed a Fast Simulated Annealing (Fast-SA) process. The annealing process consists of three stages: (1) The high temperature random search stage, (2) the pseudo greedy local search stage, and (3) the hill climbing search stage. At the first stage, we let $T \rightarrow \infty$ so that the probability of accepting a worse solution approaches 1. The process is like a random search to find the best solution. At the second stage, we let $T \rightarrow 0$. Since the temperature is very low, we only accept a very small number of worse solutions. The behavior is like a greedy local search, but we still accept a small number of ``hill-climbing" solutions. We call this process as the pseudo greedy local search stage. The third stage is the hill climbing search stage. The temperature raises again to facilitate hill climbing. Thus, it can escape from the local minimum and search for better solutions. The temperature reduces gradually, and very likely it finally converges to a globally optimal solution. Since the new simulated annealing scheme saves many iterations to explore the solution space, it could devote more time to find better solutions in the hill climbing stage. This makes the annealing much more efficient and effective. To implement the annealing scheme, we define the temperature of the Fast-SA by the following equations:

$$T_n = \begin{cases} \frac{\Delta_{avg}}{\ln P} & n = 1 \\ \frac{T_1 \langle \Delta_{cost} \rangle}{nc} & 2 \leq n \leq k \\ \frac{T_1 \langle \Delta_{cost} \rangle}{n} & n > k \end{cases}$$

Here, $\triangle_{cost}$ is the average cost change, and n is the number of iterations. At the first iteration, the temperature is set according to the given initial accepting probability P. Since P is usually set close to 1, so it performs the random search to find a good solution. Then, it enters the pseudo greedy local search stage until k iterations. Here, c is a user-defined parameter to control how low the temperature is in the second stage. We usually choose a large c to make $T \rightarrow 0$ so that it almost only accepts good solutions to perform pseudo greedy searches. After k iterations, the temperature jumps up to further improve the solution quality. $<\triangle_{cost}>$ also affects the reduction rate of the temperature. If the cost of a neighboring solution changes significantly, $<\triangle_{cost}>$ is larger and thus the temperature reduces less. In contrast, if $<\triangle_{cost}>$ is smaller, it implies that the cost of the neighboring solution only changes a little; for this case, we reduce the temperature more to save iterations. The behavior of the temperature is illustrated in Figure 2(b). The number of iterations in the second stage can be determined by the problem size. The smaller the problem size, the smaller the k value. In our cases, we set c = 100 and k = 7 for floorplanning problems and it could handle tens to hundreds of blocks. Note that the initial temperature for the Fast-SA is the same as the classical SA, i.e., $T_1 = \triangle_{avg}/\ln P$. The initial temperature T_1 needs to keep high to avoid getting bogged in a local minimum in the very beginning.

2. Fixed-outline Floorplanning

The classical floorplanning formulation determines the layout of given blocks without given a fixed outline. Many previous works focused on minimizing floorplan area based on fully or partially topological floorplan representations, such as sequence pair [13], O-tree [6], B*-tree [5], TCG [12], CBL [8], etc. These floorplanners can easily obtain floorplan solutions with dead space far below 10%. However, for the top-down design of very large-scale ASICs and SoCs, a certain outline is usually given.A floorplan with pure area minimization without any fixed-outline constraints may completely useless because it cannot fit into the given outline. Modern floorplanning should be formulated as a fixed-outline (fixed-die) problem [9], and mixed macro and cell placement can be done through the fixed-outline floorplanning technique [3]. With the outline constraint, we shall optimize the given objectives, such as wirelength and/or area. The fixed-outline floorplanning was shown to be much more difficult than the outline-free floorplanning [4].

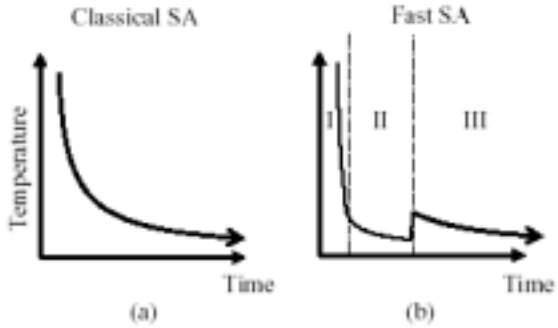

Figure 2. Temperature v.s. time for classical simulated annealing and Fast-SA. The Fast-SA consists of three stages.

