# 插图索引

[图2.1 批量模式命令 9](#_Toc96457125)

[图2.2 SAI文件示例 10](#_Toc96457126)

[图2.3 SAI流程图 11](#_Toc96457127)

[图2.4 CMOS工作电流示意图 14](#_Toc96457128)

[图2.5 不同工艺下三种功耗所占比例 14](#_Toc96457129)

[图2.6 多电源多电压技术 17](#_Toc96457130)

[图2.7 电平转换单元应用示意图 17](#_Toc96457131)

[图2.8 动态功耗占比示意图 18](#_Toc96457132)

[图2.9 ICG结构示意图 18](#_Toc96457133)

[图2.10 动态电压频率调节技术 19](#_Toc96457134)

[图2.11 电源关断技术 19](#_Toc96457135)

[图2.12 隔离单元应用的示意图 20](#_Toc96457136)

[图2.13 不同器件的漏电功耗与延迟 21](#_Toc96457137)

[图2.14 衬底偏置技术示意图 21](#_Toc96457138)

[图3.1 派系模型 24](#_Toc96457139)

[图3.2 星型模型 24](#_Toc96457139)

[图3.3 单元密度及单元受力示意图（1） 26](#_Toc96457140)

[图3.4 单元密度及单元受力示意图（2） 26](#_Toc96457140)

[图3.5 单元密度及单元受力示意图（3） 27](#_Toc96457141)

[图3.6 单元密度及单元受力示意图（4） 27](#_Toc96457141)

[图3.7 混合尺寸布图方法流程图 27](#_Toc96457142)

[图3.8 BkTrk算法 30](#_Toc96457144)

[图3.9 Nesterov算法 30](#_Toc96457144)

[图3.10 传统布图方法WNS、TNS报告 32](#_Toc96457145)

[图3.11 混合尺寸布图方法WNS、TNS报告 33](#_Toc96457146)

[图3.12 传统布图方法DRV数量报告 33](#_Toc96457147)

[图3.13 混合尺寸布图方法DRV数量报告 33](#_Toc96457148)

[图3.14 传统布图方法单元密度报告 33](#_Toc96457149)

[图3.15 混合尺寸布图方法单元密度报告 34](#_Toc96457150)

[图3.16 传统布图方法功耗报告 34](#_Toc96457151)

[图3.17 混合尺寸布图方法功耗报告 34](#_Toc96457152)

[图3.18 传统布图方法线长报告 35](#_Toc96457153)

[图3.19 混合尺寸布图方法线长报告 35](#_Toc96457154)

[图3.20 传统布图方法热点值报告 35](#_Toc96457155)

[图3.21 混合尺寸布图方法热点值报告 35](#_Toc96457156)

[图3.22 设计A布图质量提升概况 36](#_Toc96457157)

[图3.23 传统布图方法整体布局图 36](#_Toc96457158)

[图3.24 混合尺寸布图方法整体布局图 36](#_Toc96457158)

[图3.25 传统布图方法WNS、TNS报告 37](#_Toc96457159)

[图3.26 混合尺寸布图方法WNS、TNS报告 37](#_Toc96457160)

[图3.27 传统布图方法DRV数量报告 38](#_Toc96457161)

[图3.28 混合尺寸布图方法DRV数量报告 38](#_Toc96457162)

[图3.29 传统布图方法单元密度报告 38](#_Toc96457163)

[图3.30 混合尺寸布图方法单元密度报告 38](#_Toc96457164)

[图3.31 传统布图方法功耗报告 39](#_Toc96457165)

[图3.32 混合尺寸布图方法功耗报告 39](#_Toc96457166)

[图3.33 传统布图方法线长报告 39](#_Toc96457167)

[图3.34 混合尺寸布图方法线长报告 40](#_Toc96457168)

[图3.35 传统布图方法热点值报告 40](#_Toc96457169)

[图3.36 混合尺寸布图方法热点值报告 40](#_Toc96457170)

[图3.37 设计B的布图质量提升概况 41](#_Toc96457171)

[图3.38 传统布图方法整体布局图 41](#_Toc96457172)

[图3.39 混合尺寸布图方法整体布局图 41](#_Toc96457172)

[图3.40 TAT对比 43](#_Toc96457173)

[图4.1 自动布图流程 45](#_Toc96457174)

[图4.2 电源线分布示例 46](#_Toc96457175)

[图4.3 spacing示意图 48](#_Toc96457176)

[图4.4 补偿电源线密度差异流程图 48](#_Toc96457177)

[图4.5 产生的OBS示意图 49](#_Toc96457178)

[图4.6 电源线模型结果示例 51](#_Toc96457179)

[图4.7 电源线模型优化前后优化提升对比图 52](#_Toc96457180)

[图5.1 早期门级网表生成器流程图 53](#_Toc96457181)

[图5.2 库文件及单元选配 54](#_Toc96457182)

[图5.3 设计参数配置 55](#_Toc96457183)

[图5.4 宏单元配置文件 56](#_Toc96457184)

[图5.5 宏单元生成步骤 57](#_Toc96457185)

[图5.6 UPF配置文件示例 57](#_Toc96457186)

[图5.7 电源域生成步骤 58](#_Toc96457187)

[图5.8 电源域生成结果示例 58](#_Toc96457188)

[图5.9 单元查找器实现思路 60](#_Toc96457189)

[图5.10 create\_empty\_module模块 61](#_Toc96457190)

[图5.11 克隆层次化器件树状结构描述文件示例 61](#_Toc96457191)

[图5.12 结构生成模块树状结构描述文件格式示例 62](#_Toc96457192)

[图5.13 克隆层次化器件树状结构示例图（1） 64](#_Toc96457193)

[图5.14 克隆层次化器件树状结构示例图（2） 65](#_Toc96457194)

[图5.15 克隆层次化器件树状结构示例图（3） 65](#_Toc96457195)

[图5.16 connect\_module\_inst模块 67](#_Toc96457196)

[图5.17 加法电路 6](#_Toc96457197)7

[图5.18 减法电路 67](#_Toc96457197)

[图5.19 乘法电路 68](#_Toc96457197)

[图5.20 迷你电路 68](#_Toc96457197)

[图5.21 触发器与缓冲器电路 68](#_Toc96457198)

[图5.22 触发器电路 68](#_Toc96457199)

[图5.23 connect\_module\_macro模块 68](#_Toc96457200)

[图5.24 层次化器件连接的流程图 69](#_Toc96457201)

[图5.25 层次化器件输入输出端口连接关系模型图 70](#_Toc96457202)

[图5.26 宏单元配置文件 71](#_Toc96457203)

[图5.27 库文件及单元选配 71](#_Toc96457204)

[图5.28 设计基本信息 72](#_Toc96457205)

[图5.29 克隆层次化器件结构描述文件 72](#_Toc96457206)

[图5.30 predict\_floorplan.tcl 72](#_Toc96457207)

[图5.31 克隆层次化器件部分结果 73](#_Toc96457208)

[图5.32 标准层次化器件部分结果 73](#_Toc96457208)

[图5.33 器件的层次结构（1） 73](#_Toc96457209)

[图5.34 器件的层次结构（2） 73](#_Toc96457209)

[图5.35 connect\_module\_macro的结果 74](#_Toc96457210)

[图5.36 connect\_module\_inst的结果 74](#_Toc96457211)

[图5.37 触发器的数量 74](#_Toc96457212)

[图5.38 顶层添加的端口的部分结果 75](#_Toc96457213)

[图5.39 顶层添加的连线的部分结果 75](#_Toc96457213)

[图5.40 自动布图结果缩略图 75](#_Toc96457214)

[图5.41 自动布图结果细节图 75](#_Toc96457214)

[图5.42 SAI工具使用脚本 75](#_Toc96457215)

[图5.43 SAI工具设计结果（1） 76](#_Toc96457216)

[图5.44 SAI工具设计结果（2） 76](#_Toc96457216)

[图5.45 SAI布图结果 77](#_Toc96457217)

[图5.46 SAI布图结果细节图 77](#_Toc96457217)

# 第一章 绪论

## 1.1 选题意义与背景

自集成电路发明问世以来，集成电路产业快速发展壮大，成为了当今构建万物互连世界中的重要基石[1]。随着工艺技术的进步，芯片的集成能力突飞猛进，特别是进入深亚微米后，单位尺寸上可集成的晶体管数量更是不可同日而语。逐渐增加的晶体管数量使布图难度不断提高，逐渐增加的串扰和多模式多端角增加了时序的收敛难度，互连复杂性的提升加剧了拥塞程度和可制造性的问题[2]。为提高设计质量就不得不进行早期布图规划，早期阶段良好的决策可以减少设计回溯、缩短研发周期并且提高设计质量。

针对早期布图规划的需求，Cadence公司已经在INNOVUS产品中给出了解决方案——SAI。SAI工具可以在门级网表不完整的情况下补全门级网表来帮助后端工程师进行早期布图规划。除此以外，它还能够将一个高级框图转换成一个门级网表，为项目规划做好准备。SAI工具虽然可以满足此类需求，但SAI工具无法在创建的门级网表中加入连接关系和复杂的组合逻辑，并且工具的使用难度很高，要求用户必须对SAI工具的命令非常熟悉。用户在使用SAI工具时能够控制的变量很少并且自动化程度很低，因此定制设计的复杂度非常高。针对以上缺陷，需要有工具能够实现自动生成具有连接关系和组合逻辑的高级门级网表，帮助用户更快更好的完成工作。

目前，电子产品更新换代的速度越来越快，特别是消费类电子市场，大部分的手机、平板电脑厂商会在一年内发布数款新型号的产品覆盖低、中、高端市场供消费者选择[3]。消费级电子产品的更新不限于产品的外观和颜色，芯片的更新带来性能上的提升才是重中之重。

为了缩短芯片研发周期，加快芯片上市时间，芯片后端物理设计周期需要被进一步压缩，而在芯片设计后端的整个流程中，布图所需的时间往往最长，因此压缩布图时间能够有效降低后端物理设计周期。目前市面上常用的布图方法需要后端工程师手动摆放宏单元，这个过程需要经过多次迭代才能产生一个较为理想的布图结果，手动摆放不但极为耗时，并且对后端工程师的能力要求非常高。如果有一种能够实现宏单元和标准单元自动摆放的布图工具，就能够降低芯片的成本，减少芯片的开发周期，节省后端工程师的时间。但这种布图工具除了要满足降低布图时间的要求以外，还必须要满足布图质量不能比传统布图方法差的要求。

虽然近几年国内半导体行业发展迅速，但与美国相比依然有不小的差距。在设计领域，世界前十的无工厂公司中，6个是美国的，其中有博通、高通、超威、英伟达、美满电子、赛灵思[4]。在EDA领域，主流的EDA工具都是由Cadence和Synopsys两家美国公司提供。随着半导体工艺的不断进步，芯片集成度不断增加，芯片设计难度逐渐提高，后端物理设计早已离不开EDA工具的支持。后端物理设计对EDA工具的要求很高，EDA和工艺联系紧密，为最先进的制造工艺提供相应的分析计算模型和优化算法，很大程度上决定了后端设计的整体结果[5]。

EDA工具研发团队中，最重要的成员之一就是EDA验证工程师，并且EDA验证工程师在研发团队中的占比最大。EDA验证工程师的工作主要包括测试新功能和测试已开发功能中的漏洞两个部分，无论是测试新功能还是测试已开发功能中的漏洞，都需要在设计库中寻找合适的设计进行测试，当测试一些对设计要求比较极端的功能时，工程师们往往无法在现成的设计中找到符合要求的，这时候就需要工程师在现有设计的基础上进行改造，这个过程是十分繁琐且耗时的。对于EDA验证工程师来说，拥有复杂多样的设计可以大大提高工作效率和测试覆盖率。目前市面上没有一款工具可以帮助EDA验证工程师快速定制所需设计，如果能够推出这款工具，一定可以为EDA验证工程师节约大量的时间。

本论文根据以上实际应用中的需求，设计了一款早期自动布图规划工具，后端设计工程师可以使用它进行早期布图规划，EDA验证工程师可以使用其中的早期门级网表生成器产生需要的门级网表进行EDA验证工作。

## 1.2 国内外研究现状

### 1.2.1 布图算法的国内外研究

无论在规划过程中还是布图过程中，布图算法起到了至关重要的作用。随着集成电路的不断发展，布图算法也随之不断改进，布图算法经历了长久的发展之后，呈现出各式各样优秀的解决方案。对于标准单元摆放的布图算法主要包括[6]：分析布图算法（analytical placement），模拟退火算法（simulated annealing）以及聚类算法（partitioning/clustering），其中应用最广的是分析布图算法，它利用二次目标函数能够十分有效的减小线长，处理较大的电路时十分有效。

Kleinhans在分析布图算法的基础上提出了改进，取名为GORDIAN[7]。这种算法在进行单元全局优化的同时也交替进行分区的步骤，然后对每一个分区进行优化，这是一种分而治之的思想。它在优化时并不会一个一个区域的进行，而是对所有的单元位置一起进行优化。这种算法不会和以往的算法一样，为了获得一个更好的解决方案而要从随机生成的分区中选择一个，也不会陷入局部最优的陷阱中去。在每一次的迭代之后，每一个单元都会获得一个确切的位置，并且随着每一次的迭代，在可以自由移动的单元上的约束越来越多，直到能得到一个最优的结果。

Jianli Chen等人在分析布图法的基础上开发了一种新的布图工具，解决了VLSI标准单元放置问题[8]。布图的流程包括两个阶段：全局布图(GP)和详细布图(DP)。在GP阶段，他们使用非线性规划技术和最佳选择聚类算法得到一个初始的布图结果，然后利用迭代局部加密技术进一步分散单元并减小线长。在DP阶段，他们开发了一种快速合法化算法，使全局布图的解合法。该算法在IBM基准电路和Peko套件上进行了测试，实验证明，GP阶段的算法产生的全局布图质量很高，DP阶段的算法也是快速且有效的。但这个算法依旧没有解决宏单元布图无法实现自动化的问题。

高温或温度不均匀已经严重威胁到高性能集成电路的性能和可靠性，使得热效应成为电路设计或物理设计中不可忽视的问题。Jai-Ming Lin等人提出了一种将非线性布图模型与解析热模型相结合的多层布图方法，这种方法能够在固定轮廓约束的条件下同时考虑线长和热问题[9]。为了优化温度，他们提出了热感知聚类、热模块收缩、热力调制等技术。与其他方法相比，他们研究的方法能有效降低芯片温度且运行速度相当快，但降低芯片温度需要以增加芯片导线长度为代价。

在国内的研究论文中，刘正轩等人将仓库中各单位的运输成本与运输量类比为异构片上网络中的功耗和通信量，通过工业工程领域里的设施规划中的SLP方法来对IP核进行布图规划[10]。他们将宏观布图的SLP方法应用于微观的异构片上网络布图中，并结合遗传算法得到布图方案。该算法不仅能有效降低片上网络的功耗，而且还可定性与定量地优化布图。但只有当IP核之间的平均核通信量在1000Mb/s以内时，功耗优化效果才明显。

杨润萍等人针对引线压焊技术封装芯片的多电压布图问题进行了研究，提出了一种考虑电压岛边界约束的多电压布图算法[11]。首先，基于可切分布图解表达式的特点，提出一种快速的模块边界检查算法，用于判断切分树中每个节点是否位于版图的四周边界。其次，改进基于动态规划的多电压分配算法，使其仅对位于版图四周的节点进行多电压分配，以优化功耗并生成满足边界约束的矩形电压岛。最后，将上述算法集成在基于模拟退火算法的搜索框架中，并采用一个两阶段的降温策略来更新温度，以减少算法的迭代时间。

随着集成电路的规模变得越来越大，设计越来越复杂，电路设计中宏的数量急剧增加，对于这样的大规模设计，手动放置宏单元耗时耗力，因此混合尺寸布图应运而生。混合尺寸布图的主要目的是实现宏单元和标准单元的自动摆放，混合尺寸布图的主要困难是所有可移动物体的拓扑和物理属性的广泛性。

平面图指导方法结合了多种算法的优点，布图工具会同时优化宏和软块（标准单元群），之后进行增量放置，在局部扩散标准单元的坐标。例如[12]在自顶向下的布图框架上反复调用固定边界的布图算法，为宏单元的移动提供指导。[13]将所有标准单元组成软块，产生初始平面图解决方案，详细布图进一步使标准单元在局部区域内合法化。

一阶段法会将宏单元和标准单元同时放置并优化，解决了以前算法中优化空间小、布图质量损失难以恢复以及误差大等缺陷，因此在现代布图算法中广为采用。例如[14]将宏单元分解成大小与标准单元相似的小块，放置完成后，每个宏单元基于它的重心重新构建。APlace3[15]重塑了密度函数的平滑曲线，以区分宏单元移动的平滑度和标准单元的平滑度。[16]将旋转和翻转方法集成到了梯度函数中，从而可以同时优化所有可移动对象的位置以及宏单元的方向。

随着混合尺寸布图的发展，Prasun Datta等人研究了一种受植物资源配置和模式启发的新型现代绕线驱动布图算法，它模拟了VLSI电路中各个器件的资源分配技术[17]。他们在算法中设计了一种新的布图数学公式，不同于现有的绕线驱动布图算法，他们将VLSI电路模块映射到植物系统，获得了一个基于线网的布图，保持绕线驱动的方法完整。在植物中，根系统负责向茎系统提供资源，茎系统进一步向植物的不同分支提供资源。生长素是一种负责将资源分配到具有最大横截面积的枝条上的激素，他们正是基于植物资源分配生命的模型和模式，提出了绕线驱动的全局布图算法。同时，他们在另一篇文章中提出了绕线驱动的布图结果比线长驱动的布图结果更好的观点[18]，因为前者考虑了全局绕线约束和布图约束，这有助于满足集成电路设计中的详细绕线。同时，他们提出了一种绕线驱动的布图算法用于大型混合尺寸的设计。之后他们利用设计层次、引脚偏移量和引脚方向信息，设计了一种新的电路块聚类技术和一种考虑引脚信息的绕线感知单元扩展方案来实现合法化。最后，利用单元交换和移动，给出了一个与绕线、拥塞相关的详细布图。

随着特征尺寸的减小，放置具有多个阈值电压的单元可能会违反最小植入面积(MIA)层规则。Jianli Chen等人在两个主要阶段解决了具有MIA约束的混合单元高度配置问题：后全局布图和MIA-aware合法化[19]。在全局布图后阶段，他们首先提出了一个连续可微的代价函数来解决Vdd/Vss对齐约束，并动态地向MIA冲突单元添加加权伪网。然后，使用了一种基于给定全局布图结果的近端优化方法，该方法同时考虑了Vdd/Vss对齐约束、MIA约束、单元分布、单元位移和总导线长度。在MIA-aware合法化阶段，他们开发了一种基于图的方法来聚类特定阈值电压的单元，并应用二进制线性规划来重塑单元，然后利用了一种基于匹配的技术来解决行内MIA违规和减少填充单元插入。实验结果表明，该算法能有效缩短总线长。

近年来，在布图算法中引入人工智能和机器学习成为布图算法研究的新热点。边少鲜等人使用Cadence Innovus工具建立应用机器学习进行延时优化的物理设计流程，研究7nm工艺下不同层金属的特性，将融入机器学习的后端物理设计流程与传统物理设计流程进行对比分析[20]。应用机器学习后可使频率提升40 MHz并减少大量的时钟门控单元。Cadence公司在最新版本的INNOVUS布图引擎中加入了人工智能算法帮助工具快速进行布图工作，大幅缩短后端物理设计周期的同时取得了理想的结果。Cadence公司在2021年推出的新工具Cerebrus[21]，它是全球首创机器学习核心的EDA自动化工具，让客户能够高效达成要求严苛的芯片设计目标。

在 2020 年的 ISSCC 大会上，Jeff Dean 进行了名为“深度学习革命及其对计算机体系结构和芯片设计的影响”的演讲报告，介绍了利用人工智能进行芯片设计的思路和进展[22]。Jeff Dean还介绍了人工智能算法进行芯片布图规划工作的研究，分别展示了人工智能算法的布图结果和工程师的布图结果，经过对比可以看到，人工智能算法的结果线长更短、功耗更低[23]。并且人工智能算法的布图时间更短，不到一天即可完成，而工程师进行手工布图则需要 8 周左右，大幅缩短了产品的研发周期且节约了人力成本[24]。虽然目前人工智能技术的应用还不够成熟，但相信在不久的将来定会大放异彩。

目前市面上普遍应用的是传统的布图方法，传统布图方法需要人工进行宏单元的摆放，耗时耗力。而在多数论文中研究的混合尺寸布图算法存在优化步长不可控、拥塞优化程度不高以及宏单元和标准单元无法同时摆放等问题。

### 1.2.2 早期网表生成工具的国内外研究

布图规划这个主题在相关领域的研究员眼中并不陌生，在各类核心期刊及论文网站中可以搜索到大量的和布图规划算法相关的论文，如杜世民等人发表的关于固定边框的多电压布图规划算法的文章[25]、张腾等人发表的关于考虑缺陷率模型的多项目晶圆布图规划算法的文章[26]以及陈振发表的协同多目标布图规划算法研究的文章[27]等，但这些算法往往被设定于得到完整设计信息后使用。早期的门级网表中往往不包含内部细节，导致布图算法应用在设计的早期阶段时表现不佳，因此市面上对于早期布图规划问题没有丰富的解决方案。

Jeonghee Shin等人在一篇论文中分析了几种方法[28]，第一个是基于迁移算法的早期布图规划，首先提取先前设计的平面图构建一个层次结构，其中的节点与宏单元相对应，之后引入加权线性约束固定宏单元的大小和位置，然后通过最小化惩罚函数生成新的平面图。第二种方法是基于约束求解器的早期布图规划，这种方法对宏单元设置最大距离限制和最小距离限制、全局约束以及首选位置的限制，之后进行迭代，直到产生面积最小的方案。第三种方法是基于线性/非线性规划，这种方法将一个已有的设计的门级网表划分为几组，这些组被视为具有固定面积和预定义宽高比的软块。根据顶层结果，保留这些组的相对位置，将宏单元约束在其组内。在另一篇论文中，他们在早期布图规划之前选择一个之前的设计作为参考，早期的芯片规划始于建立这些参考设计的库[29]。工具根据需要放置合适的单元，这些单元是从参考设计库中提取的，并根据目标微体系结构的变化进行了扩展。根据从参考设计中提取和调整的引脚信息以及物理规划结果，使用组合逻辑单元的数据创建新的门级网表。这些方案都过于依赖参考设计，灵活性和独立性很差。

随着集成电路的不断发展，摩尔定律逐渐失效，缩小芯片尺寸的任务变得十分艰巨。为了缩小尺寸，行业内不断涌现出新工艺和新技术，其中最火热的研究方向非3D IC莫属了，它将硅晶圆或裸晶垂直堆叠到同一个封装器件中，从而带来性能、功耗和面积优势[30]。Cadence公司在2021年推出了新产品 Integrity 3D-IC平台，帮助后端工程师进行3D IC的设计[31]。Matthew Grange等人提出了从物理到系统级别的分层模型，用于高性能硅系统的体系结构探索，以量化2D和3D IC实现的性能和成本折衷[32]。这些模型被打包为一个独立的工具，可以为在早期芯片规划阶段使用的各种处理系统提供粗略快速的估计。

近年来Cadence公司在INNOVUS产品中推出了SAI工具[33]。当一个新的项目有参考设计或者有部分门级网表时，可以使用SAI工具分别产生某几个模块的门级网表进行补全，帮助进行早期布图规划。它的运行速度很快并且包含了原型设计所需的关键信息，能够提供框图到实际设计的快速路径，帮助快速进行项目可行性评估。它为前后端工程师搭建了一个平台，前端工程师可以在这个平台上很快地将设计进行物理实现，前后端工程师可以基于这个平台进行设计、策划和交流。这个工具推出后受到了众多客户的好评，被广泛的应用在各个芯片项目当中。

