# Analog Macro-Cell Placement with Very Fast Simulated Re-Annealing Algorithm*

ZHANG Li-hong[1], PEI Xian-deng[1], Ulrich Kleine[2]

[1](*National Storage System Laboratory*, *College of Computer Science and Technology*, *Huazhong University of Science and Technology*, *Wuhan* 430074, *China*);

[2](*University of Magdeburg*, *Magdeburg D*-39016, *Germany*)

E-mail: li_hong_zhang@hotmail.com

http://www.hust.edu.cn

Received July 17, 2001; accepted November 30, 2001

**Abstract:** This paper presents the analog macro-cell placement using the novel very fast simulated re-annealing algorithm, which is exponentially faster than the classical Cauchy or Bolzmann annealing. A slide function is applied to convert the absolute placement to relative placement, which greatly decreases the configuration space without degrading the searching opportunity. The cost function is set up deliberately to meet the special requirements of analog integrated circuits. Several net-length estimators are implemented which analog-circuit designers can choose with a trade-off between accuracy and efficiency. The global routing using the minimum Steiner tree is developed, which runs simultaneously during the placement configuration searching. This will ease the successive detail routing and greatly decrease the revise of final placement result. The layout example of optional amplifier is given to demonstrate its usability. The application of this algorithm drastically improves the design efficiency.

**Key words:** placement; simulated annealing; global routing; analog macro-cell; steiner tree

There is a clear tendency in the electronics market and particularly for Application-Specific Integrated Circuits (ASICs) to integrate complete systems on a single chip or a multichip module. The evolution to completely integrated systems and the inability to fully dispose of analog circuitry explain the booming market share of mixed signal ASICs nowadays. Although the analog circuits occupy only a small part of the area in these mixed signal ASICs, they require an inversely large part of the design time & cost and are often responsible for design errors and expensive design iterations. This is due to the knowledge-intensive nature and the larger number of freedom degrees in analog design compared with digital design[1]. Therefore it is necessary to develop effective algorithm and tools for the automated or computer-assisted design of analog integrated circuits[2].

Usually, the physical design process for integrated circuits is divided into the following steps: placement where the dimensions and locations of macro-cells are determined, terminal assignment where the locations of terminals on the macro-cell boundaries are set, global routing which assigns the routing regions for connecting the terminals

* Supported by the State of Saxony Anhalt and Siemens AG under Grant No.2577A/0027B (撒克林-安亥州和西门子公司资助)

**ZHANG Li-hong** was born in 1971. He is a Ph.D. candidate at Department of Computer Science and Technology, HUST. His research interests are layout optimization and VLSI computer-aided design tools. **PEI Xian-deng** was born in 1933. He is a professor and doctoral supervisor of the Department of Computer Science and Technology, HUST. His current research areas include computer storage system theory & technology, data storage technology and network multi-media storage technology. **Ulrich Kleine** was born in 1954. He is a professor and doctoral supervisor of the Department of Electrical Engineering, University of Magdeburg, Germany. His current research areas include analog and digital circuit design and design automation.

of the same net, and detailed routing which generates the actual geometric layout for the interconnections[3]. The first step of the integrated circuit design with the macro-cell design style is the decomposition of the circuit into a number of functional blocks, which are called “macro-cells” or “modules”. Each macro-cell is implemented using a “module generator”, which is anything that translates a behavioral or logic description of a cell into a symbolic or physical implementation. In many cases, the macro-cells are rectangular, although this is not always true and cells with any rectilinear shape can also occur. In general, the sooner a step is performed the more it impacts the final layout quality. If for example a very poor placement is generated, it can not be compensated in succeeding steps, no matter how well these are solved. In this sense the placement problem is more important than later steps of layout synthesis.

Simulated Annealing (SA) is a powerful and robust technique for the solution of difficult combinatorial optimization problems such as the optimal placement of components in integrated circuits. While this technique provides excellent quality solutions to layout problems, the CPU time required is very high. The Very Fast Simulated Re-Annealing algorithm (VFSRA) is used to statistically find the best global fit of a nonlinear non-convex placement cost-function over a multi-dimensional space. It is distinguished that this algorithm permits an annealing schedule for “temperature” $T$ decreasing exponentially in annealing-time $k$, $T=T_0\ \exp(-ck^{1/D})$. The introduction of re-annealing also permits adaptation to changing sensitivities in the multi-dimensional parameter-space. This annealing schedule, which is mathematically proved to suffice to give a global minimum, is faster than Cauchy Annealing (CA), where $T=T_0/k$, and much faster than Bolzmann Annealing (BA), where $T=T_0/\ln k$.