2.1. Fixed-outline constraint

For a collection of blocks with the total area A and the given maximum percent of dead space γ, we construct a fixed outline with the aspect ratio R*, i.e., height/width. Since a floorplanner can change orientations of individual blocks, we choose $R^* \geqq 1$. The height H* and width W* of the outline is defined by the following equations [4].

$$H^* = \sqrt{(1+\Gamma)AR^*} \quad W^* = \sqrt{(1+\Gamma)A/R^*}$$

2.2. Algorithm Overview

We use our Fast-SA to search for a desired solution. For some blocks, they only have one feasible orientation to fit into the fixed outline. We mark all such blocks as non-rotatable blocks and set their orientations before performing perturbations. For Op1, we can only choose a rotatable block. Since we intend to minimize the wirelength/area of the floorplan, we always record the floorplan of the minimum wirelength/area during simulated annealing. After the temperature cools down enough, we terminate the simulated annealing process and return the best floorplan. Unlike outline-free floorplaning, we only modify the objective function to cope with fixed-outline flooplanning. We do not have to add any other operations for the perturbation, and Fast-SA is capable of finding feasible solutions efficiently.

2.3. Cost Function

In addition to the wirelength/area objective, we add an aspect ratio penalty to the cost function. The idea is that if the aspect ratio of the floorplan is similar to that of the outline, and the dead space of the floorplan is smaller than the maximum percentage of dead space γ, then the floorplan can fit into the outline. Assume that the current aspect ratio of the floorplan is R. We define the cost

function Φ by the following equation:

$$\Phi = \alpha A + \beta W + (1 - \alpha - \beta)(R - R^*)^2$$

where A is the current floorplan area, W is the current wirelength, R is current floorplan aspect ratio, and R* is desired floorplan aspect ratio. α, β, γ are user-defined parameters.

2.4. Adaptive Simulated Annealing

Since R* and λ are user-specified parameters, the weight between area and the aspect ratio should be determined by the given values. We do not know what α should be, and it is not feasible to try every α value in the cost function because it would spend too much time. So we use an adaptive method to control α according to n most recent floorplans found. The area weight α is defined by the following equation:

$$\alpha = \alpha_{base} + \left(1 - \frac{n_{feasible}}{n}\right)$$

where $n_{feasible}$ is the number of feasible solutions in n most recent floorplan solutions, and α_{base} is chosen by the user. From our experiments and observations, α_{base} = 0.5 is suitable for most cases. Once α is determined, the weight of the aspect-ratio penalty is also determined.

3. Bus-Driven Floorplanning

3.1. Problem Formulation

We consider a chip with multiple metal layers, and buses are assigned on the top two layers. The orientation of buses is either horizontal or vertical. The problem of bus-driven floorplanning (BDF) is defined as follows [18]: Given n rectangular macro blocks $B = \{b_i \mid i=1, ..., n\}$ and m buses $U = \{u_i \mid i=1,...,m\}$, each bus u_i has a width t_i and goes through a set of blocks B_i, where $B_i \subseteq B$ and $|B_i| = k_i$. Decide the positions of macro blocks and buses such that there is no overlap between any two blocks or between any two horizontal (vertical) buses, and bus u_i goes through all of its k_i blocks. At the same time, the chip area as well as the bus area are minimized.

3.2. Cost Function

We apply Fast-SA to B*-tree to find optimal solutions for bus-driven floorplanning. Since the objective function of bus-driven floorplanning is to minimize the chip area and the total bus area and we must satisfy all bus constraints, we define the cost function as follows:

Cost = α A + β B + γ M

where A is the chip area, B is the bus area, and M is the number of unassigned buses. α, β, and γ are user-specified coefficients.

