C:\Users\chliao\Desktop\xiaoming.eps

VLSI布图设计方法学

题目： Timber-Wolf VLSI电路布局

姓名： 翟金源

学号： 15210720098

院系： 微电子系

时间： 2016年6月 19日

目录

[摘要 1](#_Toc454825097)

[一．背景介绍 1](#_Toc454825098)

[二．Timber-Wolf 算法 2](#_Toc454825099)

[2.1 模拟退火算法 2](#_Toc454825100)

[2.2 Timber-Wolf 步骤1 3](#_Toc454825101)

[2.3. Timber-Wolf 步骤2 5](#_Toc454825102)

[三． Timber-Wolf 算法实现 5](#_Toc454825103)

[3.1 输入文件格式 5](#_Toc454825104)

[3.2 代码说明 6](#_Toc454825105)

[四．结果与分析 7](#_Toc454825106)

[文件说明 8](#_Toc454825107)

[参考文献 9](#_Toc454825108)

# 摘要

本文详细描述了Timber-Wolf算法，一种通过模拟退火算法进行VLSI电路布局的算法。

关键词：Timber-Wolf、模拟退火、布局

# 一．背景介绍

物理设计（或版图设计）是 VLSI 设计中最重要的环节之一。由于物理设计的复杂性，整个设计过程可分为划分（partition)、布图规划（floorplanning)、布局（placement）、布线（routing)、版图压缩（layout compaction）、参数提取和版图验证（extraction and verification）。

布局的任务是要确定模块在芯片上的精确位置，设计的好坏直接影响着电路的质量。布局的输入是一组模块、各模块上的引线端信息和网表。最重要的约束条件是模块之间不存在重叠，并且所有模块都位于芯片区域内。一般来说，电路布局的目标是为了减少线长。除了线长，也需要考虑一些其他约束条件，如芯片密度（chip density）、布通率（routability）、时延（timing）等。因此，布局问题最近引起许多关注，许多新的算法蜂拥出现以解决新的设计难题。

目前，VLSI 物理设计中的设计方式主要有四种：全定制设计（Full-Custom Design）；标准单元设计（Standard Cell Design）；门阵列设计（Gate Array Design）和逻辑可编程门阵列设计（Field Programmable Gate Array Design，简称 FPGA）方式。标准单元设计最大优势在于单元具有相同高度并且排列成行的规则形式，版图设计相对简单和容易，因而适合于大规模的集成。考虑到标准单元的普遍性和易用性，本文只考虑超大规模集成电路中标准单元的布局优化方法。

为了解决大规模电路布局的问题，布局通常分为了两个步骤：全局布局（global placement）和详细布局（detail placement）。全局布局主要是对各个模块做一个粗略的布局，全局布局中各个模块之间有重合是允许的。这些重合会在详细布局中解决。

常见的布局方法有下面几种：

* Mincut Placement [1]
* Gordian Placement [2]
* Timber-Wolf Algorithm [3][4]

Mincut Placement 利用划分进行布局，待布局的电路不断地划分为两个子电路，同时布局区域也相应地水平或竖直地划分，直到每个布局区域中子电路足够小，然后对每个子电路进行布局合法化。

Gordian Placement 是很成功的布局方法，利用二次规划进行布局。Gordian Placement根据电路的连线信息，将布局问题转换为二次规划问题，同时在每次迭代中，一些约束条件对模块的移动进行了限制。

Timber-Wolf Algorithm 是一种基于模拟退火算法和交换机制来进行的布局。下一章会具体介绍该算法。

# 二．Timber-Wolf 算法

Timber-Wolf一共有两个步骤：

1. 模块在不同行或相同行之间移动，允许模块之间有重叠，当温度下降到一定程度后，第二个步骤开始。
2. 主要目的是减少重叠。模拟退火继续，但只允许同一行中相邻的模块间的交换。

## 模拟退火算法

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却。加温时，固体内部粒子随温度升高变为无序状，内能增大；而徐徐冷却时粒子渐趋有序，在每个温度点都达到平衡态，最后在常温时达到基态，内能减为最小。根据Metropolis准则，粒子在温度T时趋于平衡的概率为e-ΔE/(k T)，其中E为温度T时的内能，ΔE为其改变量，k为Boltzmann常数。