Although the placement problem has appealed to many researchers for a few years, most of work focused on digital circuits[7,11~13]. The work of Ref.[4,5] was based on analog circuits. The slicing structure was adopted for configuration construction in Ref.[4], which can not cover all the cell topologies. SA was applied in Ref.[5], however the used macro-cells were quite simple (mainly on the transistor level) so that much work was delivered to the succeeding detail routing. In this paper, we investigate the possibility of speeding up the analog macro-cell placement using VFSRA. Our algorithm is based on the characteristics of analog integrated circuits in the macro-cell design style. Analog Marco-cells are generated by a technology & application independent module generator[6], which themselves are sub-circuits with relatively concentrated functionality. So the pressure of detail routing becomes less and the placement is the key of the whole synthesis process. Differently from other work, the global routing is performed simultaneously during the placement so that the placement result can be accepted by detail routing without much adaptation. Our ultimate goal is to develop an algorithm that can generate an optimal routable placement for high-quality analog-circuit layout. The paper is organized as follows. First, the general SA is introduced and VFSRA is explained compared with CA and BA. Then the technique adopted for the analog placement application is described. Next, the global routing with the minimum Steiner tree is briefly introduced. Finally, the most relevant results are presented and the conclusion is drawn.

## 1 Very Fast Simulated Re-Annealing Algorithm

### 1.1 Generic simulated annealing algorithm

SA has been proposed by Kirkpatrick *et al.* as an effective method for the determination of global minima of combinatorial optimization problems involving many degrees of freedom[9]. Its basic feature is to explore possibly the configuration space of the optimization problem allowing hill climbing moves, i.e., the acceptance of the new configurations which increase the cost. These moves are controlled by a parameter, in analogy with temperature in the annealing process, and are less and less likely towards the end of the process.

Given a combinatorial optimization problem specified by a finite set of configurations or states $S$ and by a cost function $C$ defined on all the states $j$ in $S$, the SA algorithm is characterized by a rule to generate randomly a new configuration with a certain probability, and by a random acceptance rule according to which the new configuration is accepted or rejected. A parameter $T$ controls the acceptance rule. The generic structure of the algorithm is presented in Fig.1. Theoretical investigations of the SA optimization technique have been reported in some literatures[8].

```
SimulatedAnnealing
1   S=Initial solution S0;
2   T=Initial temperature T0;
3   while (stopping criterion is not satisfied)
4       while (not yet in equilibrium)
5           S'=Some random neighboring solution of S;
6           Δ=C(S')−C(S);
7           Prob=min(1,e^(−Δ/T));
8           if (random(0,1)<=Prob)
9               S=S';
10          endif
11      endwhile;
12      update T;
13  endwhile;
14  output the best solution;
```

Fig.1 Generic simulated annealing algorithm

### 1.2 Distinct SA approaches

The method of SA consists of three functional relationships:

(1) $g(x)$: Probability density of configuration-space of $D$ parameters, $x=\{x_i;i=1,\ldots,D\}$.

(2) $h(x)$: Probability density for acceptance of new cost-function given the just previous configuration.

(3) $T(k)$: Schedule of "annealing" the "temperature" $T$ in annealing-time step $k$, i.e., of changing the volatility or fluctuations of the two previous probability densities.

Different definition of $g(x)$, $h(x)$ and $T(x)$ provides various implementation approaches. The general two approaches are given before VFSRA is explained in this section.