3.3. Algorithm Overview

We initialize the B*-tree as a complete tree and start the Fast-SA process. The soft block adjustment is also an operation for perturbation. We perturb a B*-tree to another by the following operations:

Op1: Rotate a block.

Op2: Move a block to another place.

Op3: Swap two blocks.

Op4: Resize a soft block.

After each perturbation, we check if bus constraints are feasible and determine the locations of buses. We also check bus overlapping so that no two buses overlap each other. After evaluating the floorplan, the cost can be determined by the chip area A, bus area B of feasible buses, and the number of unassigned buses M. In the Fast-SA process, we record the floorplan solution with the most number of feasible buses and the smallest cost in the Fast-SA process. After the Fast-SA process stops, we report the smallest cost with the least number of unassigned buses. Thus, we can find the desired floorplan with the most feasible buses.

四、Bibliography

[1] E.H.L. Aarts, P.J.M. van Laarhoven, ``A new polynomial time cooling schedule," Proceedings of the ICCAD, 1985.

[2] S. N. Adya and I. L. Markov, ``Fixed-outline Floorplanning:Enabling Hierarchical Design", IEEE Trans. on VLSI Systems, vol 11(6), December 2003, pp. 1120-1135

[3] S. N. Adya and I. L. Markov, ``Consistent Placement of

Macro-Blocks using Floorplanning and Standard-Cell Placement", International Symposium on Physical Design (ISPD), pp. 12-17, San Diego, 2002.

[4] S. N. Adya and I. L. Markov, ``Fixed-outline floorplanning through better local search," Proceedings of the ICCD, pp. 328-334, 2001.

[5] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, ``B*-trees: a new presentation for non-slicing floorplans," Proceedings of the DAC, pp. 458-463, 2000.

[6] P.-N. Guo, T. Takahashi, C.-K. Cheng, and T. Yoshimura, ``Floorplanning using a tree representation," IEEE Transaction on Computer-Aided Design, pp. 281-289, 2001

[7] R.F. Hentschke, R.A.D.L. Reis, ``Improving simulated annealing placement by applying random and greedy mixed perturbations," Proceedings of 16th Symposium on Integrated Circuits and Systems Design, pp. 267-272, 2003.

[8] X. Hong, et al., ``Corner block list: an effective and efficient topological representation of non-splicing floorplan," Proceedings of the ICCAD, pp. 8-12, 2000.

[9] A. B. Kahng, ``Classical floorplanning harmful?" Proceedings of the ISPD}, pp. 207-213, 2000.

[10] S. Kirpatrick, C. D. Gelatt, and M. P. Vecchi, ``Optimization by simulated annealing," Science, pp.671-680, 1983.

[11] C.-T. Lin, D.-S. Chen, and Y.-W. Wang, ``Robust fixed-outline floorplanning through evolutionary search," Proceedings of the ASPDAC, pp. 42-44, 2004.

[12] J.-M. Lin and Y.-W. Chang, ``TCG: a transitive closure graph based representation for non-slicing floorplans," Proceeding of the DAC, pp. 764-769, 2001.

[13] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, ``Rectangle-packing based module placement," Proceedings of the ICCAD, pp. 472-479, 1995.

[14] R.H.J.M. Otten and L.P.P.P. van Ginneken, The Annealing Algorithm, Kluwer academic pulishers, Boston, 1989.

[15] R.H.J.M. Otten and L.P.P.P. van Ginneken, ``Floorplan design using annealing," Proceedings of ICCAD, pp. 96-98, 1984.

[16] J.M. Varanelli and J.P. Cohoon, ``A two-stage simulated annealing methodology," Proceedings of the Fifth Great Lakes Symposium on VLSI, pp. 50-53, 1995

[17] D.F. Wong, H.W. Leong, and C.L. Liu, Simulated Annealing for VLSI Design, Kluwer academic pulishers, Boston, 1988.

[18] Hua Xiang, Xiaoping Tang, and Martin D.F. Wong, `` Bus-Driven Floorplaning," Proceedings of the ICCAD, pp. 66-73, 2003.