用固体退火模拟组合优化问题，将内能E模拟为目标函数值f，温度T演化成控制参数T，即得到解组合优化问题的模拟退火算法：由初始解j和控制参数初值T开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减T值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表（Cooling Schedule）控制，包括控制参数的初值T0及其衰减因子α、每个T值时的迭代次数和停止条件。

## Timber-Wolf 步骤1

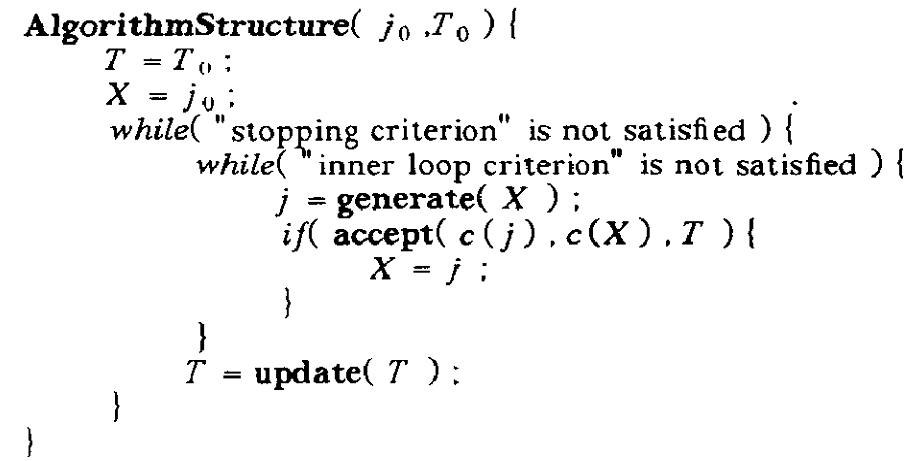


图 2- 1 Timber-Wolf 算法步骤1伪代码

Timber-Wolf 算法步骤1的框架如图2-1所示，其中j和T分别代表布局和温度，j0和T0是初始的布局和温度。

“**stopping criterion**” 为程序循环继续进行的条件，当温度降低到某个值时，循环停止，一般为0.1。

“**inner loop criterion**” 为内部循环程序继续进行的条件，当内部循环次数达到某个值时，内部循环停止。

“**generate function**” 通过移动某些模块产生一个新的布局状态。一共有三种移动，如图2-2所示：

1. 将某个模块放到某个新位置。
2. 交换两个模块的位置。
3. 改变某个模块的放置方向。

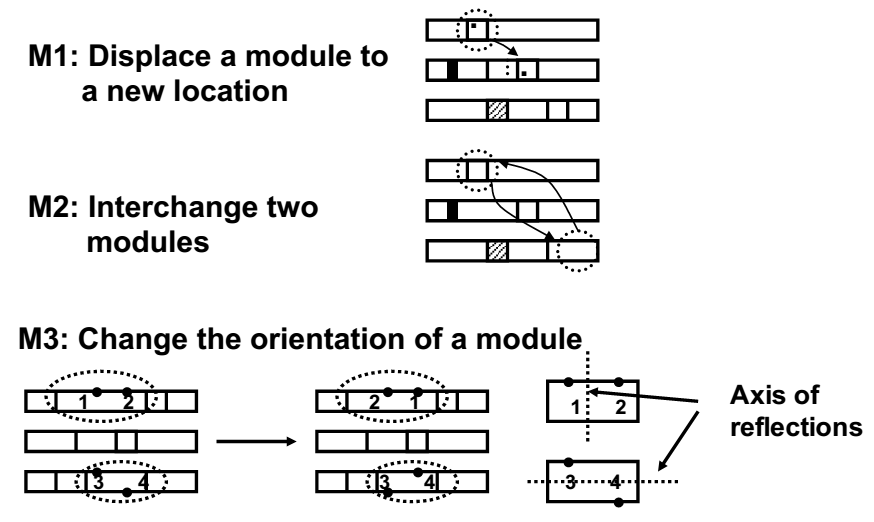


图 2- 2 模块移动的三种类型

首先，在M1和M2之间选择一种移动，M1的概率为0.8，M2的概率为0.2。如果选择了M1并且该移动被拒绝，此时对同一个模块，M3选中的概率是0.1。其次，某个模块移动的范围限定在一个矩形框内，如图2-3所示，矩形的竖直边长R正比于log(T),T为当前的温度。