Boltzmann annealing was generalized to apply more generally to fitting non-convex cost-functions arising in a variety of problems. Based on functional form derived for many physical systems belonging to the class of Gaussian-Markovian systems, the algorithm chooses

$$g(x) = (2\pi T)^{-D/2}\, e^{-\Delta x^2/(2T)} \tag{1}$$

where $\Delta x=x-x_0$ is the deviation of $x$ from $x_0$, and $T$ is a measure of the fluctuations of the Boltzmann distribution $g$ in the $D$-dimensional $x$-space. The acceptance probability is based on the chances of obtaining a new configuration $E_{k+1}$ relative to a previous configuration $E_k$,

$$h(x) = \frac{e^{-E_{k+1}/T}}{e^{-E_{k+1}/T}+e^{-E_k/T}} = \frac{1}{1+e^{\Delta E/T}} \approx \frac{1}{e^{\Delta E/T}} \tag{2}$$

where $\Delta E$ represents the "energy" difference between the present and previous values of the cost-function appropriate to the actual problem, i.e., $\Delta E=E_{k+1}-E_{k+1}$. Given $g(x)$, it has been proven that it suffices to obtain a global minimum of $E(x)$ if $T$ is selected to be not faster than

$$T(k) = \frac{T_0}{\ln k}. \tag{3}$$

For BA, if $T(k)$ is selected to be Eq.(3), then Eq.(1) gives

$$\sum_{k=k_0}^{\infty} g_k \geq \sum_{k=k_0}^{\infty} e^{-\ln k} = \sum_{k=k_0}^{\infty} \frac{1}{k} = \infty. \tag{4}$$

It is equivalent to the case that the products of probabilities of not generating a configuration $x$ for all

annealing-times successive to $k_0$ yield zero.

Specifically, it was noted that the Cauchy distribution has some definite advantages over the Boltzmann form[10]. The Cauchy distribution

$$g(x) = \frac{T}{(\Delta x^2 + T^2)^{(D+1)/2}} \tag{5}$$

has a "fatter" tail than the Gaussian form of the Boltzmann distribution, permitting easier access to test local minima in the search for the desired global minimum. The acceptance criteria can be the same as BA. It is instructive to note the similar corresponding heuristic demonstration, that the Cauchy $g(x)$ statistically finds a global minimum. If Eq.(3) is replaced by

$$T(k) = \frac{T_0}{k} \tag{6}$$

then

$$\sum_{k_0}^{\infty} g_k \approx \frac{T_0}{\Delta x^{D+1}} \sum_{k_0}^{\infty} \frac{1}{k} = \infty . \tag{7}$$

Note that the "normalization" of $g$ has introduced the annealing-time index $k$. The method of CA is thus statistically seen to have an annealing schedule exponentially faster than the method of BA.

For a $D$-dimensional parameter-space, different parameters have different finite ranges, fixed by considerations of actual application, and different annealing-time-dependent sensitivities, measured by the curvature of the cost-function at local minimal. BA and CA have $g$ distributions which sample infinite ranges, and there is no provision for considering differences in each parameter-dimension, e.g., different sensitivities might require different annealing schedules. Furthermore, there is no quick algorithm for calculating a $D$-dimensional Cauchy random generator. Consider a parameter in dimension $i$ generated at annealing-time $k$ with the range

$$\alpha_k^i \in [A_i, B_i] \tag{8}$$

calculated with the random variable $y^i$,

$$\alpha_{k+1}^i = \alpha_k^i + y^i (B_i - A_i)$$
$$y^i \in [-1,1] \tag{9}$$

Define the generating function

$$g_T(y) = \prod_{i=1}^{D} \frac{1}{2(|y^i| + T_i)\ln(1 + 1/T_i)} \equiv \prod_{i=1}^{D} g_T^i(y^i) \tag{10}$$

Its cumulative probability distribution is

$$G_T(y) = \int_{-1}^{y^1} \cdots \int_{-1}^{y^D} dy'^1 \cdots dy'^D g_T(y') \equiv \prod_{i=1}^{D} G_T^i(y^i)$$

$$G_T^i(y^i) = \frac{1}{2} + \frac{\mathrm{sgn}(y^i)}{2} \frac{\ln(1 + |y^i|/T_i)}{\ln(1 + 1/T_i)} \tag{11}$$

$y^i$ is generated from a $u^i$, a uniform distribution

$$u^i \in U[0,1]$$
$$y^i = \mathrm{sgn}(u^i - \frac{1}{2}) T_i [(1 + \frac{1}{T_i})^{|2u^i - 1|} - 1] \tag{12}$$