目前，绝大多数早期布图规划使用的门级网表生成工具产生的是没有细节的粗略网表，因此导致布图规划结果质量不高，并且这些工具在自动化程度和使用便利程度上都有待提高。同时，部分工具过于依赖参考设计的门级网表，缺乏灵活性和独立性。

## 1.3 论文的主要工作

论文基于实习单位的实习项目，设计了一款早期自动布图规划工具，它由一个早期门级网表生成器和一个混合尺寸布图方法组成。论文首先介绍了SAI和低功耗的相关基础知识。之后介绍了论文研究的混合尺寸布图方法所采用的算法以及流程，并将混合尺寸布图方法和传统布图方法得到的布图结果进行对比，证明了论文研究的混合尺寸布图方法在布图质量和效率方面的优势。接下来，论文介绍了电源线模型的缺陷以及优化策略，对比了使用优化前后的电源线模型的布图结果，证明了电源线优化模型的优势。最后，论文介绍了早期门级网表生成器的流程和实现原理，同时介绍了早期门级网表生成器所使用的部分算法。论文将早期门级网表生成器和SAI工具得到的门级网表进行对比，证明了早期门级网表生成器相比于SAI工具的优势。

论文的具体工作如下：

第一章是绪论，介绍了论文的选题意义与背景，分析了目前在相关行业所面临的实际问题。之后介绍了相关领域的国内外研究现状和论文的主要工作。

第二章主要介绍了SAI和低功耗技术的相关原理。本章首先介绍了SAI的相关功能和优势，之后重点介绍了SAI的两种使用模式和使用流程，最后讲解了SAI中常用的命令。本章的后半部分，首先介绍了数字集成电路设计中功耗的来源，最后详细介绍了低功耗后端物理设计中常见的六种低功耗技术。

第三章主要研究了混合尺寸布图方法的算法及流程，并将混合尺寸布图方法与传统布图方法的布图结果进行了对比。本章首先介绍了混合尺寸布图方法的算法中所使用的派系模型和静电密度模型，然后讲解了混合尺寸布图方法的流程。最后将两种布图方法的布图结果进行了对比，证明了论文研究的混合尺寸布图方法在布图质量和耗时方面更具优势。

第四章主要介绍了电源线模型的优化策略。本章首先讲解了电源线模型的应用和电源线模型的缺陷，之后介绍了电源线模型的改进方案及改进后的公式，最后展示了使用电源线优化模型后布图结果的提升，证明了电源线优化模型的优势。

第五章主要介绍了早期门级网表生成器的实现流程、实现原理和使用到的部分算法，最后对早期门级网表生成器与SAI工具生成的门级网表进行对比。本章首先介绍了早期门级网表生成器的流程，之后按照流程的顺序分别介绍了数据准备，树状结构的生成以及添加连接关系等环节，同时介绍了开发的模块以及部分模块使用到的算法原理。最后对比了两种工具，实验证明早期门级网表生成器在自动化程度、生成的门级网表的质量以及便利程度方面更具优势。

第六章对论文的主要工作进行总结，指出了早期自动布图规划工具的不足和需要改进之处，并对以后的工作进行了展望。

# 第二章 SAI与低功耗技术研究

本章主要介绍了SAI技术与低功耗技术。本章的前半部分首先介绍了SAI的相关功能和优势，之后重点介绍了SAI的两种使用模式和使用流程，最后讲解了SAI中常用的命令。本章的后半部分首先分析了功耗的来源，最后重点讲解了六种常见的低功耗技术。

## 2.1 SAI

SAI的全称是SoC Architecture Information，系统级芯片架构信息。SAI允许在设计过程中更早地进行布图规划和分析，当门级网表不可用时，SAI允许探索设计的可行性，并根据相关信息创建出和真实设计相同结构、版图大小以及相关IP的门级网表。此外，SAI具备强大而完备的设计规划能力，能够提供框图到实际设计的快速路径，帮助快速进行项目评估。同时，SAI的运行速度很快并且包含了原型设计所需的关键信息，还能为前端和后端设计团队提供一个理想的交流、策划和评估的平台。用户可以使用Innovus对SAI创建出的门级网表进行布图规划。SAI为用户提供了以下功能：

1、解析不完整的门级网表和SAI文件，并创建一个新的门级网表。

2、添加虚拟单元以模拟定义模块的大小。

3、添加虚拟触发器模拟模块之间的边界连接。

4、根据连接文件中的内容添加寄存器。

5、为定义的边界连接创建SDC时间约束文件，并读入Innovus。

6、定义版图尺寸并读入Innovus。

### 2.1.1 SAI的两种模式

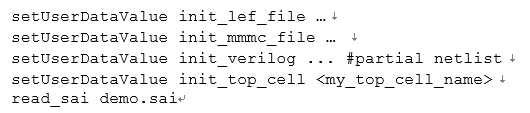


图2.1 批量模式命令

SAI支持批量模式和交互模式两种使用方式。当用户选择批量模式时，需要用到的命令如图2.1所示。在图中的前四行分别配置了需要用到的lef文件、mmmc文件、不完整的门级网表文件以及顶层名称，若在配置时没有门级网表文件也可以选择不配置它。第五行的命令“read\_sai”是SAI的开启命令，它不但能够在提示符上交互式地使用SAI命令，也可以在运行脚本中使用它来执行SAI命令。在批量模式下，可以像图2.1中一样直接使用命令“read\_sai demo.sai”。SAI文件的示例如图2.2所示。SAI文件包含了芯片架构、铸造规范的参考单元门、参考触发器、宏单元或特殊ip、不同尺寸和端口的内存、总线连接、软模块(分区)、现有的部分门级网表、时钟和版图尺寸，相关的命令会在后续介绍。

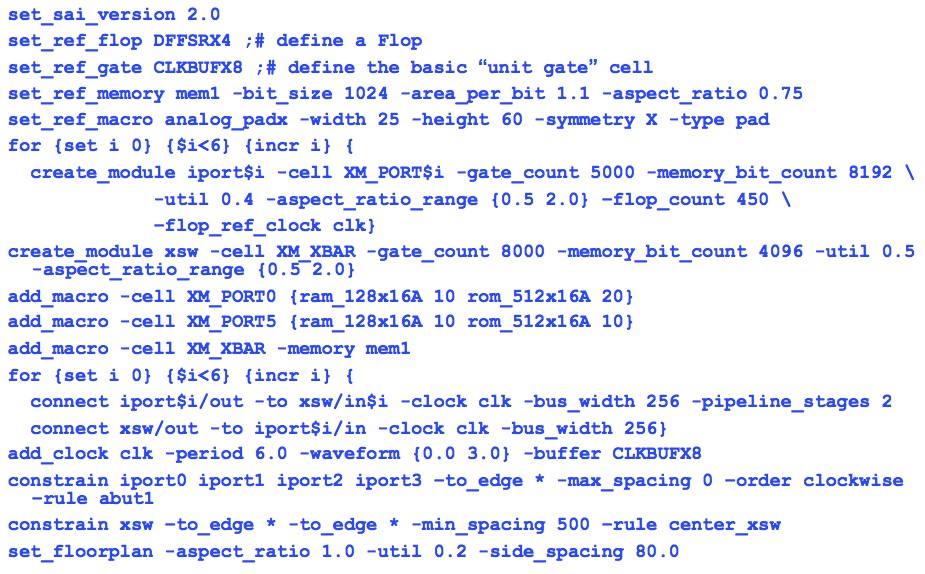


图2.2 SAI文件示例

当用户选择交互模式时，配置lef文件、mmmc文件、不完整的门级网表文件以及顶层名称的方法与批量模式相同，配置完后使用“read\_sai”命令开启SAI会话，之后再使用SAI命令构建设计结构。输入的SAI命令包括“create\_module”、“connect”、“add\_macro”等，用户还可以使用-help选项来查看该命令的所有选项。完成后用户需要使用“end\_sai”命令结束SAI会话。

### 2.1.2 SAI的流程

目前SAI已经升级为2.0版本，在SAI 2.0版本中，设计师可以指定平面图约束来指导模块和宏单元布图，设置的约束包括模块与边界的最小间距、模块之间的最小间距以及沿边界放置指定的宏单元等。同时，SAI 2.0可以在GUI上识别任何违反平面图约束的情况并报告平面图质量。如图2.3所示为SAI流程图。

推荐用于早期布图规划的SAI 2.0流程包括以下步骤:

1. 创建SAI文件。

2. 将SAI文件读入Innovus生成SAI设计。

3. 调用自动平面图综合来放置模块和宏单元。

4. 检查平面图约束并报告平面图质量指数。

5. 若平面图质量指数为零或接近零，继续正常的Hierarchical流程。

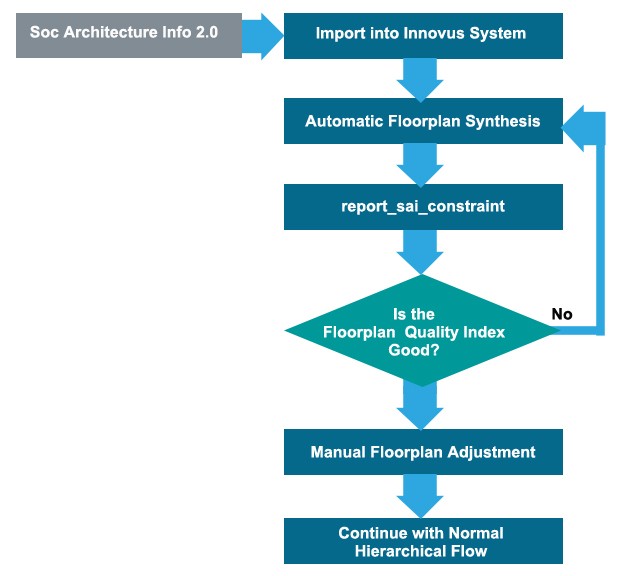


图2.3 SAI流程图

SAI命令可以用来生成顶层的门级网表，也可以用来添加新的模块或修改已有的模块。目前常见的应用环境包括：

1、顶层门级网表准备好了但一些模块还不完整，不完整的模块可能只有端口定义或部分逻辑。

2、没有顶层门级网表，但一些模块已经准备好了。

### 2.1.3 SAI的相关命令

SAI命令“create\_module”能够在创建的门级网表中添加模块。如表2.1所示为命令“create\_module”选项表。

表2.1 create\_module命令选项表

|  |  |
| --- | --- |
| -aspect\_ratio\_range | Specifies the aspect ratio range for proto\_design. These aspect ratios will appear in the sai\_proto) design.seed file. |
| -cell *moduleName* | Specifies the module name. A module is created if the specified module name does not exist. If the module name is not specified, then the local hierarchical instance name is used as the module name. |
| -flop\_count *string* | Specifies the number of flops to insert as intermediate flops. It uses the cell set for set\_ref\_flop. To add flops with multiple clock domains, specify {numFlops1 clkName1 numFlops2 clkName2 ... }. |
| -flop\_ref\_clock *string* | Specifies the flop's reference clock |
| -gate\_count *string* | Specifies the number of instance to insert. |
| -help | Outputs a brief description that includes type and default information for each create\_module parameter. |
| -incremental | Specifies to incrementally fill incomplete modules that might have only port definition or have some logic inside. |
| -memory\_bit\_count | Specifies the number of memory bits in the module. |
| *name* | Specifies hierarchical instance name to generate. |
| -util *float* | Specifies the target utilization value for proto\_design. This target utilization appears in the sai\_proto\_design.seed file. |

SAI命令“add\_macro”可以在创建的门级网表中加入宏单元，如表2.2所示为命令“add\_macro”选项表。

表2.2 add\_macro命令选项表

|  |  |
| --- | --- |
| -cell *targetModule* | Specifies target module to add macros. |
| -help | Outputs a brief description that includes type and default information |
| -memory *string* | Specifies the reference memory name。 |
| -names *string* | Specifies the names for adding memory and macro. |
| -prefix *string* | Specifies the prefix for adding macro. |
| *ref\_counts* | Specifies a list of macro cell name and count pair. |
| -update\_memory\_bit\_count | Updates memory bit count for adding reference memory further. |

“read\_sai”命令有两个选项可以使用：

1、-rule\_only：当用户只想读取平面图约束的信息时使用这个选项。

2、-reduce\_by\_flexfiller：使用与FlexModel相同的FlexFiller技术来减小生成的门级网表大小。这有助于解决规模较大的设计的功能和运行时长限制。

命令“set\_floorplan”可以在导入门级网表之前定义“floorplan”命令的规格，宽高比和利用率将被传递给“floorplan”命令。命令“connect”可以创建模块之间的连接关系。命令“set\_ref\_flop”定义了参考触发器单元，用于边界触发器和虚拟触发器。命令“set\_ref\_gate”定义了参考门单元。命令“set\_sai\_version”定义了SAI的版本。

命令“report\_sai\_constraint”可以验证平面图的结果并检查指定的SAI约束是否满足。这使得前端设计人员能够根据指定的平面图规则，轻松评估物理设计人员提供的平面图，无需了解Innovus的使用。

## 2.2 低功耗技术

随着集成电路的不断发展，芯片的功耗管理变得越来越重要。如果一个芯片的功耗过高，芯片耗散的电能主要转化成了热能，容易导致工作时的温度过高，高温不但会引起芯片降频，严重时会造成芯片功能失效或晶体管失效。因此，降低芯片功耗非常重要。

### 2.2.1 数字集成电路的功耗来源

数字IC中的功耗由动态功耗（Dynamic Power）和静态功耗（Static Power）组成，其中，动态功耗又可分为开关功耗（Switching Power）和内部功耗（Internal Power）。CMOS工作电流的示意图如图2.4所示，其中，开关功耗主要来源于外部电容的充放电，与频率、电压以及输出节点的外部电容相关（如图2.4中所示）。内部功耗主要与穿通电流（如图2.4中所示）、晶体管特性、输出节点的电容以及输入信号的转换时间有关。静态功耗又被称为漏电功耗（Leakage Power），主要与器件的漏电流（如图2.4中所示）以及电压和温度相关。

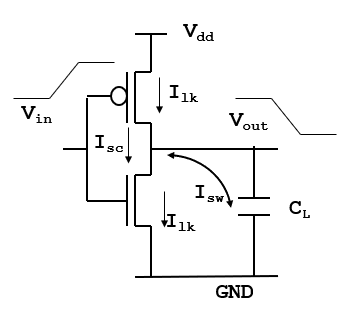


图2.4 CMOS工作电流示意图

数字集成电路的功耗可以用公式(2-1)表示。

 (2-1)

式中，表示静态功耗，表示动态功耗，表示泄漏功耗，表示开关功耗，表示内部功耗。

（1）静态功耗

从90nm工艺开始，静态功耗的管理和优化已经变成集成电路设计过程中必不可少的部分了。如图2.5所示为开关功耗、短路功耗以及漏电功耗在不同工艺的芯片下所占的比例。我们可以看到在130nm工艺下芯片中的漏电功耗只有不足20%，但在45nm工艺下芯片中的漏电功耗已经高于45%了。随着7nm、5nm等先进工艺的应用，静态功耗对于整个芯片功耗的影响越来越大，因此对静态功耗的优化变得越来越重要。

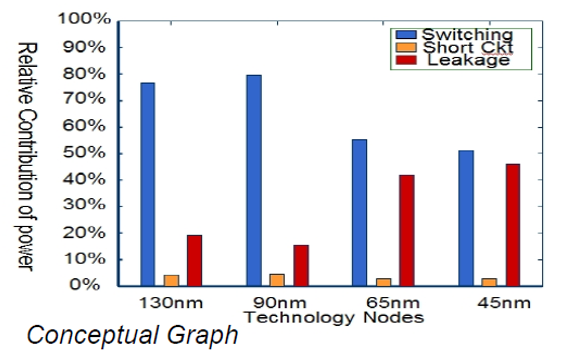


图2.5 不同工艺下三种功耗所占比例

静态功耗的主要影响因素是泄漏电流，而泄漏电流存在的原因是MOS管中的多种掺杂区形成的导电区域，这些区域会组成多个PN节，从而在通电后形成一系列微小的电流。泄漏电流主要由以下几部分组成：反偏结泄漏电流、亚阈值泄漏电流、门栅感应泄漏电流以及栅极泄漏电流。

反偏结泄漏电流：反偏结泄漏电流是反偏结耗尽区少子漂移和电子空穴对产生形成的由源极和漏极流向衬底的电流。器件中的源漏区和衬底之间会产生一个PN结，这样的器件在所加的电压下就会产生相应的二极管反偏电流。它的计算公式如公式(2-2)所示：

 (2-2)

式中，是加在结上的反向偏压，是二极管饱和电流。

亚阈值泄漏电流：当栅极的输入电压小于阈值电压时，因为亚阈值传导会产生静态电流，这时候mos管工作在弱反型区，因此漏极会有电流流向源极，这个电流就称为亚阈值泄漏电流。晶体管越窄，泄漏电流越大。要降低亚阈值电流，可以使用高阈值的器件（HVT），还可以通过衬底偏置技术提高阈值电压。亚阈值泄漏电流的计算公式如公式(2-3)和(2-4)所示：

 (2-3)

 (2-4)

式中，是经验值，n由工艺决定，范围在1-2.5之间，是栅极电压，是漏极电压，是热效电压，是阈值电压，阈值电压的计算公式如公式(2-5)所示：

 (2-5)

式中，为器件的本征阈值电压，它是时的阈值电压，主要与制造工艺有关，为体效应系数，*φF*是掺杂载流子的功函数，是源衬电势差。

门栅感应泄漏电流：门栅感应泄漏电流是由漏极流向衬底的泄漏电流，主要由漏极的强电场引起。目前市面上有多种方法来减小门栅感应泄漏电流，其中包括在等离子体工艺中将栅电极宽度方向两边加厚形成“鸟嘴”状的方法。

栅极泄漏电流：栅极泄漏电流主要是由隧道效应和热载流子效应引起，随着栅氧化层的厚度不断减小，在栅极电压的作用下经过薄栅氧化层流向衬底的电流称为栅极泄漏电流。通过使用高介电常数金属电极可以有效降低芯片的栅极泄漏电流。

（2）动态功耗

动态功耗主要是芯片中的寄生电容充放电引起的。换言之，当芯片中的电路出现任何信号翻转都会产生动态功耗。当把反相器简化成一个简单的RC电路时，就可以看到充放电时的电流走向，当芯片处于工作状态时，每一个工作中的标准单元都会随着时钟和数据信号的翻转不断重复上述过程，从而产生大量动态功耗。

在实际后端物理设计时，由于动态功耗和芯片的功能息息相关，因此在计算的时候会引入翻转率(toggle rate)的概念。在实际计算动态功耗的时候，又会分成两个部分。一部分为标准单元内部的动态功耗，又称为内部功耗，内部功耗的计算方法是，首先在lib库文件中找到对应标准单元的查找表，之后通过标准单元的输入转换和输出负载来查表得到内部功耗的具体数值。另一部分为互连线上的动态功耗，又称为开关功耗。这部分功耗的计算通过将所有互连线上的每个翻转周期的功耗乘以其翻转率并相加得到。翻转率是通过某种固定格式的文件传入EDA工具，比较常用的格式有SAIF（Switching Activity Interchange Format）、VCD（Value Change Dump）以及FSDB（Fast Signal Database）文件。

### 2.2.2 IC低功耗设计技术

目前，低功耗设计已经成为了后端物理设计的主流，功耗分析的结果也已经成为了后端物理设计的重要验收标准之一。提到低功耗设计就离不开UPF，因此在介绍低功耗技术之前首先介绍一下UPF。UPF的全称是统一功耗格式（Unified Power Format），它已经成为了行业内低功耗设计的统一标准，并被三大EDA公司（Synopsys、Cadence、Mentor）的工具所支持。

UPF标准是由Synopsys公司推出的专门用于描述电路电源功耗意图的一种语言标准。在UPF产生之前，数字芯片设计均是采用Verilog语言对电路进行描述，但是这种方式描述出的电路不包含任何的芯片供电网络信息，导致后续的流程很难完成。而UPF的出现成功的解决了这个问题，因为UPF标准使用了大量的描述电源网络的TCL命令，这些命令可以很方便的创建电源域和功耗控制的特殊单元等。UPF编写的统一功耗格式文件不仅可以用在RTL级，同时还可以被后端工具使用，在一定意义上保证了整个芯片设计过程中功耗流程的一致性。在后端工具进行处理之后也会生成相应的UPF文件，此时前端工具可以使用该UPF文件进行Power仿真分析。

（1）多电源多电压技术

在芯片的工作过程中，不同的模块对电压的需求可能会有所不同，例如在SoC芯片中，CPU模块和外设所需的电压大小就有所不同。对于这一类芯片，如果统一采用较高电压的电源供电会导致芯片具有非常高的功耗，如果统一采用较低电压的电源供电可能会导致功能缺失。因此工程师们提出了多电源多电压技术，这种技术会根据芯片中不同模块的电压需求提供不同大小的电压域，这种方式不但保证了芯片正常的电压需求，同时也降低了芯片的功耗。多电源多电压技术的示意图如图2.6所示：

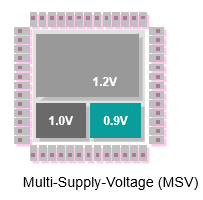


图2.6 多电源多电压技术

提到多电源多电压技术时，就不得不提到电平转换单元了（level shifter）。电平转换单元常用于不同电压域之间的连线上以防止电路中出现亚稳态，特别是对于从低电压的电源域到高电压的电源域的连线上。若有三个模块，第一个模块工作在1.1V的电源域中，第二个模块工作在0.8V的电源域中，第三个模块工作在1.0V的电源域中，并且这几个模块之间均包含连接关系，那么在他们之间的连线上就需要加上电平转换单元，电平转换单元在电路中的示意图如图2.7所示（红色方块代表电平转换单元）。

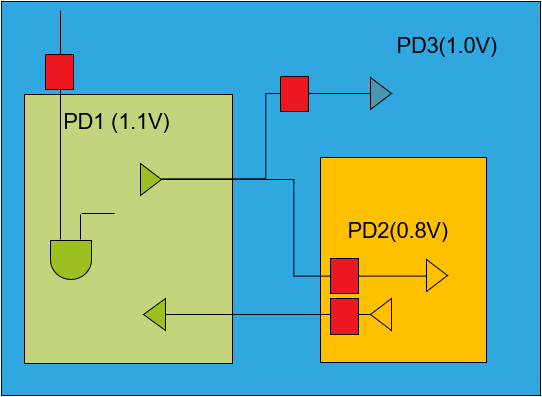


图2.7 电平转换单元应用示意图

（2）门控时钟技术

一个电路的动态功耗是由各部分电路的动态功耗相加得到的，电路中各部分的动态功耗在总功耗中的占比如图2.8所示，其中时钟信号所占的比例最大（50%），这是由于时钟信号的翻转频率最高。其次是IO模块和存储模块（20%），随机逻辑所占的动态功耗比例最小（10%）。由此可见，降低时钟信号的动态功耗对于整体电路功耗的降低大有益处。