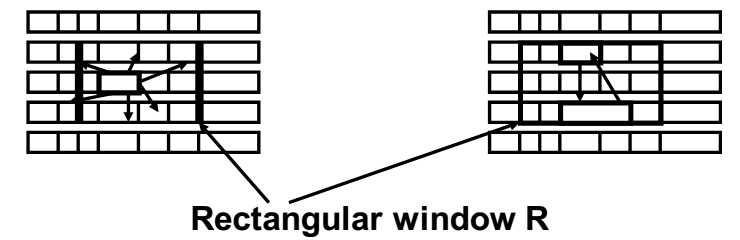
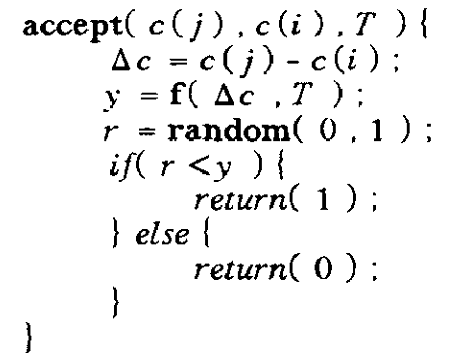


图 2- 3 模块移动范围的限制

**accept function**用于判断某个新的布局状态是否能被接受，accept的伪代码如图2-4所示。



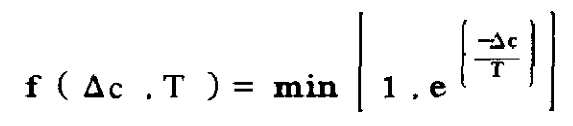


图 2- 4 accept函数伪代码

其中，c为cost function由三部分组成，c=c1+c2+c3。

1. 带权重的互联线的总长度。
2. 模块重合的惩罚函数。
3. 控制行宽度的惩罚函数，实际行长度与期望的行宽度相差越大，c3越大。

互联线的长度（即c1）的近似计算方法如图2-5所示。线长为互联线相连的各个模块的外围矩形框的半周长。

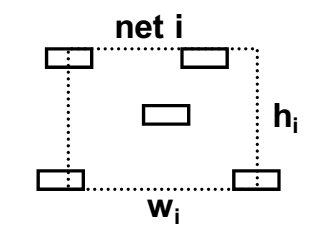


图 2- 5 线长计算

**update function**用于更新温度，其表达式为：。从0.8增长至0.94，最后降低至0.1。

## 2.3. Timber-Wolf 步骤2

当温度下降到图2-3中的竖直边长小于等于行的高度时，行间的模块交换和放置已经不被允许，Timber-Wolf算法进入步骤2。

步骤2的算法框架与步骤1类似，内循环变为：

1. 在某一行中随机选择一个模块。
2. 交换该模块与左边（或右边）的位置，如果被拒绝，换另外一边。

步骤2的cost function仅包含c1。

# 三． Timber-Wolf 算法实现

## 3.1 输入文件格式

输入文件来自ISPD 2004论文FasePlace [5]中使用的benchmarks。测试的输入文件为ibm05.nodes、ibm05.nets、ibm05.wts、ibm05.scl、ibm05.pl。各个文件的内容说明如下所示。