It is straightforward to calculate that for an annealing schedule for $T_i$

$$T_i(k) = T_{0i} e^{-c_i k^{1/D}} \tag{13}$$

a global minimum statistically can be obtained, i.e.,

$$\sum_{k_0}^{\infty} g_k \approx \sum_{k_0}^{\infty} [\prod_{i=1}^{D} \frac{1}{2|y^i|c_i}] \frac{1}{k} = \infty \tag{14}$$

It seems sensible to choose control over $c_i$, such that

$$T_{fi} = T_{0i} e^{-n_i/D}, \quad c_i = m_i e^{-n_i/D} . \tag{15}$$

It has proven fruitful to use the same annealing schedule for the acceptance function $h$ as used for the generating

function $g$. New parameters are generated from old parameters as

$$\alpha_{k+1}^{i} = \alpha_{k}^{i} + y^{i}(B_i - A_i) \tag{16}$$

constrained by

$$\alpha_{k+1}^{i} \in [A_i, B_i] \tag{17}$$

i.e., $y^j$ are generated until a set of $D$ are obtained satisfying these constraints.

Whenever doing a multi-dimensional search in the course of a nonlinear problem, inevitably one must deal with different changing sensitivities of the $\alpha^j$ in the search. At any given annealing-time, it seems sensible to attempt to "stretch out" the range over which the relatively insensitive parameters are being searched, relative to the ranges of the more sensitive parameters. This process is called re-annealing, which can be feasibly applied in VFSRA. It has proven fruitful to accomplish this by periodically rescaling the annealing-time $k$, essentially re-annealing, every hundred or so acceptance-events, in terms of the sensitivities $s_i$ calculated at the most current minimum value of $L$,

$$s_i = (A_i - B_i)\frac{\partial L}{\partial \alpha^{i}} \tag{18}$$

In terms of the largest $s_i$=$s_{\max}$, it has proven efficient to re-anneal by using a linear rescaling,

$$k_i' = \left(\frac{\ln[(T_{i0}/T_{ik})(s_i/s_{\max})}{c_i}\right)^{D} \tag{19}$$

$T_{i0}$ is set to unity to begin the search, which is ample to span each parameter dimesnion. The acceptance temperature is similarly rescaled. In addition, since the initial acceptance temperature is set equal to a trial value of $L$, this is typically very large relative to the global minimum. Therefore, when this rescaling is preformed, the initial acceptance temperature is reset to the most current minimum of $L$, and the annealing-time associated with this temperature is set to give a new temperature equal to the lowest value of the cost-function encountered to annealing-date.

## 2 Placement with VFSRA

Simulated annealing is more a general design methodology than a completely specified algorithm[11]. In order to apply this methodology in the placement of analog macro-cells, the problem should be formulated beforehand with care, i.e., describing the configuration of a placement, defining the neighboring configurations of a configuration, employing a suitable cost function and defining the annealing schedule.

### 2.1 Data for optimization

A placement solution is specified by the location of all macro-cells on the chip as well as their rotations and reflections from the standard orientations. Three types of random disturbance are used to locally modify a placement. They are:

- A horizontal or vertical translation of a macro-cell
- A 90° rotation in either direction
- A horizontal or vertical reflection

All the concerned data are classified into two groups: data related to macro-cells and related to nets. Among these data, some are used for optimization which are transferred to VFSRA while others are the description information of macro-cells and nets. Macro-cell is described by its location and orientation. Since in our applications macro-cells are rectangular, the bounding box and center coordinates are used for the description of their location. The coordinates $x$ and $y$ of each cell's center are given by two VFSRA parameters, whose range is [0, $L$]. $L$ is

$$L = \alpha \sqrt{\sum_{i=1}^{M} A(i)} \tag{20}$$

where $M$ is the number of macro-cells, and $A(i)$ is the bounding box area of the $i$th macro-cell, and $\alpha$ is a tuning factor. That is to say, all the modules are shifted within a $L*L$ square to find an optimal final configuration.