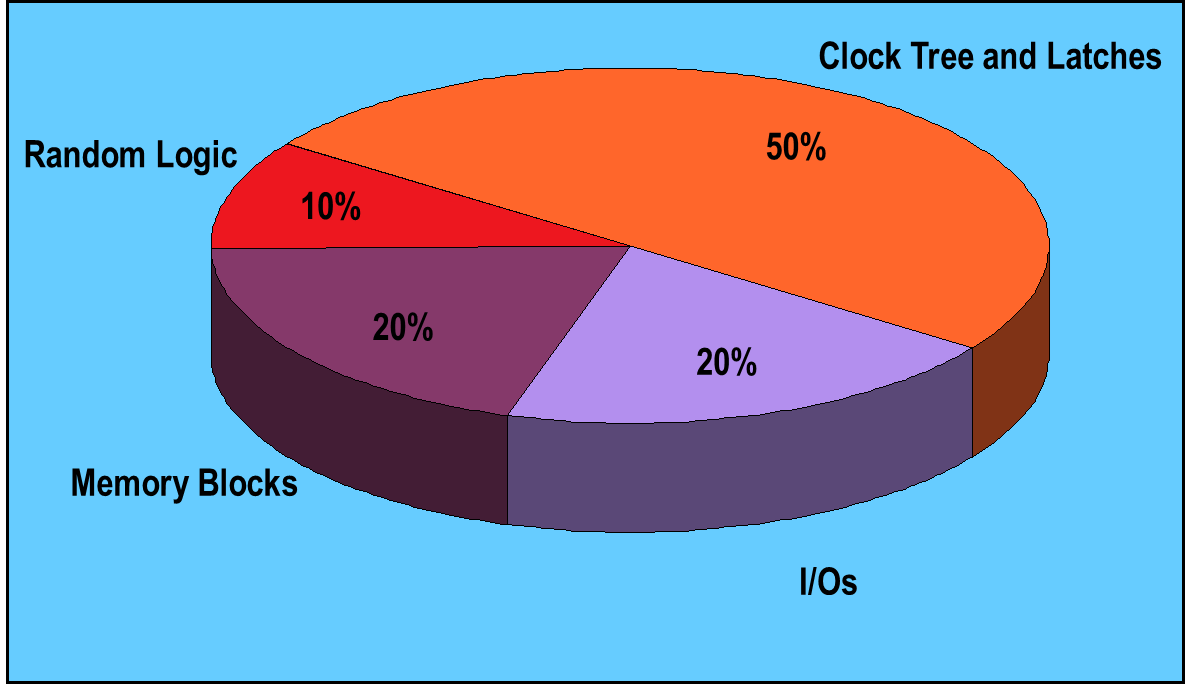


图2.8 动态功耗占比示意图

门控时钟技术的原理是，当电路处于空闲状态时，门控时钟技术会关断对应电路的时钟信号，时钟信号不再翻转，从而降低了动态功耗。门控时钟常用的是基于latch加上与门或者或门的门控时钟，这种门控时钟结构统称为ICG，ICG的结构如图2.9所示。图中enable为ICG的控制信号，当enable信号为高电平时，电路正常工作。当enable信号为低电平，会关断对应电路的时钟信号输入端，避免时钟信号翻转带来的动态功耗。



图2.9 ICG结构示意图

（3）动态电压频率调节技术（DVFS）



图2.10 动态电压频率调节技术

现代的芯片为了追求更高的性能，芯片峰值频率设计的很高，而高频率要求高电压，高电压会带来高功耗，但芯片在实际工作中并非每时每刻都在满负荷工作，这时候节省功耗就变得尤为重要了。因此动态电压频率调节技术应运而生，它可以根据芯片当时的实际需要设定工作电压和时钟频率，这样可以保证提供的功率既满足要求又不会性能过剩，从而降低功耗。如图2.10所示为动态电压频率调节技术示意图。

在调节芯片的工作频率时，电压和频率的调节顺序非常重要。在提高工作频率之前首先要提高芯片的工作电压；降低工作频率时，要在频率降低之后再降低芯片的工作电压。盲目的调节工作频率或工作电压可能会导致芯片的整体功耗提高，并不能实现低功耗的目的。

（4）电源关断技术

电源关断技术的思想是，当芯片的某一区域或是某一模块处于待机或空闲状态时，关闭该区域或者该模块的供电电压，从而达到降低器件的漏电流的目的，进而实现了低功耗。电源关断可以分为内部关断和外部关断两种，内部关断所耗费的时间短但并不能将静态功耗降为零，外部关断耗费的时间长，上电所需的能量较高，但可以将这个区域的静态功耗降为零。对于电源关断的控制上也可分为粗颗粒控制和细颗粒控制，细颗粒控制虽然精准度较高但会带来较大的面积损耗，粗颗粒控制面积损耗小但相对于细颗粒控制来说设计难度较高，在设计中使用较为广泛。如图2.11所示为电源关断技术的示意图。

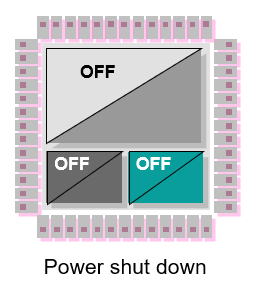


图2.11 电源关断技术

说到电源关断技术就不得不提到相关的几种单元，第一种是电源开关单元（power switch），它是一种特殊的低功耗单元，它的作用是关闭电源域的供电，实现芯片供电控制的目的。

第二种单元是隔离单元（isolation cell），它常用于关闭和打开供电的电源域之间的路径上。由于在关闭供电的电源域中的三极管已经关断了，所以这个三极管连接的外部CMOS电路的输入端是浮空状态，因此可能会导致外部电路的PMOS和NMOS都处于打开状态从而产生短路电流。隔离单元的存在就是为了避免短路电流的出现。

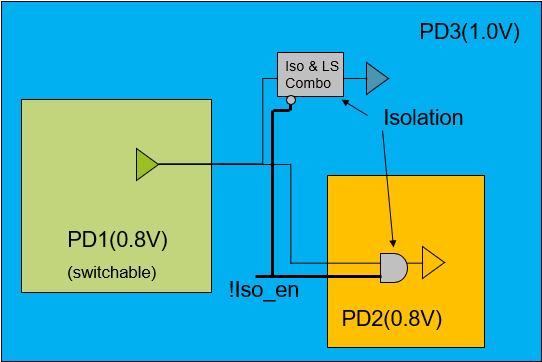


图2.12 隔离单元应用的示意图

隔离单元应用的示意图如图2.12所示，图中PD1为可关断的电源域，供电电压为0.8V，PD2为不可关断的电源域，供电电压为0.8V，PD3为不可关断的电源域，供电电压为1.0V。由于PD1是可关断的电源域，因此PD1连接到PD2和PD3的路径上都要加入隔离单元，而PD1和PD3之间还存在电压差，因此PD1和PD3之间还要加入之前介绍的电平转换单元。

第三种单元是状态保持单元（SRPG），状态保持单元的作用是保存关闭供电的电源域中的寄存器状态，如果不加入状态保持单元，当供电被切断以后寄存器会丢失状态。维持寄存器的状态只需要最小保持电压即可。

（5）多阈值电压技术

为了提高芯片的工作速度，先进工艺中器件的阈值电压不断降低，特别是在深亚微米的工艺下，由于芯片工作电压的降低，器件的阈值电压也随之降低。虽然降低阈值电压可以提高速度，但同时也会提高芯片的漏电流从而增加了芯片的功耗。在一些时序余量充足的路径上使用高阈值器件无疑会增加芯片的功耗，因此，设计人员提出了多阈值电压技术。

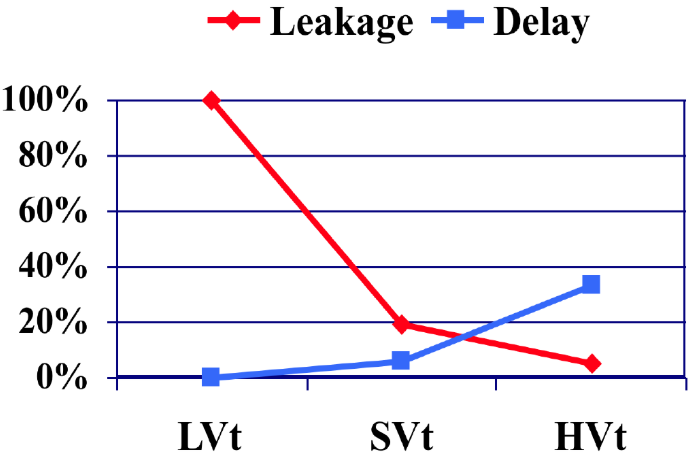


图2.13 不同器件的漏电功耗与延迟

如图2.13所示为三种版本器件的时序和漏电功耗特性。低阈值电压器件的速度更快，延时更小，更有利于时序收敛，但由于低阈值带来的漏电流很大，因此低阈值电压器件的功耗相对来说很高。高阈值电压器件的漏电流很小，功耗很低，但是高阈值电压器件的速度很慢，延时较大，不利于时序收敛。因此在16nm以后的工艺库中，相同功能的标准单元根据阈值电压的不同提供了不同的版本，即高阈值版本（HVT）、标准阈值版本（SVT）以及低阈值版本（LVT）。如表2.3所示为三种版本器件的性能参数比较。

表2.3 不同器件性能参数对比表

|  |  |  |  |
| --- | --- | --- | --- |
|  | 漏电流 | 延迟 | 驱动能力 |
| HVT | 低 | 大 | 弱 |
| SVT | 中 | 中 | 中 |
| LVT | 高 | 小 | 强 |

（6）衬底偏置技术

芯片中器件的亚阈值电流是器件静态功耗的主要影响因素，而器件的阈值电压则决定了器件的亚阈值电流，阈值电压越大，漏电流越小，器件的静态功耗也就越小。为了降低芯片的静态功耗，IC设计者们提出了衬底偏置技术，衬底偏置技术的原理是将PMOS和NMOS的衬底加上偏置电压，从而达到提高阈值电压，进而实现降低静态功耗的目的。衬底偏置技术的示意图如图2.14所示，图中，将PMOS的衬底连接到偏置电压Vbp上，Vbp比Vdd高Ve伏，将NMOS的衬底连接到偏置电压Vbn上，Vbn的电压值为-Ve伏。这样源极和衬底之间的电压差为正值，阈值电压升高，漏电流减小，静态功耗降低。但提高阈值电压会影响电路的性能，因此衬底偏置技术常在电路休眠状态或低频工作状态时使用，可以有效降低静态功耗。

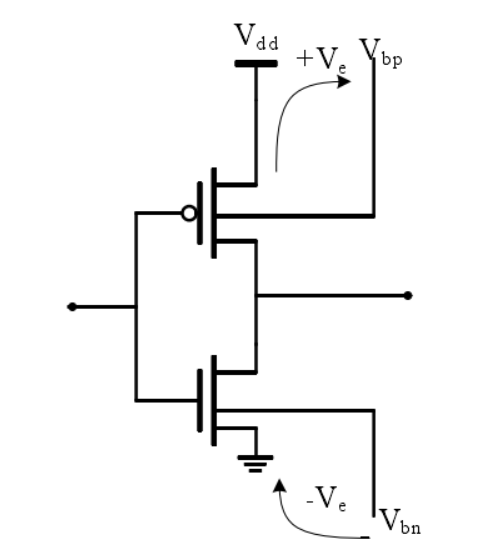


图2.14 衬底偏置技术示意图

## 2.3 本章小结

本章主要介绍了SAI和低功耗技术的相关原理，本章前半部分首先介绍了SAI的相关功能和优势，之后重点介绍了SAI的两种使用模式和使用流程，最后讲解了SAI中常用到的命令。本章后半部分首先对集成电路中的功耗类型以及相关计算公式进行了讲解，之后对集成电路中常用的低功耗技术进行了介绍。常见的低功耗技术有六种，这六种低功耗技术对芯片设计的静态功耗、动态功耗、时序及面积影响各不相同，对芯片设计和验证的影响也不相同。在实际项目中，工程师们需要根据实际设计、项目的情况以及需要解决的具体问题，选择合适的低功耗技术。

# 第三章 混合尺寸布图方法的算法及流程研究

本章主要对混合尺寸布图方法的算法及流程进行研究，混合尺寸布图的特点是可以将宏单元以及标准单元自动摆放。目前市面上依旧在使用传统布图方法进行布图，需要后端工程师手动摆放宏单元，而在多数论文中研究的混合尺寸布图方法存在优化步长不可控、拥塞优化程度不高以及宏单元和标准单元无法同时摆放等问题。本论文研究的混合尺寸布图方法创新性的通过Nesterov、预处理以及步长回溯的方法，不但能够实现自动、同时的摆放标准单元和宏单元，还解决了其他混合尺寸布图方法优化步长不可控的缺陷，同时通过引入静电密度模型和派系模型优化了拥塞及线长等参数。最后将传统布图方法与混合尺寸布图方法的结果进行对比。

## 3.1 **混合尺寸布图方法算法研究**

论文研究的混合尺寸布图方法使用了解析的非线性混合尺寸布图策略，其中构建了一个目标函数，将单元的摆放以及合理化问题转化为求目标函数的最优解，目标函数中包括线长因子以及重叠拥塞因子。目标函数的最优解是指各个单元之间都不存在重叠，线长和拥塞达到极小值的解。由于混合尺寸布图的主要困难仍然是宏单元和标准单元之间的拓扑及物理属性的差异，论文研究的混合尺寸布图方法创新性的利用非线性预处理的方法很好地平衡了它们之间的差异，从而实现了混合尺寸布图。

### 3.1.1 线长的优化

在布图优化之前首先需要一个连线模型来对电路中的连线进行建模，在这个连线模型中需要将含有多个管脚的连线划分为多个双管脚连线。常见的连线模型分为两种：派系模型（clique model）和星型模型（star model）。在派系模型中，一个含有k个管脚的连线会被划分为k(k-1)/2个双管脚连线，而在星型模型中一个含有k个管脚的连线会被划分为k个双管脚连线，但在这些管脚的重心处会插入一个星节点。如图3.1所示为一个含有五个管脚的连线被派系模型划分为十个双管脚的连线，如图3.2所示为一个含有五个管脚的连线被星型模型划分为五个双管脚的连线，并在这五个管脚的重心处插入了一个星节点。本论文研究的混合放置算法使用的是派系模型。

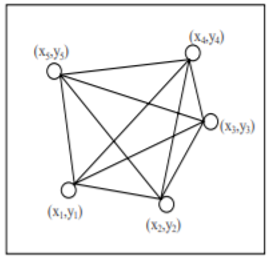
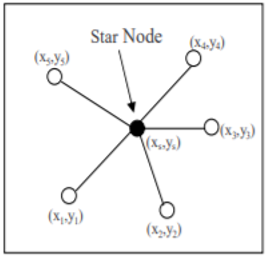
 

图3.1 派系模型 图3.2 星型模型

在VLSI标准单元布图问题中，标准单元布图问题可以表述为超图H(V, E)。其中V是单元集合，V = {v1, v2, v3，…，vn}，E为连线集合，E = {e1, e2, e3，…, em}，坐标(xi, yi)表示单元vi的中心点。整个区域R均匀分成了m × m个矩形网格，记为网格集B。对于每个网格b∈B，*ρb*的密度不应超过预定密度的上界*ρt*，称之为目标密度。布图的目标是将所有的单元都装入设计区域并确定它们的最佳位置，最佳位置要求在满足所有单元之间不存在重叠的条件下连线总长度尽可能小。线长*HPWL(v)*函数可以表述为公式(3-1)。

 (3-1)

式中，i和j是由连线e连接的所有单元中的任意两个。全局布图的非线性优化公式如公式(3-2)所示。

 (3-2)

式中，*ρb*为网格b的密度，*ρt*为预定密度的上界。

通常解析的方法都会使用基于梯度的方法进行布图优化，但由于公式(3-1)中函数*HPWL(v)*不可微，为了保证函数的可微性，因此需要将*HPWL(v)*近似为可微的光滑函数。目前市面上存在大量的目标函数近似方案，论文使用的闭合且光滑的函数W(v)是通过LSE（log-sum-exp）和WA（weighted-average）两种模型近似函数*HPWL(v)*，函数*W(v)*的构成如公式(3-3)、(3-4)、(3-5)、和(3-6)所示。

 (3-3)

式中，表示连线e的线长。

 (3-4)

式中，表示在x轴分量上的长度，表示在y轴分量上的长度。

 (3-5)

 (3-6)

式中，用于控制建模精度，较小的会提高逼近精度，但会降低函数的平滑度，较大的会降低逼近精度，但会提高函数的平滑度。公式(3-6)中的表示x轴分量上的线长的导数。

### 3.1.2 拥塞及单元重叠的优化

为了避免拥塞问题和单元重叠的问题，需要为目标函数中引入密度的模型。现代二次布图工具大都是使用密度力公式，简单来说就是将部分单元从密度过高的区域中拖到密度低的区域，将密度力作为梯度函数中的常数项。而非线性布图工具则是将密度梯度作为一个独立于线长的分量，通过钟形曲线确保密度分布可微且局部光滑，这样的方式容易造成连线过长。论文中使用了基于静电的密度建模方法名为eDensity，相比于其他的方法，eDensity的密度函数密度溢出最小，与目标密度的差异最少，因此这种密度函数的性能最好。

eDensity方法将每个物体都映射为带正电的粒子，将密度函数*N(v)*建模为总电势能。电荷之间的库仑力使所有的电荷彼此分散，最终使总势能趋于零。静电平衡状态与均匀的放置密度分布耦合，最终达到平衡状态。密度函数*N(v)*及其约束如公式(3-7)和(3-8)所示：

 (3-7)

式中，*qi*表示电荷i所携带的电量，其数值等于单元i的面积。*i*为局部电势。当系统能量等于所有电荷对之间的相互势能之和时，每一个电荷的能量是1/2。

 (3-8)

式中，定义的泊松方程将密度分布ρ(x, y)与的势分布关联起来，其中x和y是空间坐标。公式(3-8)强制将密度函数和放置域边界上的梯度设为零来防止物体移动到放置区域R之外。具体来说，沿两个垂直边界的水平密度梯度等于零，沿两个水平边界的垂直密度梯度等于零，这样，朝向放置边界的移动将逐渐减慢并最终停止，这里n是边界∂R处的外法向量。

算法利用ξ(x, y) = ∇来表示电场分布，每个电荷上的电场力等于(v)，其中可以分解为它的水平分量x和垂直分量y，即 = (x, y)。此外，和ξ(x, y)在积分过程中所产生的常数项为零，可以忽略不计。势积分设为零，泊松方程具有唯一解。

非线性优化公式中密度函数的引入将放置物体的均匀分布与静电平衡态联系起来。但是，由于所有的物体都被映射为带正电荷的粒子（设置电量为单元面积），如图3.3所示，作用于所有电荷上的库仑力都是斥力。这种排斥力会不断地把所有的电荷推向无穷大。

另一方面，由于算法禁止将单元移动到放置区域以外，因此在平衡状态下，所有电荷最终都停留在了边界线上，粒子位置的示意图如图3.4所示，因此导致放置区域密度分布不均匀。

为了解决这个问题，算法从空间密度分布ρ(x, y)中去除了零频率（即dc）分量，以便将静电平衡态与均匀的电荷密度分布耦合起来，用*ρavg*表示整体密度分布的dc，用每个网格的原密度*ρb*减去这个*ρavg*。由于dc（ρ(x, y)）等于b的平均网格密度，因此把dc (*ρavg*)从*ρb*中去掉之后，可以得到b (*ρb*) = 0，∀b∈B。

从图3.5中可以看出，除去直流分量后，所有电荷之和为零，从而在低密度区域引入了负电荷，而物体之间除了原有的斥力外，还产生了引力。所有的排斥力和吸引力矢量的组合，引导了放置区域的密度分布朝向如图3.6所示的均匀密度分布转变，在区域内任何地方都没有电荷，系统势能降为零。

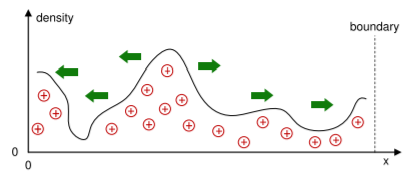
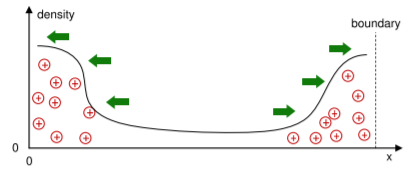
 

图3.3 单元密度及单元受力示意图（1） 图3.4 单元密度及单元受力示意图（2）

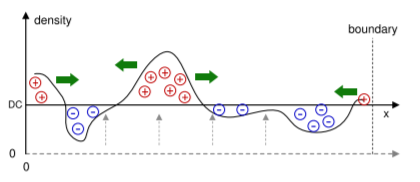
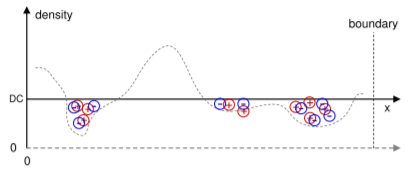
 

图3.5 单元密度及单元受力示意图（3） 图3.6 单元密度及单元受力示意图（4）

密度函数*N(v)*是广义的，*N(v)*的全局光滑性表示任何单元的局部移动都会对整个布图造成影响，因此，所有物体的势能都会因任何单个单元i的运动而改变。

在非线性优化公式中加入密度函数之后，布图优化问题便可定义为如公式(3-9)所示。

 (3-9)

式中，λ是调整长度和密度之间比率的惩罚因子。

## 3.2 混合尺寸布图方法流程研究

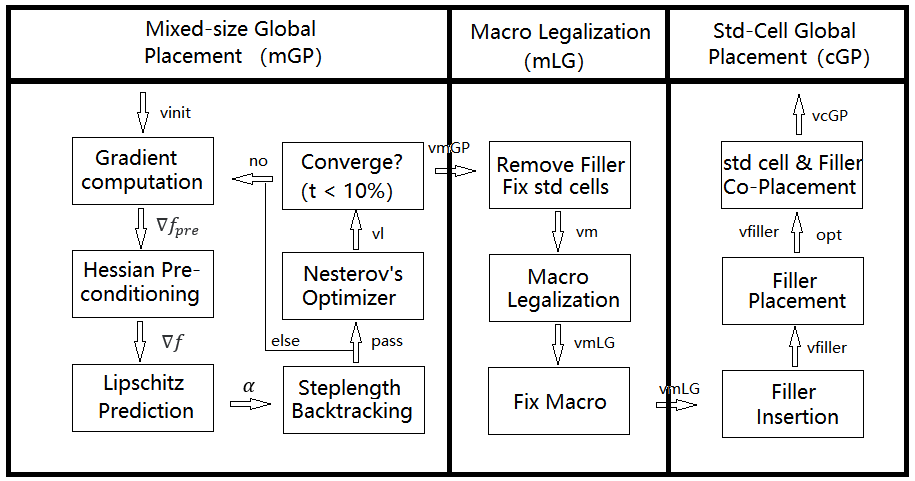


图3.7 混合尺寸布图方法流程图

当目标函数构建完毕后，混合尺寸布图方法就会迭代求解直到收敛。在求解非线性问题的方法上，布图工具在非线性计算中广泛使用的是共轭梯度法，但在线性搜索时不能保证行搜索输出的步长能够满足共轭性要求，所以共轭梯度法的局部收敛速度不能保证，因此，采用动态步长调整的解决方案变得十分必要。布图过程中每次迭代时都会计算梯度和预处理因子，预测利普希茨常数，并通过回溯调整步长。最后，混合尺寸布图方法还会进行宏单元和标准单元优化。混合尺寸布图方法的具体流程如图3.7所示。

论文研究的混合尺寸布图方法首先会进行混合尺寸的全局布图（mGP），混合尺寸初始放置会将总长度最小化，一般来说，初始的放置位置往往具有较低的长度和较高的重叠。之后基于目标网格密度*ρt*，论文研究的混合尺寸布图方法使用没有连接关系的填充单元填充额外的空白，然后使用Nesterov方法迭代地共同优化所有对象(标准单元格、宏和填充单元)，迭代过程中使用利普希茨常数预测和步长回溯来控制优化速度。

在mGP之后，混合尺寸布图方法进入第二阶段进行宏单元合法化（mLG），这一阶段会删除所有的填充单元，修复并固定标准单元的位置，然后调用模拟退火算法来使所有宏单元的位置合法化，然后固定宏单元的位置。

最后进入第三阶段进行标准单元全局优化(cGP)，这一阶段会检索所有填充单元并适当分配它们，然后释放标准单元并将它们与填充单元一起放置优化，进一步减少线长。在cGP阶段最后会将所有的标准单元布局合法化。