1. ibm05.nodes。描述每个模块的长、宽、是否固定等信息，格式为：

object\_name width height terminal

当一个模块位置固定时，会有关键词“terminal”。

1. ibm05.nets。描述连线网络。对于每个线网，按如下方式描述：

NetDegree : k net\_name

# input pin pin offset is measured from the center of corresponding object

object\_name I : x\_offset y\_offset

# output pin

object\_name O : x\_offset y\_offset

1. ibm05.wts。描述每个模块的权重。格式如下：

object\_name weight

1. ibm05.scl。电路每行的信息。格式如下：

CoreRow Horizontal

Coordinate : www # row starts at coordinate www

# (y-coordinate if horizontal, x-coordinate if vertical row)

Height : yyy # row height is yyy

Sitewidth : 1

Sitespacing : 1 # distance between the left edges of two adjacent sites

Siteorient : 1

Sitesymmetry :1

1. ibm05.pl。描述每个模块的位置和方位。格式如下：

# x\_location & y\_location are bottom left coordinates of an object

# orientation is one of N, S, E, W, FN, FS, FE, FW

# and interpreted as in LEF/DEF (see Orientation.ppt in this

# documentation for visual representations of each orientation) object\_name x\_location y\_location : orientation

# fixed object

object\_name x\_location y\_location : orientation /FIXED

输入文件中的各个模块、行等数量如表3-1所示。

表3-1 benchmark 的模块、行等的数量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| #Cells | #Pads  (terminal) | #Nets | #Pins | #Rows |
| 28146 | 1201 | 28446 | 126308 | 148 |

## 3.2 代码说明

代码在Visual Studio 2015中编写。头文件为main\_head.h，源文件为main.cpp。为了显示布局效果，使用matlab脚本文件plot\_placement.m读取输出文件，将布局结果可视化表示出来。源文件中使用了boost库，需要在compiler和linker中添加相应boost的路径。

头文件中的内容有：

1. 定义了node类型，代表布局的每个模块。node类中包含了模块的名、宽度、高度、方向、是否为terminal、左下角坐标、中心点坐标、所在行号和所连接的线网等属性，以及设置这些属性的方法。
2. 定义了row类型，代表布局的每一行。row类型中包含了行号、高度、宽度、方向、重合、模块间距、该行包括的模块等属性，以及设置这些属性的方法。
3. 定义了一些全局变量。map<string, node> nodeId，包括所有的模块信息；map<int, row> rowId，包括了所有行的信息；map<int, vector<string> > netToCell，包含了线网与模块的映射关系。

源文件主要以面向过程的思路编写，main函数为：

int main(){

readNodesFile();

readWtsFile();

readPlFile();

readNetsFile();

readSclFile(); //读入文件

gnuPlot("before1.txt"); //将布局写入 txt

timberWolfAlgorithm(); //模拟退火进行布局

timberWolfAlgorithm2();

gnuPlot("after1.txt");

return 0;

}

其中，5个readXXFile从读取对应的ibm05.XX文件中，并设置nodeId、rowId等中的属性。gnuPlot("before1.txt")函数将布局后每个模块的坐标和宽高输出出来，以方便matlab画图。timberWolfAlgorithm()和timberWolfAlgorithm2()分别是Timber-Wolf算法的第一和第二阶段。由于初始布局中模块重合比较多，在timberWolfAlgorithm()中，会进行一次初始布局，以减少重合，初始布局由模块顺序排列生成。

# 四．结果与分析

布局前线长为10828065，重合为2459632



布局后线长为10828006，重合为2456192，运行时间为98.1min。

|  |
| --- |
|  |

由于布局的规模比较大，原始布局也比较好，Timber-Wolf 算法不能在有限的时间内明显减少线长和重合。

# 文件说明

|  |  |
| --- | --- |
| 文件名 | 说明 |
| main.cpp | 布局程序源文件 |
| main\_head.h | 布局程序头文件 |
| placement2/placement2.sln | VS solution文件 |
| placement2/placement2/after1.txt | 布局后输出文件 |
| placement2/placement2/after1.fig | 布局结果图 |
| placement2/placement2/before1.txt | 布局前输出文件 |
| placement2/placement2/before1.fig | 布局前的图 |
| placement2/placement2/ibm05.xx | 输入testbench文件 |
| placement2/placement2/plot\_placement.m | 根据输出文件作图的matlab脚本 |

# 参考文献

1. Gerlhof C, Kemper A, Kilger C, et al. Partition-based clustering in object bases: From theory to practice[J]. Lecture Notes in Computer Science, 1995, 730:301-316.
2. Kleinhans J M, Sigl G, Johannes F M, et al. GORDIAN: VLSI placement by quadratic programming and slicing optimization[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1991, 10(3): 356-365.
3. Sechen C, Sangiovannivincentelli A. The TimberWolf placement and routing package[J]. IEEE Journal of Solid-state Circuits, 1985, 20(2): 510-522.
4. Sechen C, Sangiovannivincentelli A. TimberWolf3.2: A New Standard Cell Placement and Global Routing Package[J]. Design Automation Conference, 1986.
5. Viswanathan N, Chu C C. FastPlace (efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model)[J]., 2004.