The orientation is not only helpful for obtaining a minimal total area, but also important in the search for an optimal net length. One VFSRA parameter gives the choice of orientation for one module. There are 8 different possible orientations for one macro-cell:

- the default orientation
- rotated by 90°
- rotated by 180°
- rotated by 270° (or –90°)
- mirrored relative to the $Y$-axis
- mirrored relative to the $Y$-axis and rotated by 90°
- mirrored relative to the $X$-axis (or to the $Y$-axis and rotated by 180°)
- mirrored relative to the $X$-axis and rotated by 90°(or to the $Y$-axis and rotated by 270°)

### 2.2 Cost function

The cost function is the goodness criterion of searched configurations. It is quite important as it includes all the interesting considerations and controls the procedure of algorithm searching. Our cost function consists of four parts, which are given in (21).

$$C = (\alpha_{all\_area}C_{all\_area} + \alpha_{N\_area}C_{N\_area} + \alpha_{P\_area}C_{P\_area}) + \alpha_{nets}C_{nets} + \alpha_{size}C_{size} + \alpha_{overlap}C_{overlap}. \tag{21}$$

The first is the area cost which is made up of the whole area and NMOS, PMOS region areas. They will help to decrease the whole area and make NMOS or PMOS region relatively concentrated. The second is the net-length cost, in which critical factors can be specified for each net. Some sensitive nets are set with the greater critical factors so that they keep as short as possible. In the analog integrated circuits, some nets, for instance, differential input, should keep shortest in order to decrease the parasitic capacitance. So the choice of critical factors is strongly designer-knowledge intensive which influences the final result greatly. The approaches used for net-length estimation will be discussed in Section 3. The third is the size cost, which is used to control the shape of the final layout. Designers can define the ideal width-length ratio or exact width value. The more the real layout shape differs from the ideal definition, the more penalty is brought about. The last is the overlap penalty cost. Since the slide function, which will be explained in the next section, can be optionally used in VFSRA, this penalty cost is only useful if the slide function is disabled during the algorithm process. It should be very strict because a configuration with overlapping cells is absolutely unacceptable. However it is better it exists, since a set of illegal placements probably could be used to reach a global minimum. In this sense a penalty function that measures the amount of macro-cell overlap is used to discourage overlaps. In order to inflict a much larger penalty to greater overlap, each overlap area between two cells is powered by square operation. The overlap cost is

$$C_{overlap} = \sum_{i=1}^{M} \sum_{j=i+1}^{M} [Area_{overlap}(i, j)]^2 \tag{22}$$

where $M$ is the sum of macro-cells. $Area_{overlap}(i, j)$ is the overlap area beween cell $i$ and cell $j$. Respectively $\alpha_{all\_area}$, $\alpha_{N\_area}$, $\alpha_{P\_area}$, $\alpha_{nets}$, $\alpha_{size}$ and $\alpha_{overlap}$ are the weight coefficient for the whole area cost $C_{all\_area}$, the NMOS region area cost $C_{N\_area}$, the PMOS region area cost $C_{P\_area}$, the net length cost $C_{net}$, the size cost $C_{size}$ and the overlap cost $C_{overlap}$. They are used to balance the importance of all the possible considerations. Different weight coefficients can be given to the four components according to the different requirements of designers.

### 2.3 Slide function

During the application of the VFSRA, a slide function is used to convert an absolute placement problem to a relative placement problem. In theory, neither the bottom-left placement [12] nor slicing-structure placement[4,13], both of which belong to relative placement, can substitute each other or cover all possible topologies. Although absolute placement has the advantage to cover every possible topology theoretically, the great complexity of multi-dimensional space in the placement problem frustrates the searching efficiency practically. So the slide function is applied which adapts the position of macro-cells after they are absolutely placed. During the adaptation, overlap among macro-cells is avoided so that the relative position instead of absolute coordinates becomes the ultimate focus. Its application makes full use of the advantages of both the absolute and relative placements. It can cover all the possible macro-cell topologies.