混合尺寸布图方法主要的优化工作在混合尺寸布图阶段完成，因为所有单元都允许移动并且可以同时进行优化，优化的程度也最高。mLG和cGP阶段只允许宏单元或标准单元在固定其他对象的情况下小范围移动，从而起到约束作用，这实际上限制了搜索空间，具体来说，就是在局部范围内只会发生很小的位置变化，因此mLG和cGP阶段只能对mGP阶段的布图结果起到优化作用。并且由于光刻问题，论文研究的混合尺寸布图方法不允许旋转或翻转任何物体，但是，它可以灵活地平滑整合旋转和翻转梯度，迭代指导布图优化。填充单元用于平衡全局范围内的静电直流分量，填充单元的总面积等于空白面积乘以目标密度再减去所有可移动单元的总面积。所有填充单元的大小与所有标准单元的平均物理尺寸相同。

### 3.2.1 混合全局布图阶段

mGP阶段的第一步是求出目标函数的梯度，然后对梯度进行预处理。预处理减少了问题的条件数，使之更适合于求解数值。传统的预处理技术是计算目标函数的Hessian矩阵***H****f*的逆。预处理在二次型布图算法中有广泛的应用，但在非线性布图算法中很少使用，主要是由于目标函数的非凸性。因此论文研究的混合尺寸布图方法使用了一个正定对角矩阵作为前置因子来近似原始的Hessian矩阵***H****f*。算法把这个正定对角矩阵的逆乘上梯度向量来指导非线性布图优化。通过预处理的梯度向量把函数空间拉伸得更加近似球形，从而实现了平滑和加快数值优化的作用。然而，由于全局布图的目标函数十分庞大并且高度非线性，因此计算Hessian矩阵变得十分困难，这和实际的计算不切实际。因此，只保留了Hessian矩阵***H****f*的对角项，矩阵的其他项均置为零。具体表达式如公式(3-10)、(3-11)和(3-12)所示。

 (3-10)

 (3-11)

 (3-12)

在mGP阶段采用了利普希茨常数预测和步长回溯控制优化速度。由于使用了Nesterov方法求解非线性方程（Nesterov方法会在后面的部分详细介绍），而当步长*αk*满足如下不等式(3-13)时，Nesterov方法的收敛速度为1/k2，达到了一阶优化方法全局收敛速度的上限。因此为了缩短算法的运行时间，需要在不违反不等式的情况下生成最大步长*αk*。

 (3-13)

有些论文中使用二分搜索的方法生成步长*αk*，这种方法与线搜索类似，它通常会引入大量的运行时间开销。函数 *f (vk−αk∇f (vk))* 将沿着搜索方向进行 O(log L) 次评估，迭代之后会将复杂度增加到 O(n log n log L)，这里L是利普希茨常数。因此，需要步长预测来加速优化过程。首先mGP阶段需要预测利普希茨常数。与线性搜索不同的是，回溯调整步长的方法需要通过梯度的利普希茨常数的封闭式公式计算步长。由于精确的利普希茨常数计算起来非常耗时（甚至比直线搜索更耗时），因此我们使用近似的方法来计算利普希茨常数。这里使用利普希茨常数的倒数作为步长来加速算法，这个数值不但能够满足不等式，并且所选的步长并不会影响函数的收敛速度。利普希茨常数的近似值和步长*αk*如公式(3-14)和(3-15)所示，利普希茨常数的计算开销可以忽略不计，因为 *∇f (vk-1)* 和 *∇f (vk)* 都是已知的，因此没有额外的计算。

 (3-14)

 (3-15)

为了防止步长公式预测出的步长过大导致优化精度降低，论文研究了一种回溯方法，它通过调节步长防止不合理的步长给非线性方程求解带来误差。步长预测公式在预测步长时使用的是*vk*和*vk-1*，而回溯方法使用的是*vk*和*vk+1*，步长回溯算法BkTrk如图3.8所示。

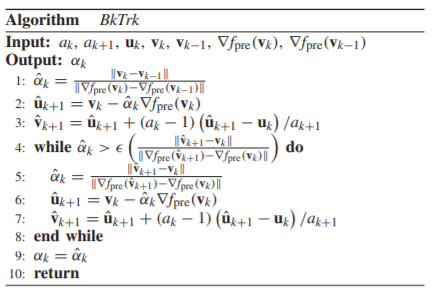


图3.8 BkTrk算法

算法的第一行将步长预测公式计算的步长设为一个临时变量。之后算法会计算*vk+1*的临时解，并利用临时解*vk+1*（第三行）来产生一个参考步长。如果它被步长*αk*超过（第四行），那么算法会在第五行和第七行更新*αk*和*vk+1*，并循环回溯直到第四行不等式得到满足。*vk*和*vk-1*是当前迭代k和上一次迭代k-1的布图解。*uk*是与*vk*同时更新的另一个解决方案（第k次迭代），具体细节会在后面介绍。在算法的第四行设置了缩放因子，其值等于0.95，缩放因子可以减少回溯算法的回溯次数，从而防止算法过度回溯。算法的回溯次数过少会导致步长预测精度过低，而回溯次数过多会消耗过多的运行时间而精度的提升却十分有限。如果第4行第一次检查通过，则运行开销为零，因为新计算的梯度*f(vk+1)*可以在接下来的迭代中重复使用。

得到步长之后，混合尺寸布图工具最后会使用Nesterov算法求得布图解，具体算法如图3.9所示。算法中有两个同时更新的解，*uk*和*vk*，其中只有*u*作为最终解（在mGP和cGP的末尾）输出，而*v*则用于步长预测。∇*fpre*表示预处理的梯度向量。最初，*u0*和*v0*都为初始解。BkTrk表示步长回溯算法。

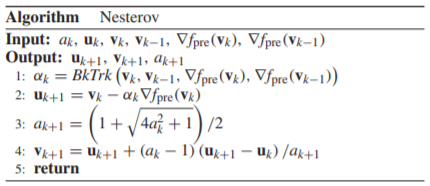


图3.9 Nesterov算法

当重叠区域足够小时终止，我们采用密度溢出τ作为停止判据。当τ≤10%时终止mGP阶段，当τ≤7%时终止cGP。整个布图区域R会均匀分解为m × m个矩形网格，m的数值计算公式如公式(3-16)所示，其中n等于所有可移动单元的数目总和（包括标准单元、宏单元以及填充单元），m数值的上界为1024。惩罚因子λ初始设为10，之后迭代更新mGP中的 = ，以平衡线长和密度，的值如公式(3-17)、(3-18)和(3-19)所示。

 (3-16)

 (3-17)

 (3-18)

 (3-19)

### 3.2.2 宏单元合法化阶段

mLG基于mGP的布图结果，通过模拟退火(SA)的方法使宏单元布局合法化。宏单元的摆放主要依靠mGP阶段提供高质量的解决方案，因为在mLG中每个宏单元只有微小的位置变化。但SA更高效，并且可以很好地探索收缩的设计空间。mLG阶段的目标函数如公式(3-20)所示。

 (3-20)

式中，HPWL(v)、D(v)和(v)分别表示总长度、宏单元覆盖的标准单元的面积和重叠区域的总面积。mLG设置为约束优化，它的主要目的是最小化*HPWL(v) + D(v)*。由于在cGP过程中D(v)的部分会转化为长度，所以我们在mLG中对它们一视同仁，因此静态设置*μD= (HPWL(v)/D(v))*。我们将*μO*设为惩罚因子，初始化为*(HPWL(v) + μDD (v)) / (v)*。在mLG阶段需要将总重叠面积降为零，因此在每次mLG迭代中，会将惩罚因子*μO*乘以*β*，使优化对总重叠面积的减少更加积极。

### 3.2.3 标准单元全局优化阶段

cGP阶段通过全局的标准单元布图优化，减轻了mLG阶段带来的对标准单元布图质量的影响。在固定宏单元的情况下，cGP阶段使用了与mGP阶段相同的非线性算法，但mGP阶段收敛得更快，cGP只会对标准单元布图产生微小的变化。由于mLG阶段不关心填充单元的布图，因此可能会导致大量的宏单元与填充单元重叠。因此cGP阶段首先会检索并随机分配填充单元，在固定标准单元的情况下，将填充单元重新放置到最佳位置上，确保得到的布图结果的质量不会被填充单元所破坏。最后，cGP阶段会将填充单元和标准单元共同优化。

在mGP阶段的最后一次迭代时，会根据mGP阶段惩罚因子的结果确定cGP阶段的初始惩罚因子，即 = × 1.1(m次方)。其中，m代表缓冲迭代次数，对cGP阶段使用m个缓冲迭代来提高优化能力。在cGP的初始阶段时，线长会急剧减少，重叠会急剧增加。随着迭代次数的增加，λ的值不断增加，线长开始缓慢增加，与此同时重叠面积显著减少。在实际应用中，通常将m设为mGP阶段的迭代次数除以10。

## 3.3 混合尺寸布图方法与传统布图方法的质量对比

为了验证论文研究的混合尺寸布图方法在布图质量方面的优势，论文选取了多家设计公司的十个芯片设计项目，将十个设计基于传统布图方法和论文研究的混合尺寸布图方法得到的布图结果进行对比。对比的参数包括WNS、TNS、max transition违例、单元密度、泄漏功耗、线长、最大热点值以及总热点值。另外，本节还在这十个设计中选取了布图结果质量提升程度较大的设计A和布图结果质量提升程度相比于其他九个设计较小的设计B进行详细介绍。

论文基于两种布图方法进行对比的数据报告均来自于INNOVUS软件22.10版本的place opt阶段。为了减少干扰因素同时保证对比结果的准确性，论文在两个流程中删除了一些非共有的优化命令，经过多位专家的确认，这样的数据更具说服力和可信度。

### 3.3.1 设计A的结果对比

首先介绍布图质量提升较大的设计A。这里先比较传统布图方法和论文研究的混合尺寸布图方法的WNS和TNS结果。如图3.10和3.11所示为两种布图方法关于WNS、TNS的报告，我们可以看到基于传统布图方法得到的WNS为-0.555，TNS为-64.910，基于混合尺寸布图方法得到的WNS为-0.187，TNS为-47.143，WNS减少了66.31%，TNS减少了27.37%。

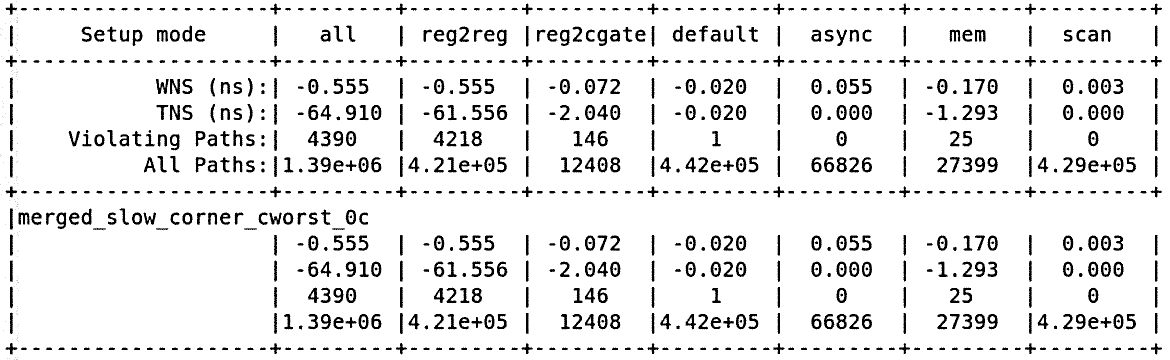


图3.10 传统布图方法WNS、TNS报告

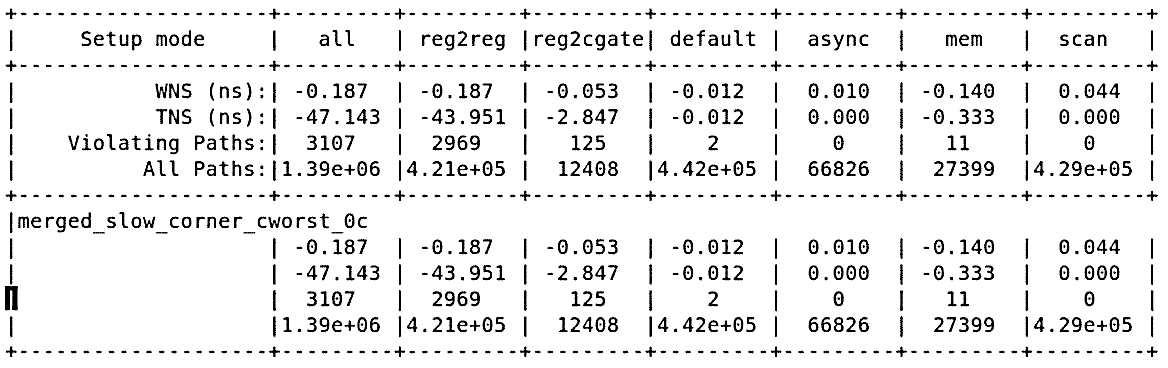


图3.11 混合尺寸布图方法WNS、TNS报告

接下来再比较两种布图方法的max transition违例结果。如图3.12和3.13所示为两种布图方法关于DRV数量的报告，我们可以看到基于传统布图方法得到的max transition的违例数量为7，基于混合尺寸布图方法得到的max transition的违例数量为3，违例数量减少了57.14%。

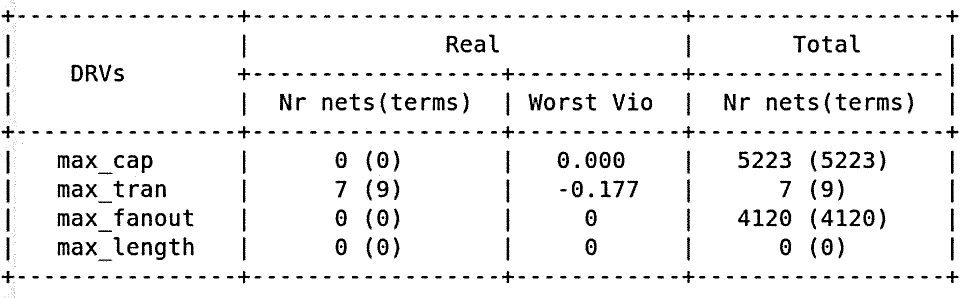


图3.12 传统布图方法DRV数量报告



图3.13 混合尺寸布图方法DRV数量报告

接着再比较两种布图方法的单元密度结果。如图3.14和3.15所示为两种布图方法关于单元密度的报告，我们可以看到基于传统布图方法得到的单元密度为68.356%，基于混合尺寸布图方法得到的单元密度为66.405%，单元密度减少了2.85%。

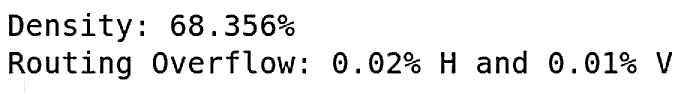


图3.14 传统布图方法单元密度报告

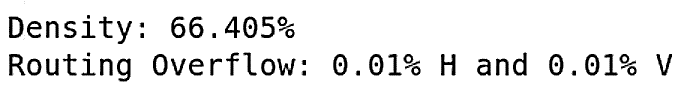


图3.15 混合尺寸布图方法单元密度报告

接着再比较两种布图方法的泄漏功耗结果。如图3.16和3.17所示为两种布图方法关于功耗的报告，我们可以看到内部功耗、开关功耗以及泄漏功耗都有所减少，这里只比较泄漏功耗。基于传统布图方法得到的泄漏功耗为184.3mw，基于混合尺寸布图方法得到的泄漏功耗为182.6mw，泄漏功耗减少了0.92%。

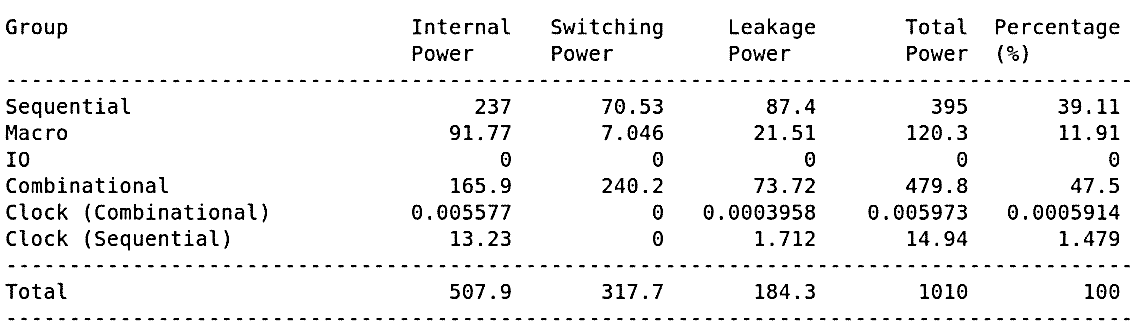


图3.16 传统布图方法功耗报告

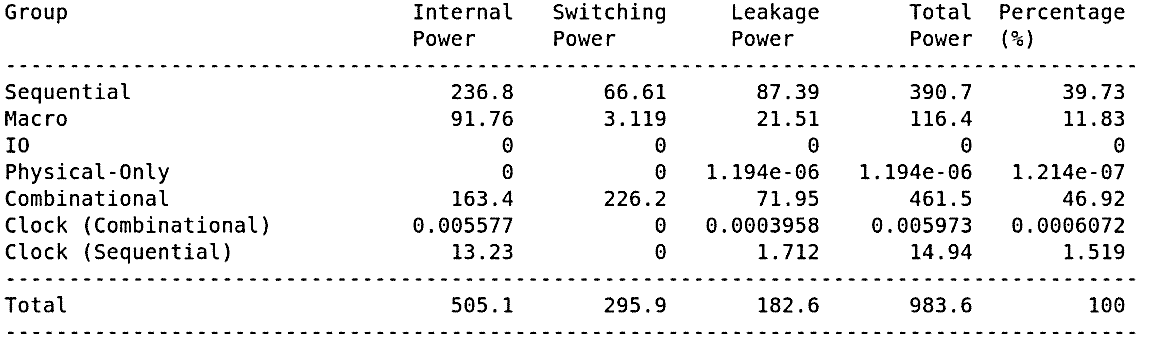


图3.17 混合尺寸布图方法功耗报告

接着再比较两种布图方法的线长结果。如图3.18和3.19所示为两种布图方法关于线长的报告，我们可以看到基于传统布图方法得到的线长为29087360 um，基于混合尺寸布图方法得到的线长为25572417 um，线长减少了12.08%。

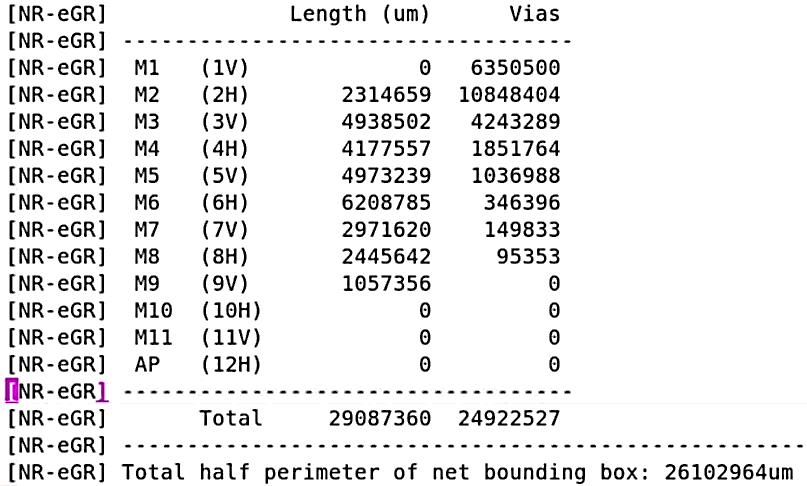


图3.18 传统布图方法线长报告

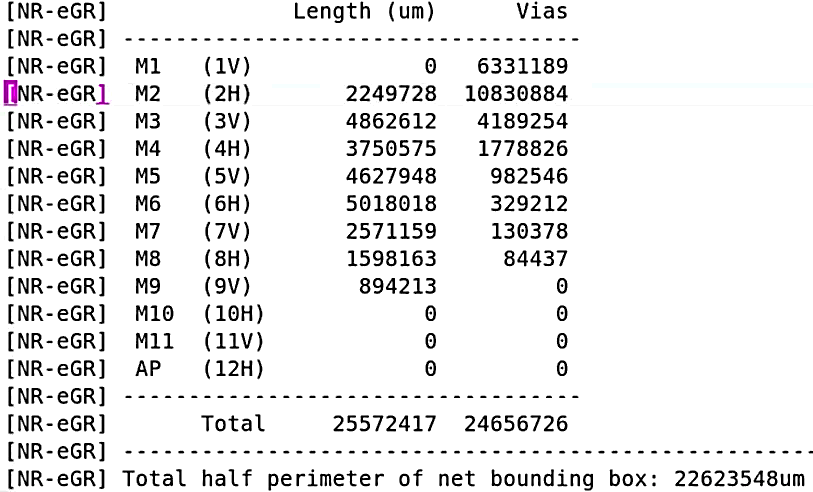


图3.19 混合尺寸布图方法线长报告

接着再比较两种布图方法的拥塞结果。如图3.20和3.21所示为两种布图方法关于拥塞的报告，图中hotspot是指热点，当某一区域出现拥塞时就会出现热点，拥塞越严重，热点值越高。在图中我们可以看到基于传统布图方法得到的最大热点值为0.17，总热点值为0.52，基于混合尺寸布图方法得到的最大热点值为0.17，总热点值为0.17。最大热点值没有减少但总热点值减少了67.31%。

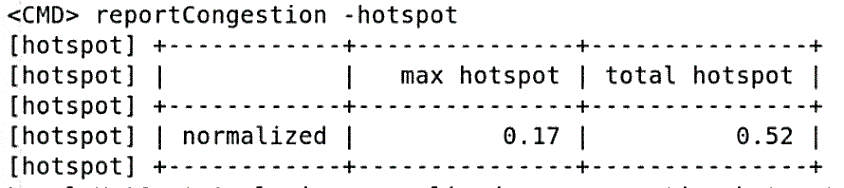


图3.20 传统布图方法热点值报告

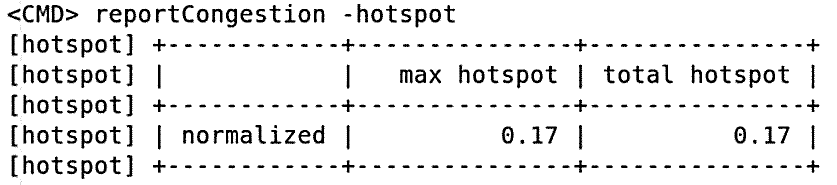


图3.21 混合尺寸布图方法热点值报告

从上文中我们可以看到，基于设计A，混合尺寸布图方法与传统布图方法相比具有非常大的优势。在布图质量上，混合尺寸布图方法相比于传统布图方法，WNS减少了66.31%，TNS减少了27.37%，max transition的DRV数量减少了57.14%，单元密度减少了2.85%，泄漏功耗减少了0.92%，线长减少了12.08%，最大热点值没有减少，但总热点值减少了67.31%，布图质量有明显提升，如图3.22所示为设计A的布图质量提升概况。

图3.22 设计A布图质量提升概况

如图3.23和3.24所示为传统布图方法与混合尺寸布图方法生成的floorplan整体布局图。在图中我们可以很明显的看到宏单元位置的差异，由于传统布图方法都是后端工程师手动调整宏单元位置，宏单元一般摆放在floorplan的边缘位置。而混合尺寸布图方法将宏单元视为优化单位，因此位置不固定，除了一些比较大的宏单元与一些和接口有连接关系的宏单元外，其他宏单元都散落在整个floorplan上。

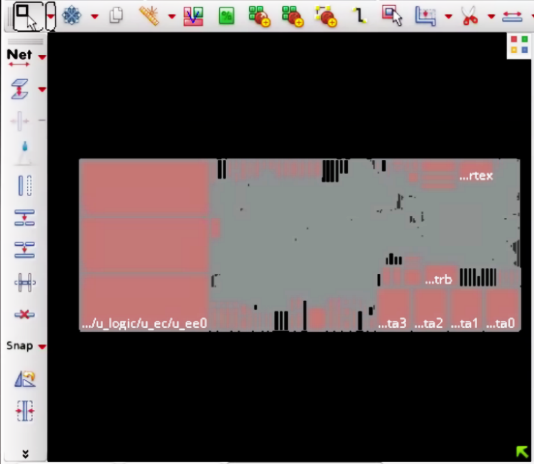
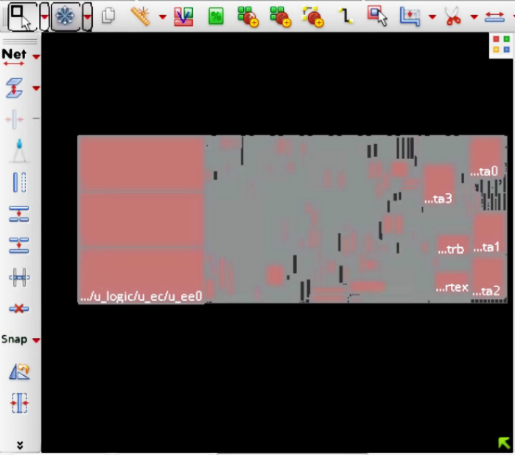
 

图3.23 传统布图方法整体布局图 图3.24 混合尺寸布图方法整体布局图

### 3.3.2 设计B的结果对比

为了证明论文研究的混合尺寸布图方法的普适性，我们在十个设计中选择了布图结果质量提升程度较小的设计B进行详细介绍。

首先比较传统布图方法和论文研究的混合尺寸布图方法的WNS和TNS结果。如图3.25和3.26所示为两种布图方法关于WNS、TNS的报告，基于传统布图方法得到的WNS为-0.257，TNS为-64.136，基于混合尺寸布图方法得到的WNS为-0.140，TNS为-13.914。WNS减少了45.53%，TNS减少了78.31%。

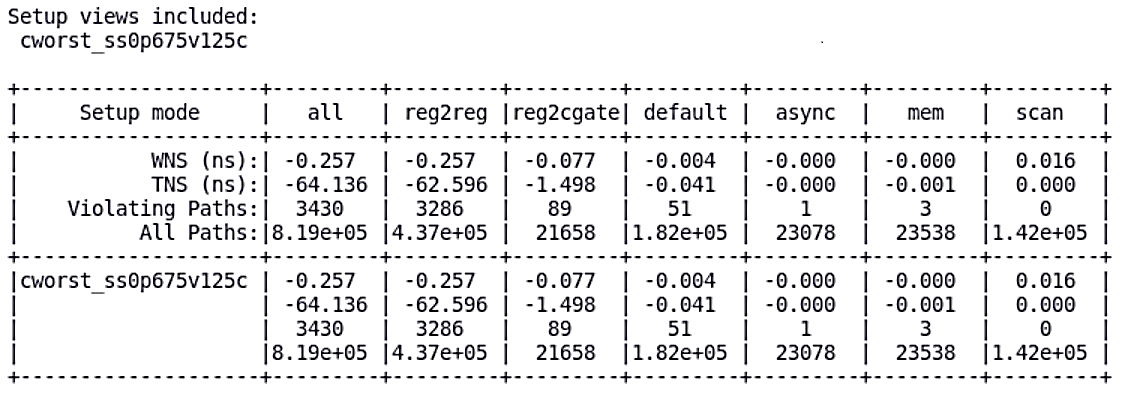


图3.25 传统布图方法WNS、TNS报告

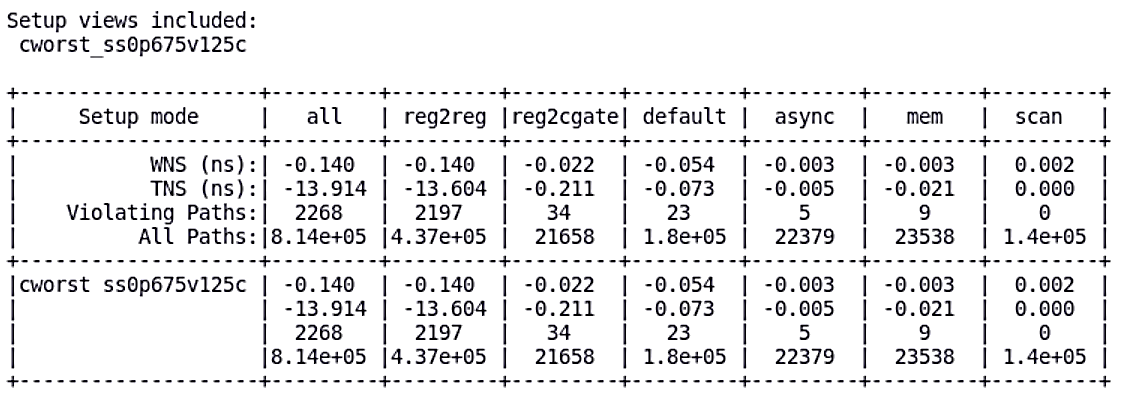


图3.26 混合尺寸布图方法WNS、TNS报告

接下来再比较两种布图方法的max transition违例结果。如图3.27和3.28所示为两种布图方法关于DRV数量的报告，我们可以看到基于传统布图方法得到的max transition的DRV问题的数量为2，基于混合尺寸布图方法得到的max transition的DRV问题的数量为0，max transition的违例都被清除了。

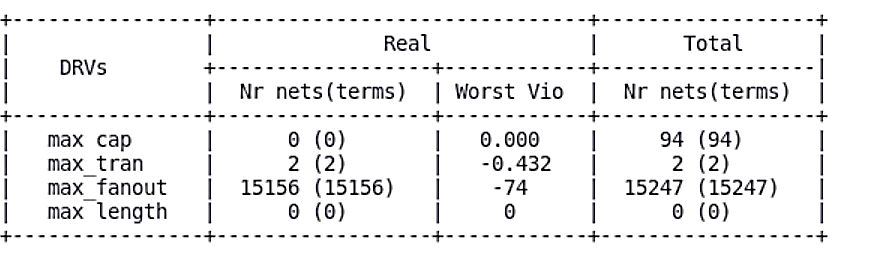


图3.27 传统布图方法DRV数量报告

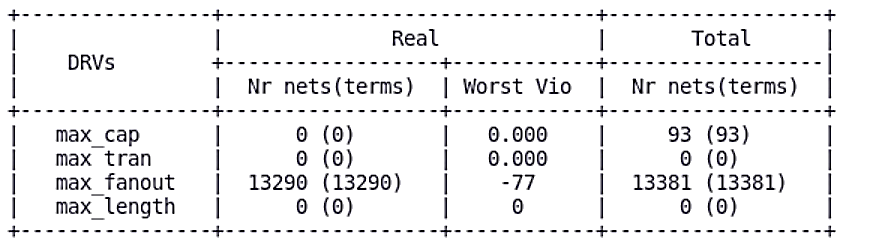


图3.28 混合尺寸布图方法DRV数量报告

接着再比较两种布图方法的单元密度结果。如图3.29和3.30所示为两种布图方法关于单元密度的报告，我们可以看到基于传统布图方法得到的单元密度为65.771%，基于混合尺寸布图方法得到的单元密度为66.776%。单元密度增加了1.54%，单元密度出现了略微的增长，但基本上能和传统布图方法的结果保持在同一个水平。



图3.29 传统布图方法单元密度报告



图3.30 混合尺寸布图方法单元密度报告

接着再比较两种布图方法的泄漏功耗结果。如图3.31和3.32所示为两种布图方法关于功耗的报告，这里只比较泄漏功耗。我们可以看到基于传统布图方法得到的泄漏功耗为6.193w，基于混合尺寸布图方法得到的泄漏功耗为5.952w，泄漏功耗减少了3.89%。

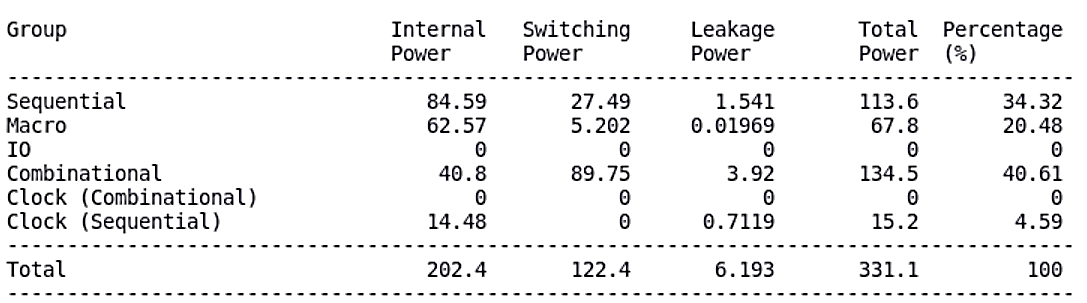


图3.31 传统布图方法功耗报告

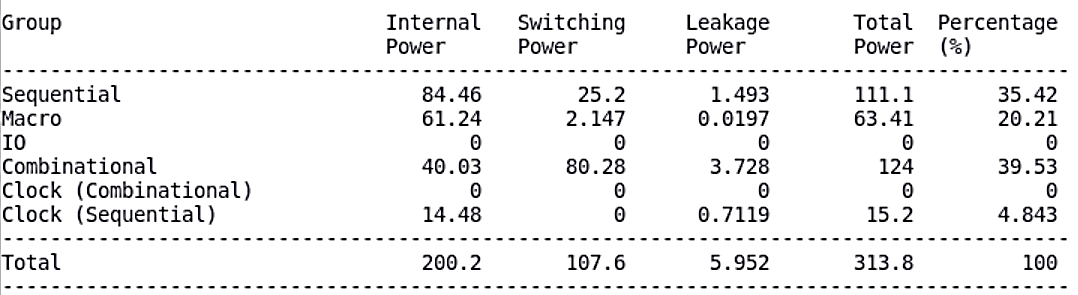


图3.32 混合尺寸布图方法功耗报告

接着再比较两种布图方法的线长结果。如图3.33和3.34所示为两种布图方法关于线长的报告，我们可以看到基于传统布图方法得到的线长为22648546 um，基于混合尺寸布图方法得到的线长为20507931um，线长减少了9.45%。

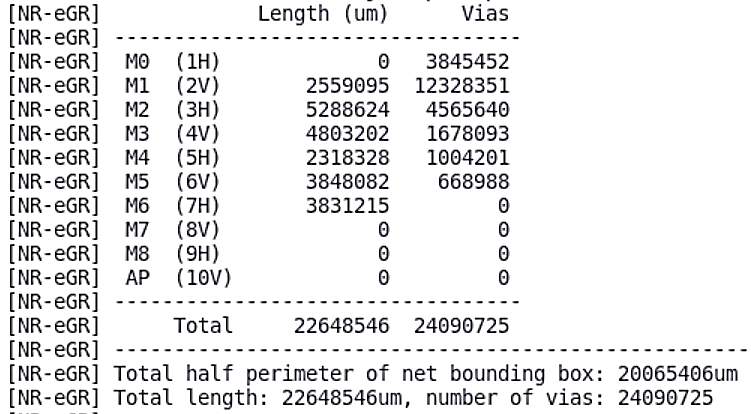


图3.33 传统布图方法线长报告

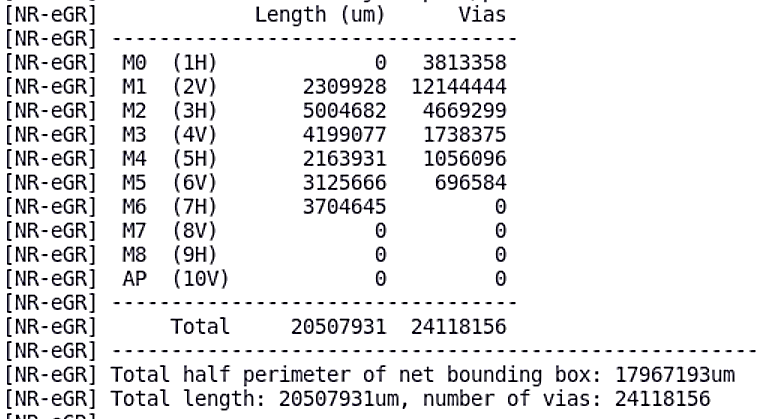


图3.34 混合尺寸布图方法线长报告

接着再比较两种布图方法的拥塞结果。如图3.35和3.36所示为两种布图方法关于拥塞的报告，我们可以看到基于传统布图方法得到的最大热点值为235.01，总热点值为15497.32，基于混合尺寸布图方法得到的最大热点值为325.68，总热点值为13362.51，最大热点值增加了38.58%，但总热点值减少了13.78%。

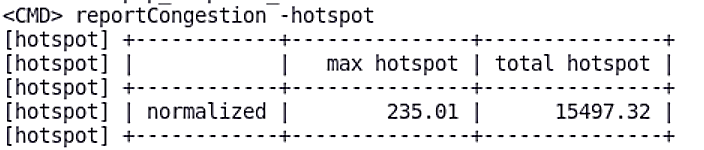


图3.35 传统布图方法热点值报告

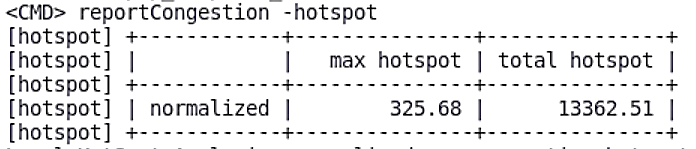


图3.36 混合尺寸布图方法热点值报告

在设计B上我们依然能看到混合尺寸布图方法的优势。除了在布图时间上的优势外，在布图质量上，混合尺寸布图方法相比于传统布图方法，WNS减少了45.53%，TNS减少了78.31%，max transition的DRV数量减少了100%，单元密度增加了1.54%，在泄漏功耗减少了3.89%，线长减少了9.45%，最大热点值增加了38.58%，总热点值减少了13.78%。如图3.37所示为设计B的布图质量提升概况。

图3.37 设计B的布图质量提升概况

整体来看，虽然单元密度有所增加，但差距可以忽略。虽然最大热点数值有所增加，但这样的情况也在合理范围内，因为全局的拥塞情况得到了缓解并且WNS和TNS优化程度非常高。一个设计中的各项指标之间的关联和制约程度很高，由于WNS和TNS得到了大量的优化，因此影响到了最大热点参数的优化。如图3.38和3.39所示为传统布图方法和混合尺寸布图方法生成的floorplan的整体布局图。

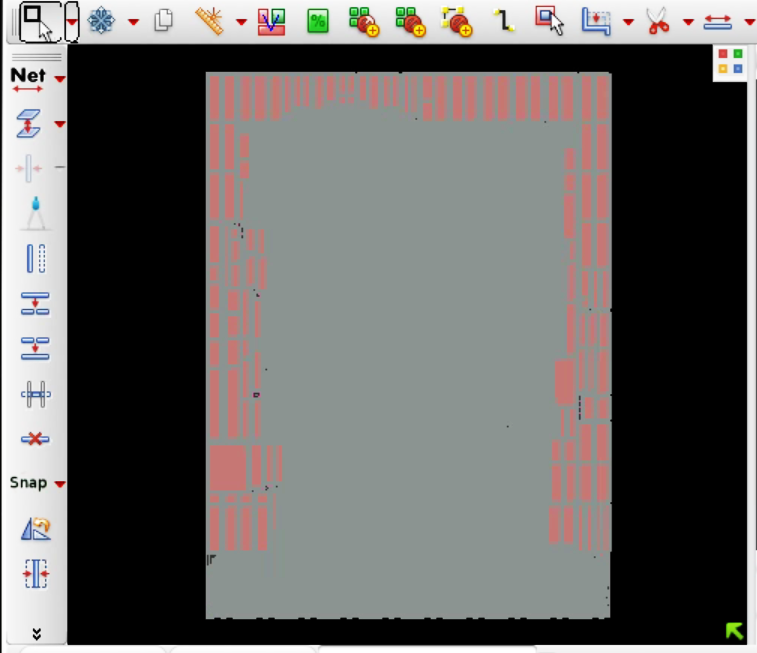
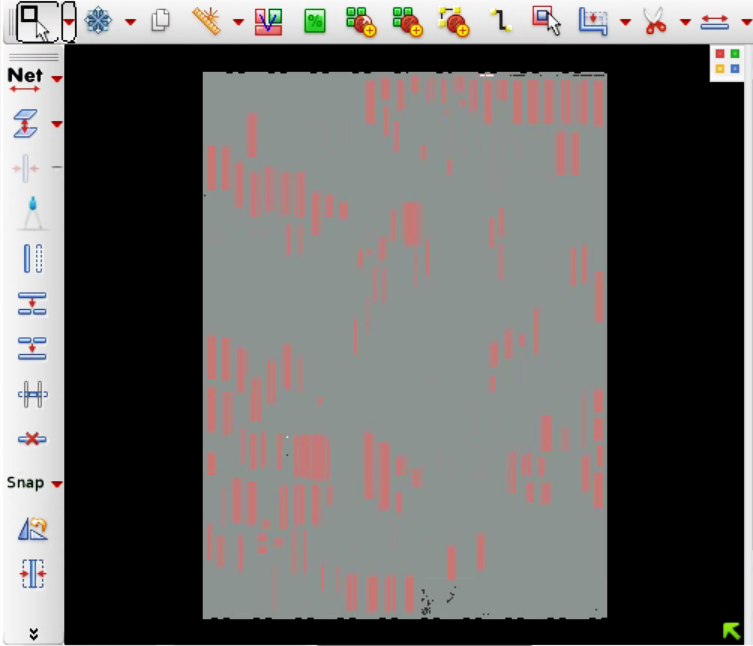
 

图3.38 传统布图方法整体布局图  图3.39 混合尺寸布图方法整体布局图

### 3.3.3 对比与总结

表3.1 质量参数对比表（1）

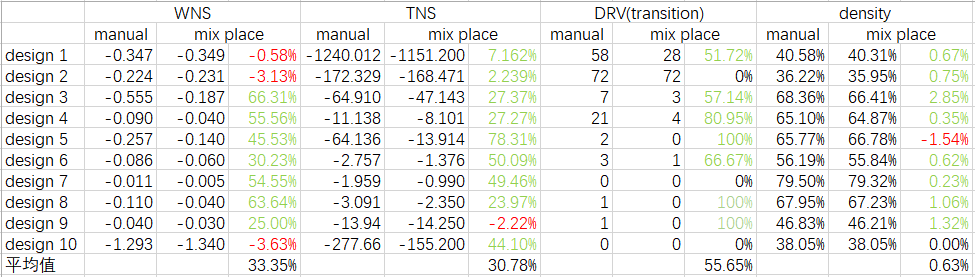


表3.2 质量参数对比表（2）

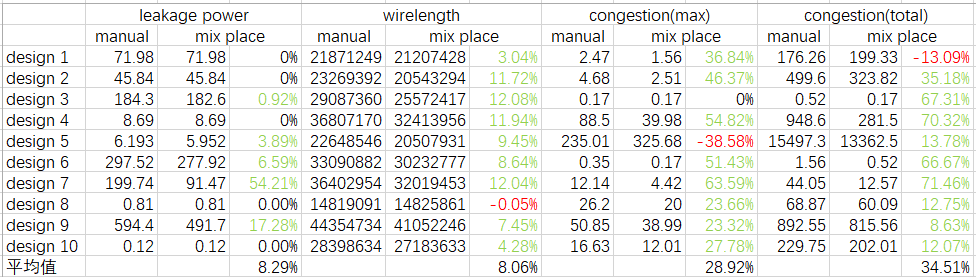


表3.1和3.2所示为论文选取的十个设计基于传统布图方法和论文研究的混合尺寸布图方法进行布图后的参数对比。混合尺寸布图方法相比于传统布图方法，WNS平均减少了33.35%，TNS平均减少了30.78%，max transition 的DRV数量平均减少了55.65%，单元密度平均减少了0.63%，泄漏功耗平均减少了8.29%，线长平均减少了8.06%，最大热点值平均减少了28.92%，总热点值平均减少了34.51%。由此可以看出本论文研究的混合尺寸布图方法在布图质量方面的巨大优势。目前已有芯片项目使用了混合尺寸布图方法并成功流片。

## 3.4 混合尺寸布图方法与传统布图方法的TAT对比

TAT（Turn Around Time）的意思是运行时间，对于芯片后端物理设计来说，直接关系着项目tape out的时间，因此非常重要。如图3.40所示为选取的十个设计的宏单元位置确定后布图阶段的TAT对比，经过对比与计算，混合尺寸布图方法相比传统布图方法TAT平均缩短了25.7%。

图3.40 TAT对比

## 3.5 本章小结

本章主要介绍了混合尺寸布图方法使用到的混合放置算法及流程。混合尺寸布图方法主要分为混合全局布图、宏单元合法化和标准单元全局优化三个部分。在混合全局布图阶段主要进行宏单元和标准单元的布图。在宏单元合法化阶段和标准单元全局优化阶段主要对宏单元及标准单元进行优化及合法化。最后将混合尺寸布图方法与传统布图方法的结果进行对比。实验证明，本论文研究的混合尺寸布图方法WNS平均减少了33.35%，TNS平均减少了30.78%，max transition 的DRV数量平均减少了55.65%，单元密度平均减少了0.63%，泄漏功耗平均减少了8.29%，线长平均减少了8.06%，最大热点值平均减少了28.92%，总热点值平均减少了34.51%，布图质量有明显的提升。除此以外，混合尺寸布图方法不需要手工布图，并且TAT比传统布图方法缩短了25.7%，需要的人力成本更少，耗时更短。

# 第四章 电源线模型优化研究

本章主要研究了电源线模型的优化策略。旧电源线模型只能计算每一层电源线密度的平均值，无法记录同一层电源线的分布情况，计算结果不够精确。因此，论文使用了独立统计的策略研究出了电源线优化模型的计算公式。它可以分别计算并记录宏单元区域和标准单元区域电源线的密度及分布情况，同时还会参考OBS和blockage的情况，最后对电源线密度的差异进行补偿或标注。本节的最后分别使用两种电源线模型进行布图，实验证明，电源线优化模型解决了混合尺寸布图方法在特殊条件下布图质量下降的问题。

## 4.1 电源线模型

### 4.1.1 电源线模型的应用

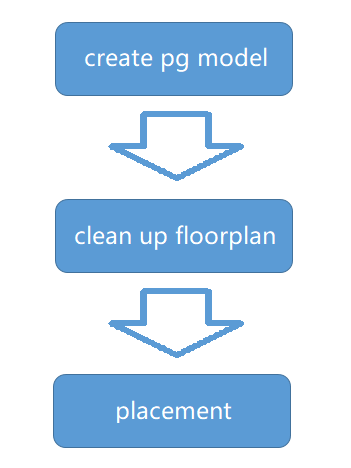


图4.1 自动布图流程

如图4.1所示为自动布图的流程，混合尺寸布图方法在布图之前需要先保存电源线模型，清空floorplan上的单元和连线。保存电源线模型的目的是在清空floorplan之前保存电源线占用的布图绕线资源情况。由于在此之前用户已经为他们的设计添加了电源线，而电源线的存在会占用一部分资源，清空floorplan之后无法获悉电源线占用资源的情况，因此需要根据电源线所消耗的资源创建一个电源线模型。电源线模型能够帮助混合尺寸布图方法准确预估剩余的绕线资源，电源线模型越精确，对资源的预估也就越准确，布图的质量也就越高。