In order that the program could be more adaptive, much effort is done during the implementation. As an option, a critical macro-cell can be set as a fixed cell during the searching so that the rest of cells are placed in reference to it afterwards. This technique can decrease the configuration space without degrading the searching opportunities. As well, this is the purpose of the slide function, which is depicted in Fig.2. All the macro-cells are divided into two sets: *placedSet* in which the cells having been placed are stored and *unplacedSet* in which the cells not having been placed are stored. If a fixed cell exists, it is used as the reference cell. Otherwise the cell closest to the center of the minimal bounding-box, which encloses all the macro-cells, is set as the reference cell. In the beginning, *placedSet* only has one cell, i.e., the reference cell, and *unplacedSet* has the rest of cells. The cells in *unplacedSet* are stored regularly according to their distance to the reference cell. The closer cell precedes to be stored. Then from the closer cell to the farther cell, each cell in *unplacedSet* is shifted so that it finally unoverlaps with any cell in *placedSet*. One example is shown in Fig.3. Cell A is the fixed cell which is used for the reference cell. The number inside cell block is the slide index. For instance, cell B is shifted prior to cell C. Cell B has two possibilities to avoid the overlap with cell A: shifts upwards or leftwards. Since the shifted distance upwards is shorter than that leftwards, the direction with the shorter distance is chosen. Then cell B is removed from *unplacedSet* and merged into *placedSet*. Likely, cell C is shifted leftwards and the slide of cell D follows. Although cell D doesn't overlap with the original cell B, the slide should be based on the most current positions of all the cells in *placedSet*. So cell D should be shifted rightwards. The loop terminates if *unplacedSet* is empty and new positions of the cells in *placedSet* are output as the current configuration for the cost estimation.

```
MacrocellSlide
(placedSet: the set of the cells having been placed. unplacedSet: the set of the cells not having been placed.)
1     choose the reference cell among all the macro-cells;
2     placedSet = {the-reference-cell}, unplacedSet = {the-rest-of-cells};
3     sort unplacedSet (incrementally) according to the distance to the reference cell;
4     while (unplacedSet is non-empty)
5        choose the first component in unplacedSet as the current cell;
6        foreach (placedSet)
7           slide the current cell in order to avoid overlap with any cell in placedSet;
8        endfor
9        add the current cell into placedSet and remove it from unplacedSet;
10       update to sort placedSet (incrementally) according to the distance to the reference cell;
11    endwhile
12    output all the macro-cells in placedSet;
```

Fig.2 Slide of macro-cells

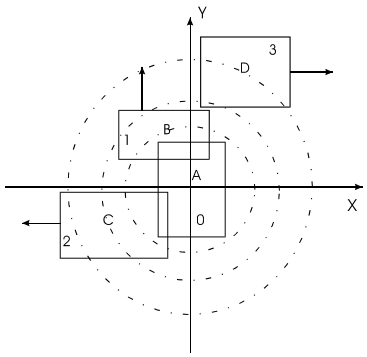



Fig.3 Example of four macro-cells' slide

## 3 Global Routing

During the net-length estimation, five approaches (half-perimeter, center-of-mass, complete-graph, minimum spanning tree and minimum Steiner tree) are implemented. Each involves a different trade-off between accuracy and efficiency [14]. As our application is about analog integrated circuits, the size of circuit is in general less than 20 modules and 30 nets [5]. Because the minimum Steiner tree is not only used for the net-length estimation, but also perform the global routing simultaneously, it is preferred so that the placement result is much closer to the reality and no much revise is needed for the detail routing. Furthermore the minimum Steiner tree finally outputs the interconnection Steiner point and path which will be directly respected and used afterwards so that the detail routing becomes quite simple in principle.