### 4.1.2 电源线模型存在的问题

通过对上述混合尺寸布图方法的应用和测试，发现无论在布图时宏单元和标准单元的摆放效率方面还是在布图质量方面都带来了非常大的提升。为了验证布图方法的可靠性，我们在大量真实的设计中进行了实验并比对。在大多数情况下，混合尺寸布图方法都能够得到一个非常良好的结果，但是在少数情况下，布图质量并没有明显的提升，这些设计的QoR和参考流程的结果相差无几甚至更差。为了弄清楚这些较大差异的由来，我们又做了大量的实验，最终发现是在应用过程中使用的电源线模型不够准确导致了混合尺寸布图方法对绕线资源的估算上出现了较大的差异，进而影响了最终的布图质量。

在优化前的电源线模型中，模型所使用的公式如公式(4-1)所示，其中pg\_density代表的是电源线的密度，stripes area代表电源线所占的面积，spacing代表电源线两边保留的空间面积，user routing blockage代表用户在设计中添加的绕线阻塞区域的面积，core area代表整个设计的核心区域的面积。这个模型在某些情况下会出现预估资源不准确的情况。。

 (4-1)

如图4.2所示，为某一款芯片的版图，在左边的图中灰色的方块代表宏单元，其余的深色区域都是标准单元的区域，宏单元上红色的线条代表了电源线。我们可以看到，在第一个图中，电源线基本上只存在于宏单元上。第二个图与第一个图是同一个芯片的不同层，在第二个图中，浅色部分为宏单元，红色部分是标准单元区域（由于上面布满了密度很高的红色电源线，因此看起来是红色的区域），显然电源线基本上只存在于标准单元区域。

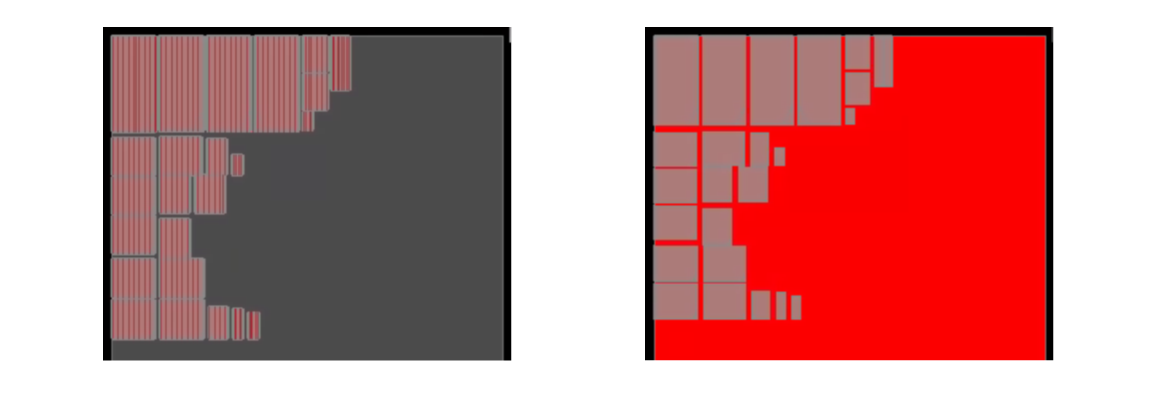


图4.2 电源线分布示例

如果用优化前的电源线模型进行建模，对于电源线在整个设计中均匀分布的情况，电源线在标准单元区域和宏单元区域密度相差不大，因此利用优化前的模型得到的标准单元区域和宏单元区域的电源线密度就不会和标准值之间产生较大的偏差。但对于如图4.2所示的第一种和第二种情况，由于优化前的模型只计算平均值，并不考虑电源线在标准单元区域和宏单元区域的分布，因此得到的标准单元区域和宏单元区域的电源线密度就会和标准值之间产生较大的差异。因此需要对电源线模型进行优化。

## 4.2 电源线模型优化研究

电源线模型的优化针对旧电源线模型的缺陷，重新设计了电源线模型的计算公式，使得每一层电源线的分布情况都能被准确的记录。

### 4.2.1 电源线密度计算

在电源线优化模型中，模型的建立共分为两步，第一步是计算电源线的密度，第二步是根据标准单元区域和宏单元区域电源线密度的不同，在相应的区域进行标注和补偿。在第一步计算电源线密度的时候，模型会将标准单元区域的电源线密度和宏单元区域的电源线密度分开考虑。标准单元区域的电源线密度公式和宏单元区域的电源线密度公式如公式(4-2)和(4-3)所示。标准单元区域的电源线密度等于标准单元区域电源线的面积加上电源线两边spacing的面积加上用户在标准单元区域添加的绕线阻塞面积，之后除以整个设计的面积减去宏单元的面积。宏单元区域的电源线密度等于宏单元区域电源线的面积加上电源线两边spacing的面积加上用户在宏单元区域添加的绕线阻塞面积，减去宏单元区域绕线阻塞和OBS重合的面积，最后除以宏单元的面积减去OBS的面积（OBS代表单元的不可布线区域）。

 (4-2)

 (4-3)

值得注意的是，在计算标准单元和宏单元区域的电源线密度时，还要考虑用户在floorplan上的对应区域添加的绕线阻塞以及OBS的面积。在统计宏单元面积时还要加上halo的面积。在统计电源线的面积时，除了电源线本身的面积之外还需要计算电源线两边spacing的面积。如图4.3所示为电源线spacing的示意图，在图中可以看到，蓝色的部分为电源线，在电源线两边的红色部分为电源线的spacing。spacing是半导体工艺的要求，spacing的数值一般定义在工艺文件中。

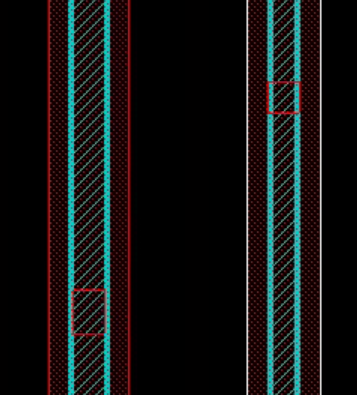


图4.3 spacing示意图

如图4.4所示为补偿电源线密度差异的流程图。在图中可以看到，第二步补偿和标注差异之前，首先要对比标准单元区域和宏单元区域的电源线密度的差异。如果标准单元区域的电源线密度和宏单元区域的电源线密度差异小于0.02，则认为两个区域的电源线密度一致；如果标准单元区域的电源线密度小于宏单元区域的电源线密度，则在宏单元区域添加OBS以补偿电源线密度的差异；如果标准单元区域的电源线密度大于宏单元区域的电源线密度，则在宏单元区域添加screen以标注电源线密度的差异。

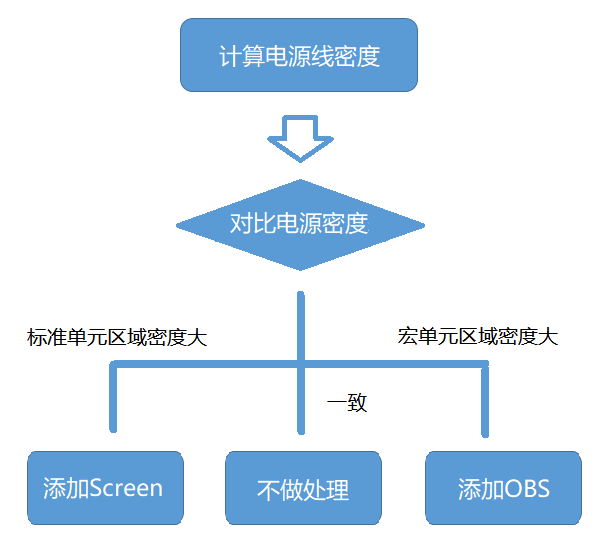


图4.4 补偿电源线密度差异流程图

### 4.2.2 OBS的计算

如果宏单元区域的电源线密度和标准单元区域的电源线密度相差不超过百分之二，则不需要任何操作。如果宏单元区域的电源线密度大于标准单元区域的电源线密度，就需要在宏单元区域生成OBS来补偿电源线密度的差异。OBS在版图上以多个条状的方式产生，条状OBS的高度为宏单元的高度。条状OBS数量的计算公式如公式(4-4)所示；条状OBS之间间距的计算公式如公式(4-5)所示；条状OBS宽度的计算公式如公式(4-6)所示。

 (4-4)

 (4-5)

 (4-6)

需要产生的条状OBS的数量等于宏单元区域的电源线密度和标准单元区域的电源线密度的差值乘以宏单元的宽度再除以三倍的最小线宽。产生的OBS的宽度等于计算得到的OBS的数量乘以最小线宽再除以OBS的真实数量，这个真实值为计算得到的OBS的数量向下取整的数值。Pitch为OBS之间的间距，它的数值为宏单元的宽度减去OBS的真实数量与OBS宽度的乘积，再除以OBS的真实数量减1。产生的OBS的示意图如图4.5所示，网格状的OBS为原设计中包含的部分，它被定义在库文件中，深蓝色的部分为标注电源线密度差值所产生的OBS，他们之间的宽度为pitch。

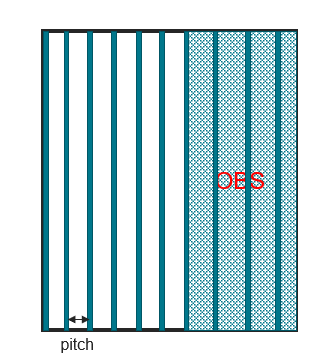


图4.5 产生的OBS示意图

产生了OBS之后，宏单元区域和标准单元区域电源线密度的差异就补偿成功了。在预估绕线资源时就可以使用标准单元区域的电源线密度而不会在宏单元区域产生误差。当布图结束之后，这些用于补偿电源线密度差值的OBS会被删除，因此用户在操作界面看不到它们。

### 4.2.3 Screen的计算

如果标准单元区域的电源线密度大于宏单元区域的电源线密度，模型会在宏单元区域生成screen来标注电源线密度的差异。screen的密度公式如公式(4-7)所示。screen的密度计算很简单，它等于标准单元区域的电源线密度和宏单元区域电源线密度的差值加一。screen不会产生在设计当中，它只会作为screen density被保存下来，在混合尺寸布图方法预估宏单元区域的绕线资源时，帮助工具更准确的计算剩余的绕线资源。有了screen density后，在预估绕线资源时就可以使用标准单元区域的电源线密度而不会在宏单元区域产生误差。当布图结束之后，screen density会被删除。

 (4-7)

## 4.3 电源线模型及布图结果对比

电源线模型计算完成后会将结果保存在文件中，文件的内容是以“set\_macro\_place\_constraint”命令的形式进行保存，目的是便于后端布图布线工具能够直接读取电源线模型的内容。在后端工具中创建电源线模型使用的命令是“create\_pg\_model\_for\_macro\_place”。电源线优化模型保存的格式如公式(4-8)和(4-9)所示，分别为某种宏单元需要添加的OBS信息和screen信息。OBS的信息包括金属层名称、OBS的宽度以及OBS的间距。screen的信息包括金属层名称以及screen的密度。

set\_macro\_place\_constraint -cells $cell\_name –cell\_obs {{$layer\_name $obsWidth $pitch}} (4-8)

set\_macro\_place\_constraint -cells $cell\_name –track\_adjusting {{$layer\_name $screen\_density}} (4-9)

如图4.6所示为电源线模型保存的结果示例。图中一共有三部分，第一部分是优化前的电源线模型保存的结果，在图中可以看到，优化前的电源线模型保存的是每一层的电源线模型密度，不区分标准单元区域和宏单元区域。图中的第二部分是电源线优化模型产生的OBS的结果，相比于优化前的电源线模型，优化后的模型每一条“set\_macro\_place\_constraint”命令都只涵盖一种宏单元，命令后面记录了需要添加OBS的金属层名称以及这些层的OBS宽度和OBS之间的间距。由于每一层的电源线情况和最小线宽不同，因此每一层添加的OBS间距也不同。图中的第三部分是电源线优化模型产生的screen的结果，每一条“set\_macro\_place\_constraint”命令都只涵盖一种宏单元，命令后面记录了需要添加screen的金属层名称以及这些层需要添加的screen的密度。

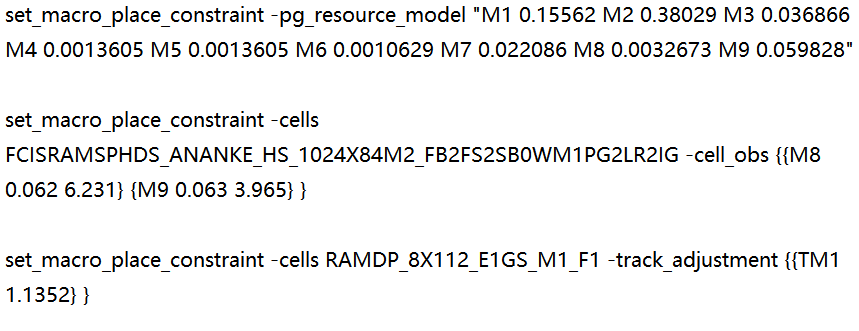


图4.6 电源线模型结果示例

电源线优化模型计算的结果更准确，优化后的公式中除了计算电源线密度外，还包含了设计中OBS以及routing blockage的资源占用情况；电源线优化模型保存的有效信息更多，只保存了和标准单元区域电源线密度有差异的宏单元区域的名称、层次以及OBS或screen的相关数据；电源线优化模型可以补偿或标注电源线密度的差异，电源线优化模型会根据电源线密度差异的大小在对应层的宏单元区域生成OBS或screen。

表4.1 电源线模型优化前后设计质量参数对比表

|  |  |  |  |
| --- | --- | --- | --- |
|  | manal | old pg model | new pg model |
| WNS | -0.09 | -0.07 | -0.04 |
| TNS | -11.138 | -10.267 | -8.101 |
| DRV(tran) | 21 | 7 | 4 |
| density | 65.10% | 65.03% | 64.87% |
| leakage power | 8.69 | 8.69 | 8.69 |
| wirelength | 36807170 | 36435787 | 32413956 |
| max hotspots | 88.5 | 63.6 | 39.98 |
| total hotspots | 948.6 | 732.9 | 281.5 |

为了验证电源线优化模型对布图质量的提升，本节选择了第三章中所选择的十个设计中的design4进行验证（因为design4的电源线分布差异较大），布图质量数据如表4.1所示。从表中我们可以看到，优化了电源线模型后，design4的WNS可以额外减少33.34%，TNS可以额外减少19.45%，max transition 的DRV数量可以额外减少14.28%，单元密度可以额外减少0.24%，线长可以额外减少10.93%，最大热点值可以额外减少26.68%，总热点值可以额外减少47.58%。电源线模型优化前后优化提升对比图如图4.7所示。

图4.7 电源线模型优化前后优化提升对比图

## 4.4 本章小结

本章主要介绍了电源线模型的优化策略。电源线优化模型不同于优化前的电源线模型，它将标准单元区域的电源线密度和宏单元区域的电源线密度分开计算，并且在标准单元区域和宏单元区域的电源线密度有差异时进行补偿或标注，帮助布图工具更准确的预估绕线资源，提高布图质量。电源线优化模型具有计算结果更准确、保存的有效信息更多以及可以补偿或标注电源线密度差异等特点。

# 第五章 早期门级网表生成器设计

本章主要研究了早期门级网表生成器的流程以及实现方式，提出了基于SAI工具的优化策略。为了解决生成的门级网表粗糙的问题，加入了自动添加连接关系、自动添加组合逻辑以及模拟真实设计分配单元的功能。为了解决自动化程度低、便利性差和依赖宏单元库文件的问题，优化了设计参数控制方式，加入了层次化器件自动生成、单元查找和宏单元自动生成的功能。同时，加入了UPF自动生成的功能，提供了快速支持低功耗设计的途径。最后，对早期门级网表生成器与SAI工具生成的同一结构的门级网表进行对比。

## 5.1 早期门级网表生成器流程

本论文基于Cadence公司的INNOVUS产品的SAI工具进行优化，设计了一款早期门级网表生成器，其具有便利性高，仿真度高，生成的门级网表质量高等特点。它克服了市面上其他工具生成的早期门级网表不包含组合逻辑和连接关系、与真实设计结构差异大、支持的工艺和设计种类有限以及使用难度高等缺陷。如图5.1所示为早期门级网表生成器的流程图，流程中共有三个阶段。

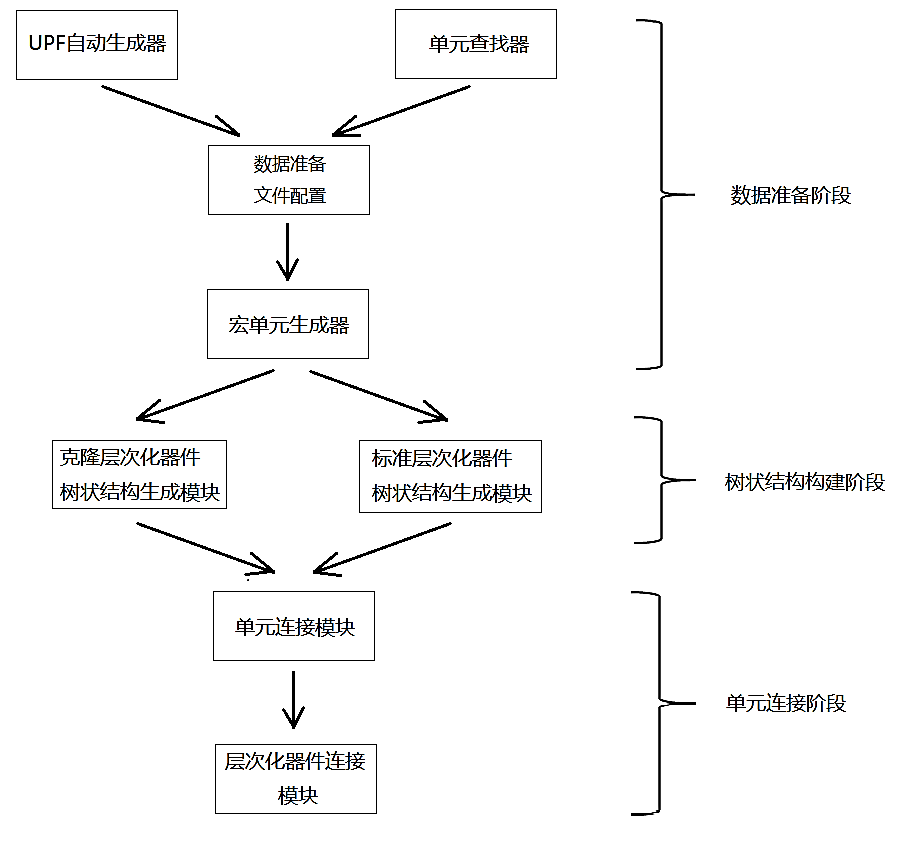


图5.1 早期门级网表生成器流程图

数据准备阶段的主要目的是准备门级网表生成所必需的文件以及配置设计参数。早期门级网表生成器只需要修改配置文件中的数据即可，避免了SAI工具使用命令配置的繁琐步骤。早期门级网表生成器在数据准备阶段配有三个辅助工具，其中，单元查找器可以在库文件中查找标准单元的名称，宏单元生成器可以根据需求定制宏单元，UPF自动生成器可以帮助生成UPF文件。

树状结构构建阶段的主要目的是生成用户需要的层次化器件结构，并模拟真实设计分配标准单元和宏单元。在层次化器件规划以及单元分配完成后，树状结构生成模块会调用SAI工具的相关命令生成层次化器件。相比于SAI工具，早期门级网表生成器自动化程度更高，能够实现自动生成层次化器件。

单元连接阶段的主要目的是将添加的标准单元和宏单元按照一定的规律进行连接，并将层次化器件按照层次化结构进行连接。单元连接阶段主要由单元连接模块和层次化器件连接模块构成。早期门级网表生成器克服了SAI工具生成的门级网表不包含连接关系和组合逻辑的缺陷。

## 5.2 数据准备

数据准备阶段解决了SAI工具控制设计参数困难和使用难度高等问题。早期门级网表生成器只需要修改配置文件中的参数和文件即可控制生成的门级网表。

### 5.2.1库文件及单元选配

早期门级网表生成器需要用户提供一些文件和单元名称，如图5.2所示为门级网表配置文件的库文件及单元的选配部分：



图5.2 库文件及单元选配

图中lef\_file是指库文件，用户可以在这里更换不同工艺的库文件。图中还可以看到三种宏单元和触发器、缓冲器、与门、异或门以及或门的配置选项。用户可以使用宏单元生成器产生需要的宏单元。当不熟悉单元名称时可以使用单元查找器查找库文件中符合要求的标准单元。三个辅助工具会在后面详细介绍。

### 5.2.2 修改设计参数

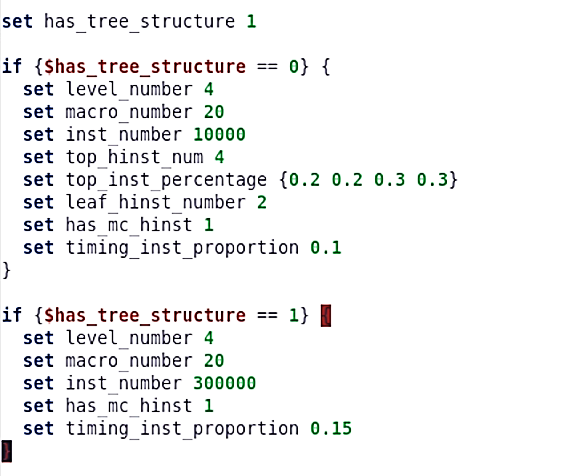


图5.3 设计参数配置

如图5.3所示为门级网表配置文件的设计参数配置部分。早期门级网表生成器提供了两种器件结构生成方式，分别是默认方式和自定义方式。

当变量“has\_tree\_structure”设为0时，代表默认方式，需要在门级网表配置文件中指定以下八个参数：

1、门级网表的层次数量。

2、门级网表的宏单元数量。

3、门级网表的标准单元数量。

4、是否包含克隆层次化器件。

5、触发器在标准单元中所占的比例。

6、层次化器件包含的子器件数量。

7、顶层层次化器件的数量。

8、顶层层次化器件的标准单元数量比例。

当变量“has\_tree\_structure”设为1时，代表自定义方式，只需要在门级网表配置文件中的对应位置指定前五个参数即可。

在配置的参数中，门级网表的层次数量表示层次化器件结构包含的层级关系数量；宏单元数量和标准单元数量表示在生成的门级网表中包含的标准单元以及宏单元的总数；触发器在标准单元中所占的比例会传递给后续步骤，当工具给层次化器件添加标准单元时会依据这个参数添加触发器。

层次化器件包含的子器件数量是指在树状结构中，每个节点在下一层包含的叶节点的数量。用户设定了顶层层次化器件数量的同时也要设定这些器件所包含的标准单元的数量所占的比例，如若没有设定或者只设定其中的一部分，工具会将剩余的标准单元平均分配在顶层中未标注数量比例的器件上。只有当用户使用默认的结构生成方式时才需要设定这几个参数。

## 5.3 数据准备辅助工具

### 5.3.1 宏单元生成器

由于在门级网表的生成过程中，库文件中可能不包含宏单元，因此早期门级网表生成器附带了一个宏单元生成器，它能够帮助用户定制宏单元。定制过程中可以根据需求产生相应数量的pipeline和接口数量，然后经过布图、时钟树综合、绕线以及优化等后端物理设计步骤生成出来。

（1）宏单元属性配置

在使用宏单元生成器之前，用户需要在宏单元配置文件中配置一些宏单元基本信息并添加所需文件。如图5.4所示为宏单元配置文件。在填写所需文件时，除了所需的mmmc和sdc文件以外，还需要提供lef库文件。“hardmacro\_bus\_width”代表宏单元的总线宽度，即输入输出端口数量，“hardmacro\_pipe\_stage”代表每条路径上的pipeline数量。最后提供一个触发器和一个缓冲器名称即可。