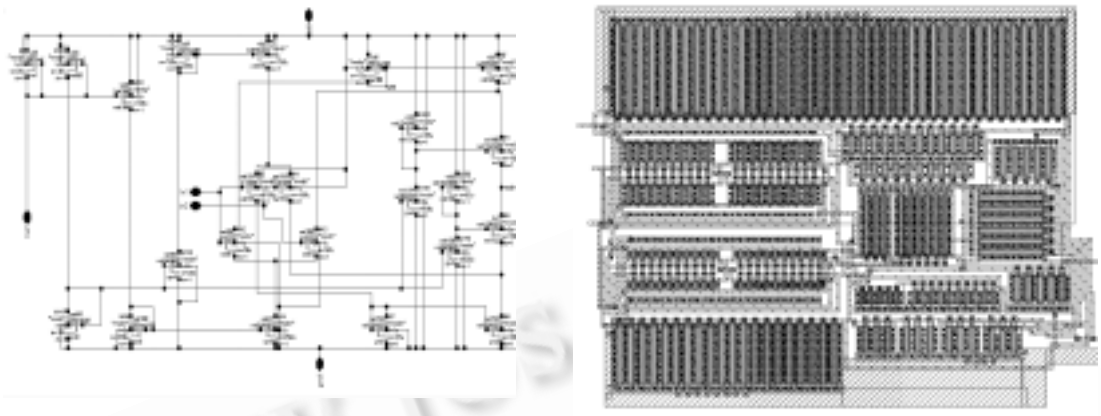
## 4 Experimental Results

The program is written in C++ under Solaris-UNIX system. A large number of experiments were run to tune and test the algorithm. The first series of experiments allowed us to set the parameters of the algorithm. Finally the initial temperature is set as 1. The temperature control coefficients $m_i$ and $n_i$ are $ln(1.0E-30)=-69$ and $ln(200)=5.29$ respectively. The weight coefficients are set as follows: $\alpha_{all\_area}=2$, $\alpha_{N\_area}=0.2$, $\alpha_{P\_area}=0.2$, $\alpha_{nets}=10$, $\alpha_{size}=5$, and $\alpha_{overlap}=50$. Then three circuits are used for the evaluation of the algorithm. The first circuit is a typical two-stage optional amplifier with 6 cells and 9 nets. The second is a rail-to-rail optional amplifier with 11 cells and 16 nets. The third is a common mode feedback optional amplifier with 15 cells and 17 nets. In order to demonstrate the efficiency of VFSRA, BA and CA were also implemented. Each algorithm was run for 10 times so that the mean and standard deviation ($\sigma$) are used for evaluation. The result is given in Table 1. The experimental result clearly shows that VFSRA is significant better than CA and BA on both the cost values and execution time. Moreover CA is noticeably better than BA, which explains why CA is widely used as the conventional SA approach solving the macro-cell placement problem.

**Table 1** Comparison among distinct SA algorithms

| | | BA | CA | VFSRA |
|---|---|---|---|---|
| Circuit 1 | $C_{mean}$ | 24 923 | 11 799 | 11 729 |
| | $C_\sigma$ | 1 289 | 176 | 123 |
| | $T_{mean}(s)$ | 1 320 | 846 | 688 |
| | $T_\sigma(s)$ | 340 | 210 | 21 |
| Circuit 2 | $C_{mean}$ | 178 761 | 86 653 | 83 844 |
| | $C_\sigma$ | 23 658 | 3 087 | 3 247 |
| | $T_{mean}(s)$ | 8 416 | 5 068 | 4 335 |
| | $T_\sigma(s)$ | 1 407 | 783 | 1 391 |
| Circuit 3 | $C_{mean}$ | 540 292 | 215 720 | 214 135 |
| | $C_\sigma$ | 82 931 | 13 554 | 8 317 |
| | $T_{mean}(s)$ | 1 942 | 1 143 | 842 |
| | $T_\sigma(s)$ | 314 | 4 | 29 |

In Fig.4(a) the schematic of the second circuit is depicted. The corresponding macro-cell placement with the final detail routing is shown in Fig.4(b). The layout of this amplifier is with a 0.8um CMOS technology. The simulated performance data of the amplifier are: power supply voltage 3.3V, open loop gain 66dB, transit frequency is 65MHz, slew rate 42.5V/us and power consumption 6mW.



(a) (b)

Fig.4 Schematic (a) and layout (b) of a rail-to-rail optional amplifier

## 5 Conclusion

In this paper the application of a novel very fast simulated re-annealing algorithm in the analog macro-cell placement has been presented. Two characteristics of this algorithm (faster convergence and re-annealing) are made full use of so that the minimum searching proceeds efficiently by changing sensitivities in the placement multi-dimensional parameter-space. The adoption of the slide function combines the advantages of both absolute and relative placements. The cost function is made adaptive, which allows designers to impose special analog-circuit constraints to control the placement result. Furthermore a few net-length estimation approaches are implemented including the minimum Steiner tree which processes the global routing. The simultaneous execution of placement and global routing instead of successive runs makes the result much closer to the reality without much modification afterwards. The example of optional amplifiers are given to show its usability.