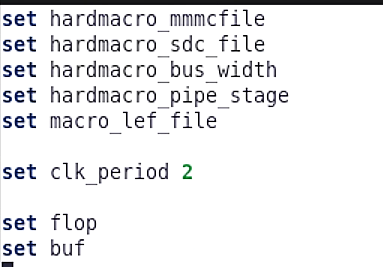


图5.4 宏单元配置文件

（2）宏单元的生成

宏单元生成器在运行过程中会自动读取用户的配置信息并生成宏单元。首先生成宏单元的门级网表，再重新启动INNOVUS进行后端物理设计。宏单元生成器会帮助用户生成三种形状的宏单元：长宽比为一比一的宏单元、长宽比为二比一的宏单元以及长宽比为一比二的宏单元。如果用户对宏单元的长宽比有特殊的要求，也可以在宏单元生成器的脚本中手动修改。

如图5.5所示为宏单元的生成步骤，它用到了大量的后端物理设计理论知识。首先读入库文件和sdf等文件，然后在空白的设计中添加触发器和缓冲器两种标准单元，由它们组成多级pipeline。接下来进行布图工作并连接电源线，之后工具会进行时钟树综合和时钟树绕线，最后进行信号线绕线并优化DRC和时序问题。做完所有的工作后，抽取并保存宏单元的lef文件，早期门级网表生成器读入宏单元的lef文件后就可以在门级网表中添加对应的宏单元了。由于门级网表配置文件中设置了三种不同长宽比的宏单元，因此整个过程需要重复循环三次才能完成。

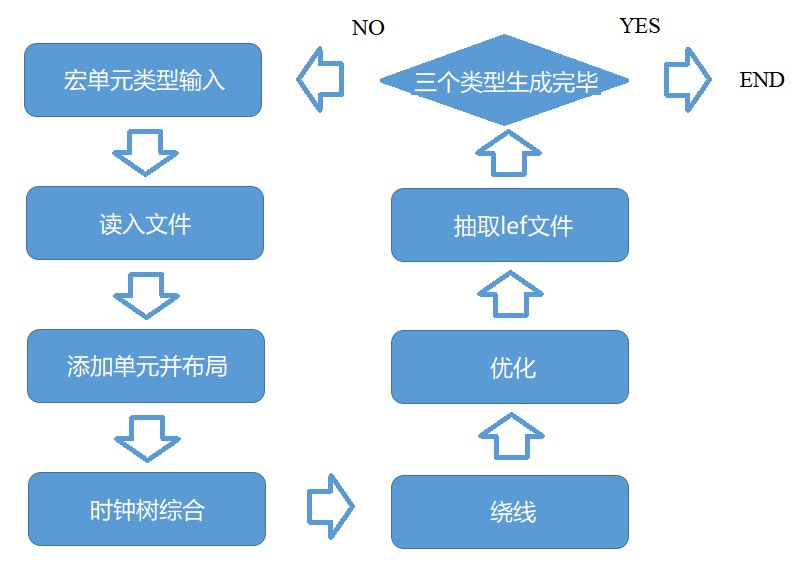


图5.5 宏单元生成步骤

### 5.3.2 UPF自动生成器

编写UPF文件需要对IEEE1801的低功耗设计规则和命令非常熟悉，但即使非常有经验的后端工程师编写UPF文件依然要花费大量的时间。如果遇到结构复杂的UPF文件或编写过程中人为的疏忽需要修改，则需要花费更多的时间。为了帮助用户快速生成UPF文件，论文为早期门级网表生成器设计了一个UPF自动生成器。

UPF自动生成器简单易用，只需修改UPF配置文件即可，UPF自动生成器的算法会自动根据配置文件的内容生成UPF文件。UPF自动生成器的应用范围十分广泛，它不但可以用在早期布图规划阶段，还可以帮助前端工程师进行Power仿真分析，还可以帮助后端工程师进行低功耗物理设计。

（1）准备配置文件

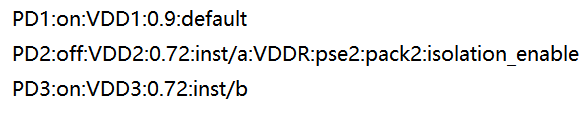


图5.6 UPF配置文件示例

如图5.6所示为一个UPF配置文件的示例。以第二行为例，“PD2”代表这个电源域的名称，“off”代表这个电源域是可关断的，“VDD2:0.72”代表电源域的电源线及其电压值，“inst/a”代表这个电源域所涵盖的器件名称，“VDDR”代表连入这个电源域的不会被关断的电源线，“pse2”代表电源开关的控制线，“pack2”代表电源开关的ack端口的连线，“isolation\_enable\_2”是隔离单元的信号线。第一行为默认的电源域的配置。第三行为不会关断的电源域的配置，由于它不会被关断，因此不需要添加电源开关。

（2）生成电源域

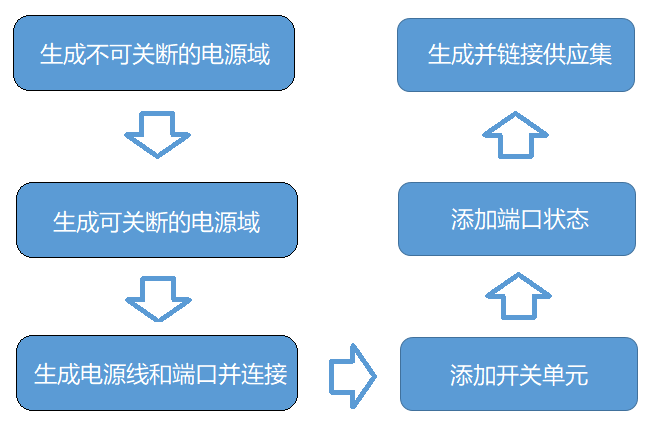


图5.7 电源域生成步骤

如图5.7所示为算法生成电源域的步骤。算法将可关断的电源域和不可关断的电源域分开生成，这样可以避免算法后续添加低功耗单元时再次检索，提高了算法的运行速度。算法首先生成不可关断的电源域，再生成可关断的电源域，之后算法将电源域的电源线和电源端口产生并连接。

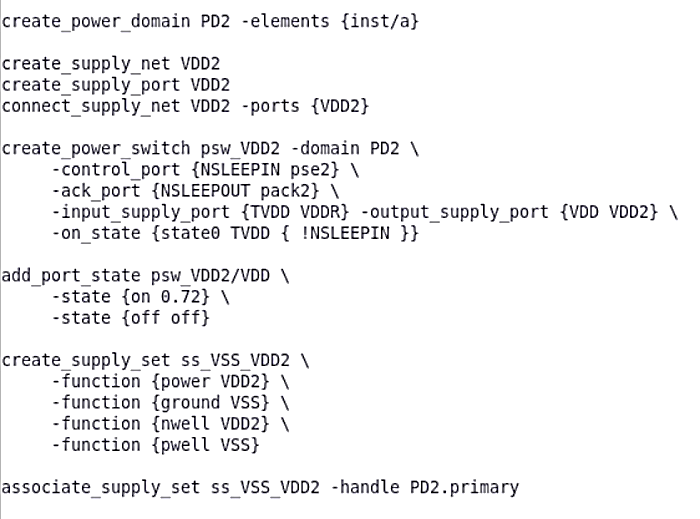


图5.8 电源域生成结果示例

如果这个电源域是可关断的，接下来就需要为这个电源域添加电源开关并添加端口状态，最后生成供应集并链接供应集。如图5.8所示为电源域生成的结果示例。算法运行完毕后会将生成的结果保存在新创建的空白UPF文件中。

（3）生成低功耗单元

最后，需要为可关断的电源域添加低功耗单元，算法需要在电源域之间寻找并判断需要添加低功耗单元的路径。在不同供电电压的电源域之间的路径上需要添加电平转换单元，在可关断到不可关断的电源域之间的路径上需要添加隔离单元。

表5.1 电源状态组合表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| state\supply | VDD1 | VDD2 | VDD3 | VSS |
| STATE1 | 1(ON) | 1(ON) | 1(ON) | 1(ON) |
| STATE2 | 1(ON) | 0(OFF) | 1(ON) | 1(ON) |
| STATE3 | 1(ON) | 1(ON) | 0(OFF) | 1(ON) |
| STATE4 | 1(ON) | 0(OFF) | 0(OFF) | 1(ON) |

UPF文件最后需要添加PST（Power State Table），这个状态表的作用是描述各个电源线的开关组合。由于UPF生成器无法获悉设计中实际的组合状态，因此工具会帮助用户生成每一种组合，用户可以根据实际需要在生成的文件中进行删减。如表5.1所示为电源状态组合示意表，其中VDD2和VDD3是可关断的，因此便有四种组合状态。

为了算法能够最终产生所有的组合状态，这里将表中的开关状态用0和1表示。观察可关断电源的状态不难看出，状态的表述近似于二进制的加和，状态的数量也等于，n代表可关断电源的数量。因此算法在状态组合的问题上使用了二进制的叠加，二进制中每一位代表一个电源线的状态，0和1代表开关状态，每叠加一次就会产生一种组合状态。

低功耗单元和电源状态组合生成完毕后，算法会将结果保存在上一步创建的UPF文件中。

### 5.3.3 单元查找器

使用早期门级网表生成器时需要提供五种标准单元，分别是两输入与门、两输入或门、两输入异或门、缓冲器以及触发器。在数量庞大的库文件中进行筛选并非易事。因此，论文为早期门级网表生成器准备了单元查找器，它可以在库文件中快速找出符合要求的单元名称，用户只需要在名称列表中进行筛选填入配置文件即可。后端工程师在进行后端物理设计时，如果需要用到不清楚名称的单元，也可以使用单元查找器进行查找，非常方便。

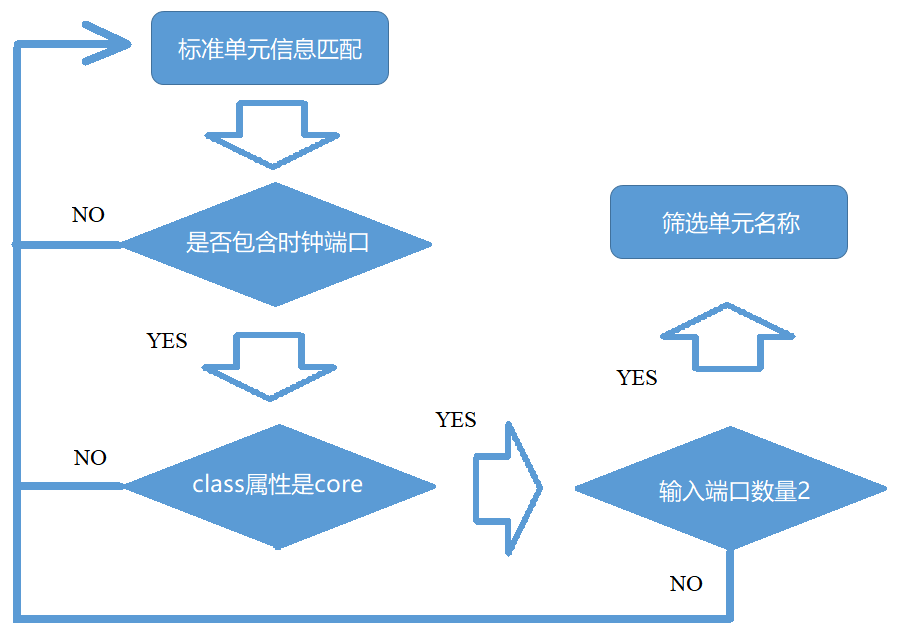


图5.9 单元查找器实现思路

如图5.9所示为单元查找器的实现思路。首先根据需要筛选出是否包含时钟接口的标准单元，然后在剩余单元中筛选出class属性为“core”的单元，由于配置文件中只需要两输入的单元，因此在剩余单元中排除信号输入端口不等于二的单元。经过这样的操作就保存了所有类型的门单元以及部分其他单元，由于所有单元都混在一起，因此还需要进一步的筛选。经过调研和观察，我们发现在大多数库文件中，与门的名称中都包含“and”或“AND”的关键词，或门的名称中都包含“or”或“OR”的关键词，异或门的名称中都包含“xor”或“XOR”的关键词，根据这样的规律就可以在剩余的单元名称中找出二输入与门、或门以及异或门。

缓冲器和门单元的特点一样，需要筛选出没有时钟输入端口并且class属性为“core”的单元。但缓冲器除了电源端口以外只有一个输入端口一个输出端口。在筛选单元名称时，在剩余的名称中筛选出包含“buf”或“BUF”关键词的单元即可。触发器和门单元有所不同，它的定义中存在时钟端口，因此查找的单元需要包含时钟接口，最后筛选单元名称中包含“dff”或“DFF”关键词的单元即可。

## 5.4 标准层次化器件树状结构生成模块

标准层次化器件树状结构生成模块（后面统称为结构生成模块）的主要功能是构建用户所需的标准层次化器件树状结构。首先结构生成模块会梳理层次化器件并为每一个层次化器件命名，之后根据一定的规则为层次化器件分配宏单元和标准单元，最后使用SAI工具的相关命令生成。结构生成模块不但高度模拟了真实设计的单元分布结构，并且成功解决了SAI工具无法自动生成层次化器件的问题。

如图5.10所示为论文开发的提供给用户使用的“create\_empty\_module”结构生成模块。在使用时需要提供层次数量、是否包含克隆器件以及时钟周期三个参数。它会先规划两种层次化器件并分配单元，之后调用命令“create\_macro”生成标准层次化器件。生成完毕后“create\_empty\_module”模块会调用克隆层次化器件树状结构生成模块“create\_empty\_mc\_module”，调用时需要提供层次数量。

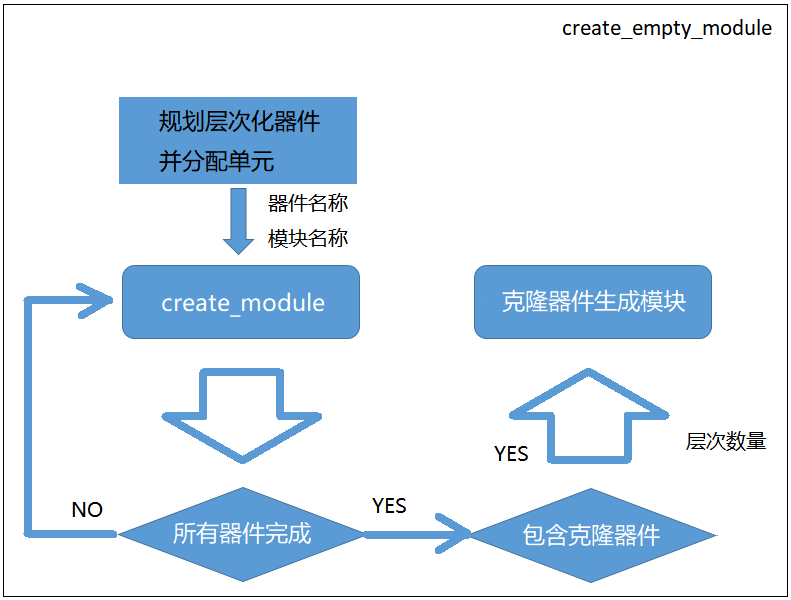


图5.10 create\_empty\_module模块

### 5.4.1 规划克隆层次化器件树状结构

“create\_empty\_module”模块的第一步是规划克隆层次化器件树状结构，同时将结果保存在一个临时文件中。当设计中包含克隆层次化器件时，首先要将门级网表配置文件中的变量“has\_mc\_hinst”设置为1，并提供克隆层次化器件树状结构描述文件，文件的格式示例如图5.11所示。

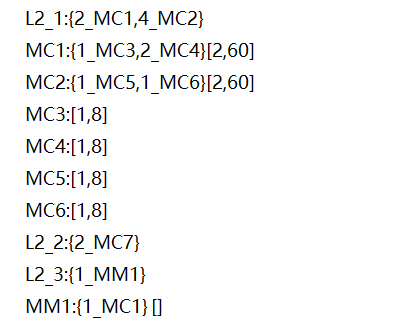


图5.11 克隆层次化器件树状结构描述文件示例

图中一共出现了两种类型的层次化器件，其中MC代表克隆层次化器件，MM代表标准层次化器件，它们都是层次化器件的模块名称。L2\_1代表了该树状结构的位置，其中2代表在第二层，1代表在第一个树下。位置信息后的大括号代表了该树状结构所包含的层次化器件。以2\_MC1为例，它的含义是该树状结构含有两个MC1类型的层次化器件。层次化器件后的大括号代表了该层次化器件包含的器件，中括号内的两个数字分别代表宏单元数量以及标准单元数量。以[2,60]为例，2代表含有两个宏单元，60代表含有六十个标准单元。若没有指定单元数量，模块会为器件分配一个默认值（不包含宏单元，含有十个标准单元）。

这种配置方式有很多优势，第一，可以自由的在标准层次化器件结构中配置所需要的克隆器件树状结构。第二，可以在克隆层次化器件的结构中加入标准层次化器件，满足了设计的多样性。第三，满足同种克隆器件位于不同层这种极端情况。

### 5.4.2 规划标准层次化器件树状结构

“create\_empty\_module”模块的第二步是规划标准层次化器件树状结构。结构生成模块提供了自定义和默认两种方式，用户可以在门级网表配置文件中进行选择。结构生成模块在这一步会将层次化器件的名称和层次化器件对应的模块名称规划出来并将其保存在一个临时文件中。当用户选择默认的方式时，结构生成模块会根据用户设定的子器件数量为层次化器件生成子器件，直到整个树状结构的层数达到用户的设定值为止。当用户选择自定义的方式时，需要为结构生成模块提供标准层次化器件树状结构描述文件，文件格式的示例如图5.12所示。

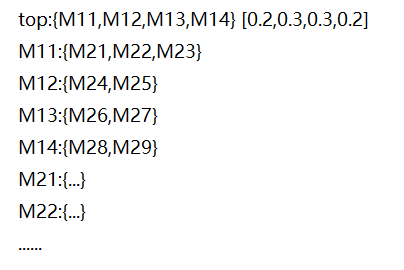


图5.12 结构生成模块树状结构描述文件格式示例

图中M代表这个层次化器件的模块名称，top代表在顶层的层次化器件。中括号的内容表示顶层器件所包含的标准单元数量的比例。如若用户没有设定或者只设定其中的一部分，结构生成模块会自动将剩余的标准单元平均分配在顶层中未标注比例的器件上。结构生成模块对器件的命名规则有一定的约束，字母后第一个数字代表器件所在的层数，第二个数字代表器件的序号，便于进行区分。

结构生成模块使用了自研的层次化器件查找算法，这个算法通过自上而下的顺序规划层次化器件，这样做的优势是提高了算法的运算速度，减少了层次化器件名称的查找次数。层次化器件查找算法使用了队列的概念，算法首先会查找顶层的层次化器件并设定好器件名称，将它们的模块名称保存在模块名队列中，将器件名称保存在器件名队列中，并在文件中录入器件信息。之后查找下一层的器件，算法会吐出模块名队列中存储的第一个器件，并根据其模块名称查找该器件所包含的子器件，将找到的器件的模块名称加入模块名队列中，再根据父器件的名称生成子器件名称，将生成的器件名称加入器件名队列并在文件中录入。以此迭代，直到队列中所有的器件都规划完毕。

自定义的方式虽然需要提供树状结构描述文件，但能够满足树状结构的复杂需求。默认的方式可以方便用户处理结构简单的设计。用户可以根据需要自由选择。

### 5.4.3 分配单元与层次化器件的生成

“create\_empty\_module”模块的第三步是分配单元。结构生成模块会优先为克隆层次化器件分配宏单元和标准单元。宏单元和标准单元单元分配的数量在克隆层次化器件树状结构描述文件中指定。之后为标准层次化器件分配单元。

为了使生成的门级网表高度模拟真实设计的单元分布结构，论文对多家芯片公司所设计的芯片结构进行了调研（其中包括CPU、GPU等多种类型，也包括了5nm和7nm等先进工艺，出于保密的目的，暂不列举芯片名称）。我们发现在真实的设计中，宏单元大多位于树状结构的底层和顶层上，并且宏单元在树状结构的底层分布均匀。因此在分配宏单元时，会优先给底层的器件分配宏单元，每个器件会获得两个同类型的宏单元，如果底层的器件不足以分配所有的宏单元，结构生成模块会将分配完之后剩余的宏单元分配在顶层。在分配标准单元时，会模拟真实设计中器件的规模随着所在层次数的增加由大变小的结构。结构生成模块将通过自上而下、由多至少的方式为每一层分配标准单元，具体公式如(5-1)所示。

 (5-1)

式中，inst\_num代表该树状结构layer层的器件所包含的标准单元数量。remain\_inst\_num代表该树状结构中未分配的标准单元数量。hinst\_num代表该树状结构layer层的器件数量。生成的结果保存在器件名称的临时文件中。

最后，结构生成模块“create\_empty\_module”会根据临时文件中的内容，利用SAI工具的命令“create\_module”生成层次化器件。如果设计中包含克隆层次化器件，由于克隆层次化器件的特殊性，无法直接和标准层次化器件一起生成，需要调用克隆层次化器件树状结构生成模块“create\_empty\_mc\_module”。

## 5.5克隆层次化器件树状结构生成模块

克隆层次化器件树状结构生成模块（后面统称为克隆结构生成模块）的主要功能是生成用户指定的克隆层次化器件树状结构。

### 5.5.1 克隆层次化器件简介

所谓克隆层次化器件是指多个（至少两个）层次化器件使用同一个模块名称，但器件名称不同。这种器件具有相同的内部结构以及相同的内部连接关系，并且包含了相同的单元类型。

由于克隆层次化器件情况特殊，其结构多变且复杂，因此市面上的布图规划工具都需要用户手动规划克隆层次化器件的生成顺序，然后利用工具的命令手动生成。这种方式大大提高了时间成本和复杂度，因此论文为克隆结构生成模块设计了一种排序生成算法，它可以自动规划克隆层次化器件的生成顺序并生成。这种算法不但自动化程度高，并且支持各种复杂结构。为了简化工具的使用难度，论文将克隆结构生成模块集成在层次化器件树状结构生成模块中，工具自动判断是否调用。

### 5.5.2 克隆层次化器件的排序生成算法

为了克服SAI工具的局限性，实现自动化生成克隆器件的各种复杂结构，克隆结构生成模块“create\_empty\_mc\_module”使用了论文设计的排序生成算法并根据临时文件中的内容生成克隆层次化器件。为了帮助读者理解这个算法，这里提供了一个示例进行讲解。