**Acknowledgement** This work was done in the University of Magdeburg (Germany). It was partially sponsored by the state of Saxony Anhalt and Siemens AG.

**References:**

[1] Gielen, G., Debyser, G., Lampaert, K., *et al*. An analog module generator for mixed analog/digital ASIC design. John Wiley International Journal of Circuit Theory and Applications, 1995,23:269~283.

[2] Zhang, L., Kleine, U., Rudolph, T., *et al*. A new design rule description for automated layout tools. In: Sawan, M., ed. Proceedings of the IEEE Conference on Electronics, Circuit and System. Lebanon: IEEE Press, 2000. 988~992.

[3] Koh, H.Y., *et al*. OPASYN: a compiler for CMOS operational amplifiers. IEEE Transactions on Computer-Aided Design, 1990,9(2):113~125.

[4] Rijmenants, J., Litsios, J.B., Schwarz, T.R., *et al*. ILAC: an automated layout tool for analog CMOS circuits. IEEE Journal of Solid-State Circuits, 1989,24(2):417~425.

[5] Cohn, J.M., Garrod, D.J., Rutenbar, R.A., *et al*. KOAN/ANAGRAM II: new tools for device-level analog placement and routing. IEEE Journal of Solid-State Circuits, 1991,26:330~342.

[6] Wolf, M., Kleine, U. A novel design assistant for analog circuits. In: Taguchi, T., ed. Proceedings of the Asia and South Pacific Design Automation Conference. Tokyo: World Scientific Publishing Company, 1998. 495~500.

[7] Eschermann, B., Dai, W.M., Kuh, E.S., *et al*. Hierarchical placement for macrocells: a "meet in the middle" approach. In: Davis, N., ed. Proceedings of the International Conference on Computer-Aided Design. Los Angeles: IEEE Press, 1988. 460~463.

[8] Geman, S., Geman, D. Stochastic relaxation, gibbs distributions, and Bayesian restoration of images. IEEE Transactions on Pattern Anal. Machine Intelligence, 1984,PAMI-6:721~741.

[9] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. Optimization by simulated annealing. Science, 1983,220(4598):671~680.

[10] Szu, H., Hartley, R. Fast simulated annealing. Physical Letter, 1987,A122(3-4):157~162.

[11] Sechen, C., Sangiovanni-Vincentelli, A. The Timberwolf placement and routing package. IEEE Journal of Solid-State Circuits, 1985,20(2):510~522.

[12] Esbensen, H. A genetic algorithm for macro cell placement. In Geilen, T., ed. Proceedings of the European Design Automation Conference. Paris: Springer-Verlag Company, 1992. 52~57.

[13] Wong, D.F., Liu, C.L. A new algorithm for floorplan design. In: Kuh, S., ed. Proceedings of the 23rd ACM/IEEE Design Automation Conference. Michigan: IEEE Press, 1986. 101~107.

[14] Sait, S.M., Youssef, H. VLSI Physical Design Automation: Theory and Practice. New York: McGRAW-HILL, 1995.

[15] Johns, D.A., Martin, K. Analog Integrated Circuit Design. New York: John Wiley & Sons, 1997.

# 用非常快速模拟重复退火算法实现的模拟电路模块布局

张理洪[1], 裴先登[1], Ulrich Kleine[2]

[1](华中科技大学计算机学院 外存储国家实验室,湖北 武汉 430074);

[2](马格德堡大学,马格德堡 D-39016,德国)

摘要: 提出了基于非常快速模拟重复退火算法实现模拟电路模块布局的方法,该算法指数倍地快于传统的 Cauchy 或 Bolzmann 退火算法.其中使用一个滑行函数将绝对布局问题转化为相对布局问题,这样极大地减少了算法的搜索空间,而不会降低搜索成功率.价值函数根据模拟集成电路固有的特点设计而成,模拟电路设计者可根据电路的具体要求选择合适的网络长度估算器.使用最小 steiner 树方法的全局布线器与布局器同时工作,减轻了后续细节布线环节的工作量,并保证最后布局结果的可用性.最后,给出了使用该布局方法实现运算放大器的版图事例.

关键词: 布局;模拟退火;全局布线;模拟电路模块;steiner 树

中图法分类号: TP391 文献标识码: A