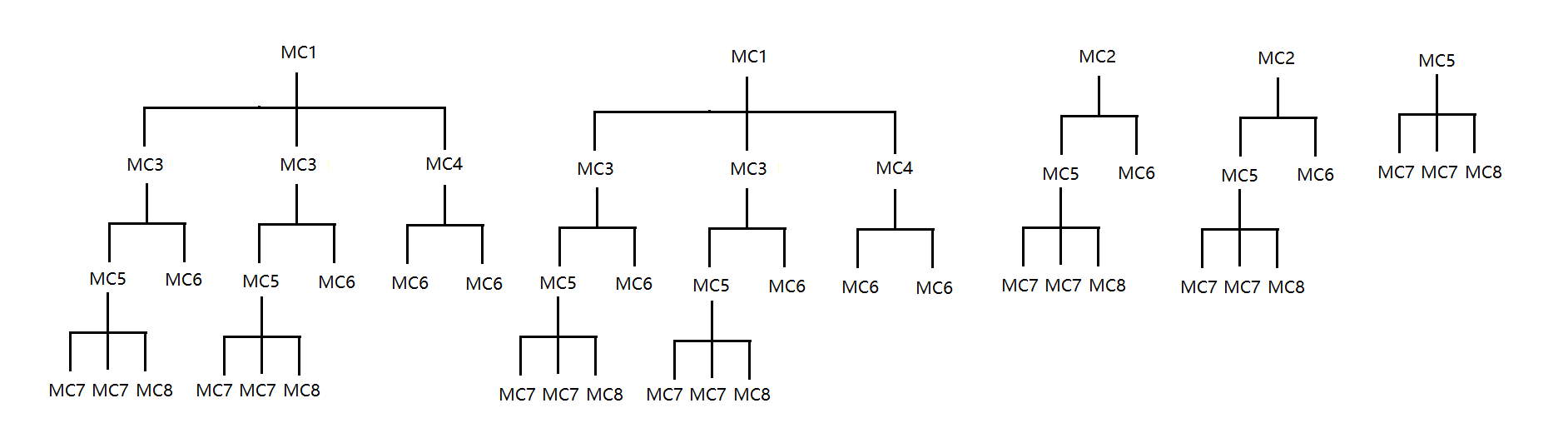


图5.13 克隆层次化器件树状结构示例图（1）

如图5.13所示是一个复杂的克隆层次化器件树状结构图。这个图中一共包含了五个独立的树状结构，克隆层次化器件共分布在三层。图中一共包含了八种器件，其中“MC5”分别出现在第一层和第二层，并且包含两个“MC7”。这个树状结构图的复杂之处在于：第一，不止一个树状结构；第二，部分克隆层次化器件分布在不同的层；第三，部分克隆层次化器件不在同一树状结构下。面对这样的结构，首先算法会提取五个独立的树状结构的头节点并保存在队列中，方便对每一个树状结构独立处理。针对每一个独立的树状结构，首先提取出这个结构中所有不重复的模块名称，根据这个名称选取第一个对应的层次化器件。以第一个树状结构MC1为例，如图5.14所示，克隆结构生成模块会先选取并生成被椭圆形圈出的层次化器件，生成的顺序为MC1、MC3、MC4、MC5、MC6、MC7、MC8。

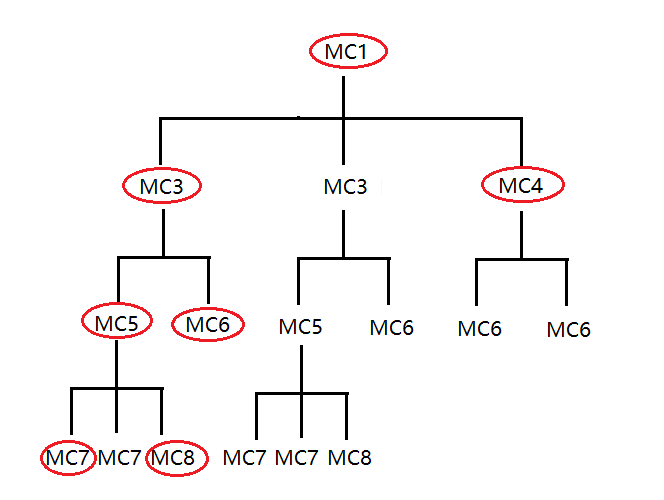


图5.14 克隆层次化器件树状结构示例图（2）

抓出对应的器件后，克隆结构生成模块会根据层次的顺序由上而下依次生成它们。在此之前，克隆结构生成模块会将要生成的器件的模块名在已生成模块文件中进行匹配，如果文件中不包含这个模块名，当这个模块名对应的层次化器件被完整的构建之后，会将这个模块名保存在已生成模块文件中。如果文件中包含这个模块名，则在生成这个模块名对应的层次化器件之后，其子器件均不必再次生成。这是因为克隆层次化器件具有相同的内部结构，这个模块的内部结构被构建后，其他的克隆器件只需要调用这个模块就能获得相同的内部结构。当一个不同种类的模块内部结构被构建完成之后，将其模块名保存在已生成模块文件中，后续生成的同种类克隆层次化器件只需使用这个模块名即可。

之后，克隆结构生成模块会生成这些器件所包含的子器件，生成的顺序会根据这些子器件的所在层由下至上的完成。以MC1为例，如图5.15所示，这一步被生成的克隆层次化器件被长方形圈出，生成的顺序为MC7、MC6、MC6。

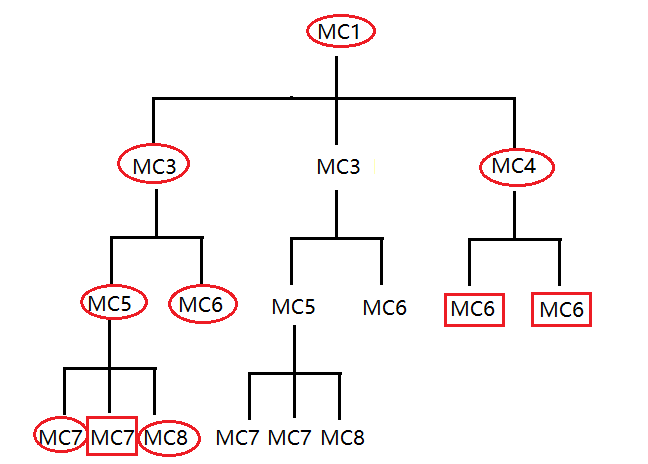


图5.15 克隆层次化器件树状结构示例图（3）

最后一步补足这个树状结构。补足的方式也是根据剩余器件的层次顺序由上而下依次生成它们。还是以第一个树状结构MC1为例，克隆结构生成模块会优先生成MC3，并且只需要生成MC3，不需要生成MC3所包含的层次化器件。具体原因正如之前介绍的那样，MC3在之前的步骤中其内部结构已经被构建完毕，不需要重复操作。至此，一个树状结构就被处理完毕了，接下来只要把其余的树状结构利用排序生成算法处理一下，就能够实现自动并且快捷的完成克隆层次化器件树状结构的搭建。

## 5.6 单元连接

单元连接的主要功能是将标准单元和宏单元进行连接。为了加入组合逻辑与连接关系，本节需要解决以下几个问题：触发器在标准单元中的比例是多少？如何搭建组合逻辑电路？如何连接宏单元？

### 5.6.1 标准单元比例分配问题

论文对多个芯片公司的多种类型的芯片设计进行了调研，这些设计都使用了先进的半导体工艺，如表5.2所示为调研的部分结果（出于保密的目的，这些设计的名字被design所代替）。

表5.2 设计调研表



在调研表中我们可以看出，触发器在标准单元中所占的比例的平均值为10%，因此对于第一个问题，标准单元连接模块在连接标准单元时，会将触发器的比例控制在10%左右。为了满足设计的多样性，标准单元连接模块支持用户在配置文件中指定触发器的比例。

5.6.2 标准单元连接模块

如图5.16所示为论文开发的标准单元连接模块“connect\_module\_inst”。首先它会调用器件内部标准单元连接模块“connect\_inst”为层次化器件连接内部的标准单元，调用时需要提供模块名称，五种标准单元的名称以及触发器比例共七个参数。模块“connect\_inst”会读取分配的标准单元数量，根据提供的触发器比例计算出触发器的数量和组合逻辑单元的数量，最后根据设定的六种基本电路进行连接。所有器件的标准单元连接完毕后“connect\_module\_inst”模块结束。使用“connect\_module\_inst”模块时需提供五种标准单元的名称和触发器比例六个参数。

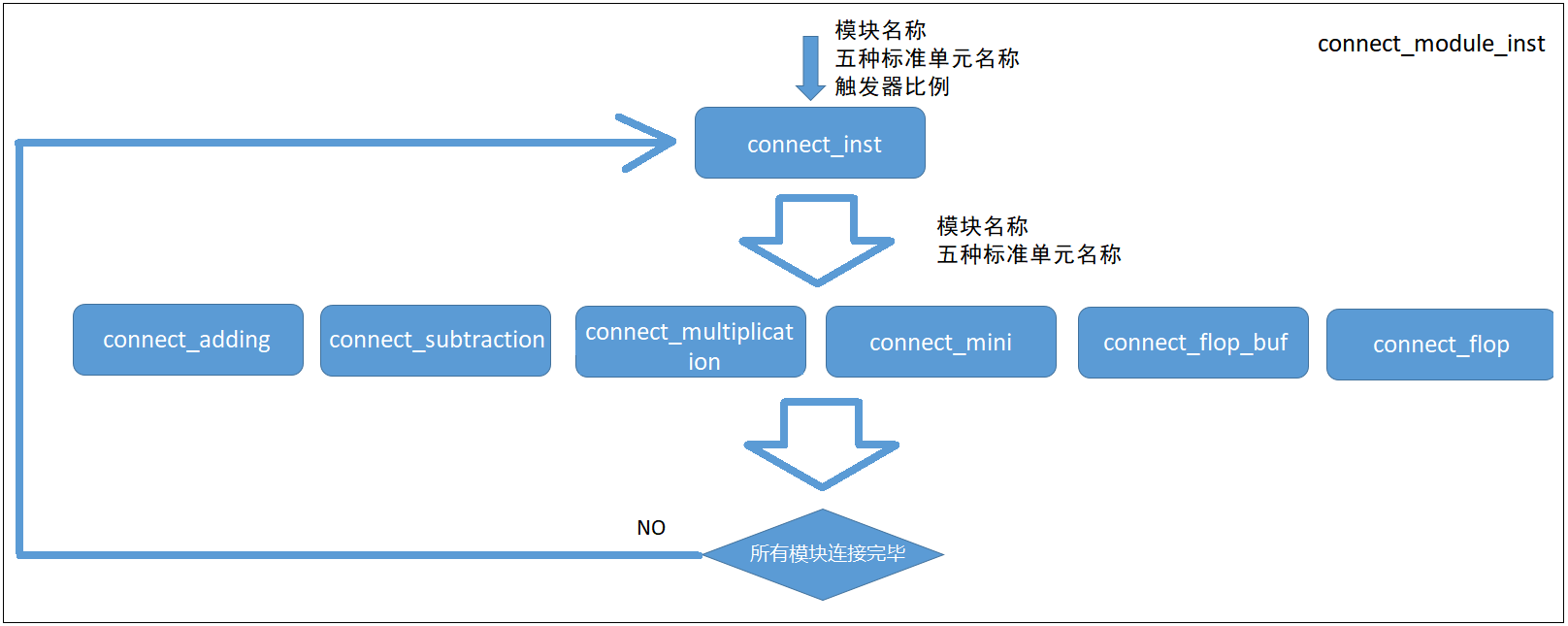


图5.16 connect\_module\_inst模块

对于第二个问题，标准单元连接模块会将逻辑门和触发器组合成六输入三输出的四种计算电路，每三个计算电路为一组，每组之间用触发器与缓冲器电路相连，一组为一个pipeline。一条路径上pipeline的最大值默认是16（可修改）。在连接时算法优先为第一条路径安插计算单元，如果这条路径不够安插所有的计算单元，算法会自动添加输入输出端口开辟第二条路径，以此类推，直到所有的标准单元都被连接。实现过程是模块“connect\_inst”内部调用加法电路连接模块（connect\_adding）、减法电路连接模块（connect\_subtraction）、乘法电路连接模块（connect\_multiplication）、迷你电路连接模块（connect\_mini）、触发器缓冲器连接模块（connect\_flop\_buf）和触发器连接模块（connect\_flop）。迷你电路是指当剩余的标准单元不足以构成一个完整的计算电路时，为保证整个设计的单元数量而构建的小型电路。这六个模块所连接的电路的电路图如图5.17、图5.18、图5.19、图5.20、图5.21和图5.22所示。

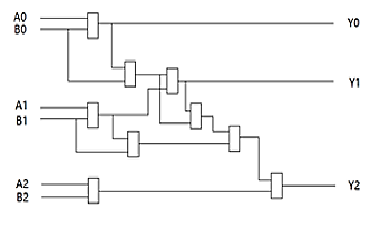
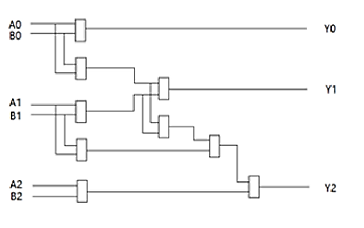


图5.17 加法电路 图5.18 减法电路

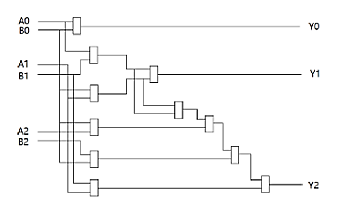
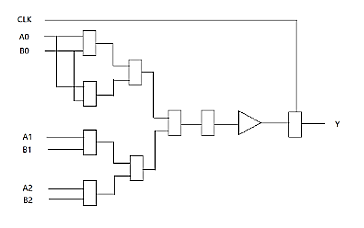
 

图5.19 乘法电路 图5.20 迷你电路

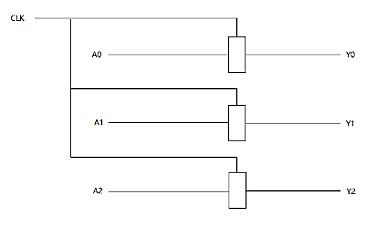
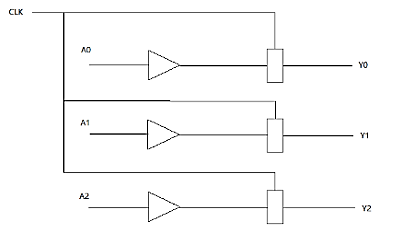


图5.21 触发器与缓冲器电路 图5.22 触发器电路

### 5.6.3宏单元连接模块

如图5.23所示为论文开发的提供给用户使用的宏单元连接模块“connect\_module\_macro”。在使用时需要提供宏单元、触发器和缓冲器名称以及顶层宏单元的数量六个参数。首先它会调用器件内部宏单元连接模块“connect\_macro”为包含宏单元的层次化器件连接内部的宏单元，调用时需提供层次化器件名称、触发器名称、缓冲器名称和器件的宏单元名称共四个参数。所有器件连接完后调用顶层宏单元连接模块“connect\_top\_macro”，调用时需提供顶层宏单元数量、触发器名称、缓冲器名称和三种宏单元名称共六个参数。

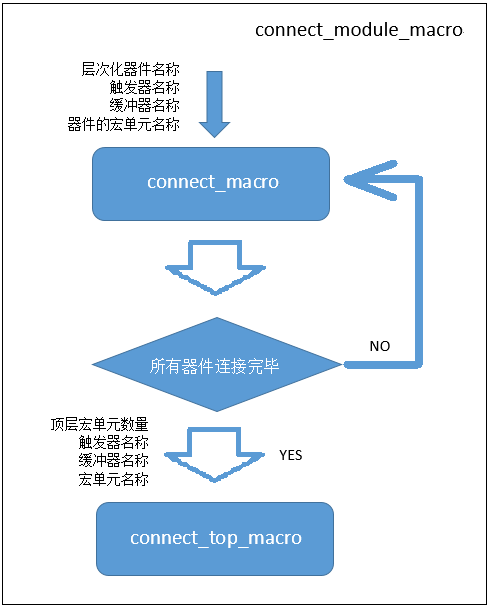


图5.23 connect\_module\_macro模块

对于第三个问题，宏单元连接模块会将同一层次化器件中的宏单元连接在一条路径上，宏单元之间会添加触发器。宏单元的类型共有三种，同一模块名的层次化器件被分配了相同类型的宏单元以保证克隆层次化器件具有相同的内部结构。

## 5.7 层次化器件连接

层次化器件连接包括了顶层层次化器件与顶层端口的连接以及子层次化器件与父器件的连接两部分。若器件中包含了克隆层次化器件，则还需要将克隆层次化器件与顶层端口进行连接。

### 5.7.1 连接层次化器件的模块构成

如图5.24所示为层次化器件连接的流程图。图中的两个模块为论文开发的提供给用户使用的标准层次化器件的连接模块“connect\_nested\_module”和标准层次化器件与顶层端口的连接模块“connect\_top\_module”。“connect\_nested\_module”模块会将所有非顶层的标准层次化器件的输入输出端口连接到其上一层的层次化器件上。之后使用模块“connect\_top\_module”将顶层的标准层次化器件的输入输出端口和芯片的输入输出管脚相连。若设计中包含克隆器件，“connect\_top\_module”模块会调用克隆层次化器件与顶层端口的连接模块“connect\_top\_mc\_module”。调用时需提供设计的层次数量作为参数。使用模块“connect\_nested\_module”时不需要提供参数，使用模块“connect\_top\_module”时需要提供设计的层次数量作为参数。

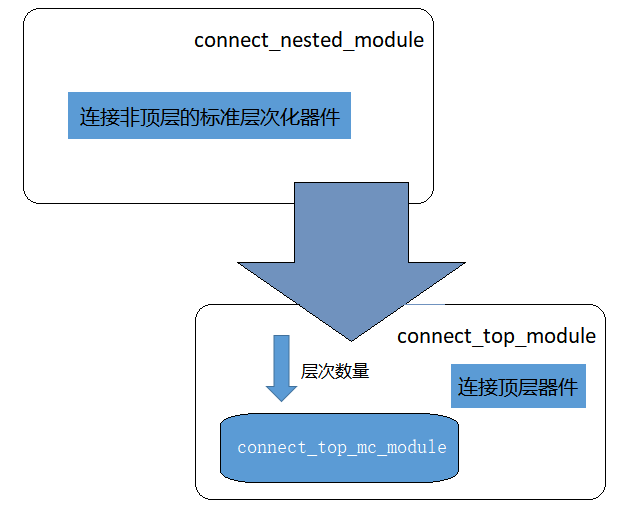


图5.24 层次化器件连接的流程图

### 5.7.2 层次化器件的连接关系

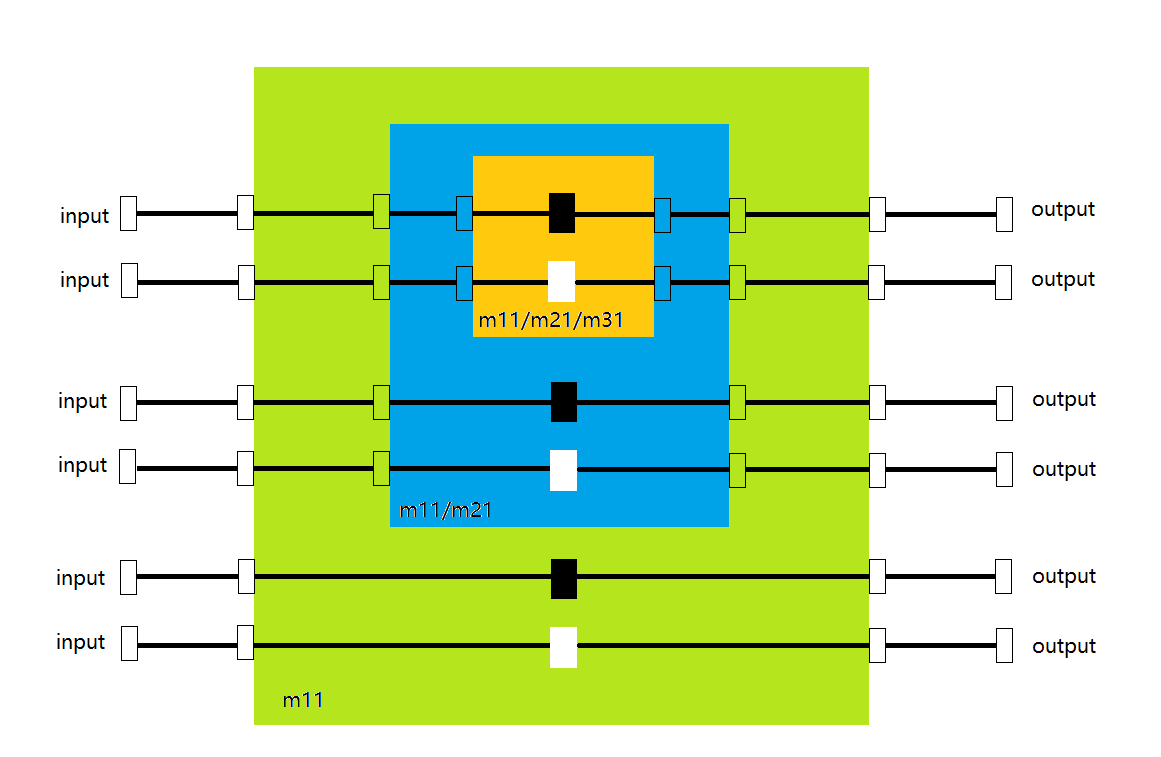


图5.25 层次化器件输入输出端口连接关系模型图

为了帮助读者理解上述三个模块的实现原理，这里展示了一个层次化器件输入输出端口的连接关系模型图，如图5.25所示。图中黑色的方块代表宏单元，白色的方块代表组合逻辑，它们和层次化器件端口的连线代表的是x位的总线。图中绿色的方块代表顶层层次化器件m11，蓝色的方块代表层次化器件m11/m21，黄色的方块代表层次化器件m11/m21/m31，透明的方块代表输入输出端口。如果这是一个标准层次化器件结构，“connect\_nested\_module”会将层次化器件m11/m21/m31的输入输出端口连接在m11/m21的端口上，并将m11/m21的输入输出端口连接在m11上，之后“connect\_top\_module”会将m11的所有输入输出端口连接在顶层的输入输出管脚上。如果这是一个克隆层次化器件的结构，“connect\_top\_mc\_module”会先在m11和m11/m21两个器件上创建端口，通过它们将器件m11/m21/m31的输入输出端口与顶层端口相连，m11/m21和m11也是如此。“connect\_top\_mc\_module”模块被集成在模块“connect\_top\_module”中，不建议用户手动使用。

## 5.8 早期门级网表生成器与SAI工具的结果对比

为了展示论文优化后的早期门级网表生成器相比于SAI工具的优势与特点，本节分别使用两个工具基于40nm库文件对同一结构的门级网表进行复刻，最后将产生的门级网表进行对比并分析。

### 5.8.1 早期门级网表生成器的结果呈现

首先这里选择了一个40nm的库文件和一个简易的mmmc文件填在了门级网表配置文件中。之后使用单元查找器在库文件中查找所需的单元名称，单元查找器将单元查找的结果分别保存在and.rpt、or.rpt、xor.rpt、buffer.rpt及dff.rpt几个文件中。论文在这五个文件中分别选择了“AND2X1”、“OR2X1”、“XOR2X1”、“BUFX4”及“DFFQX4”填入了门级网表配置文件中。

接下来需要使用宏单元生成器定制宏单元。首先在宏单元配置文件中将宏单元的输入输出端口数量设为8，pipeline的数量设为5，库文件与门级网表配置文件中保持一致，宏单元配置文件如图5.26所示。配置完成后运行脚本，生成宏单元的lef文件。门级网表配置文件的库文件及单元选配部分如图5.27所示。



图5.26 宏单元配置文件



图5.27 库文件及单元选配

门级网表配置文件的设计基本信息部分如图5.28所示。图中选择了默认的方式生成标准层次化器件，设计一共有五层，其中共有五十个宏单元，九十万个标准单元。其中每个标准层次化器件包含两个子标准层次化器件，顶层共有六个层次化器件，其包含的器件比例如图。设计中包含了克隆层次化器件，触发器的比例设为百分之十。

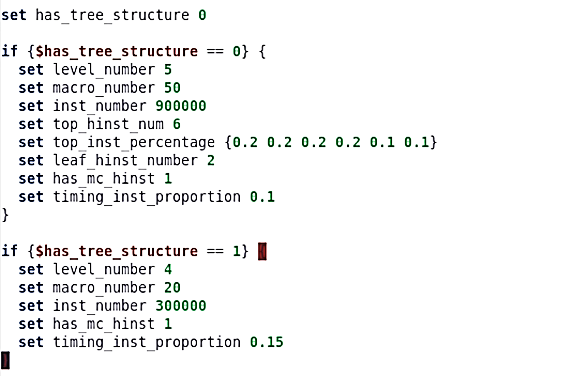


图5.28 设计基本信息

设计中的克隆层次化器件结构复杂，结构中包含了标准层次化器件以及不在同一层和同一树状结构的克隆层次化器件，克隆层次化器件结构描述文件如图5.29所示。

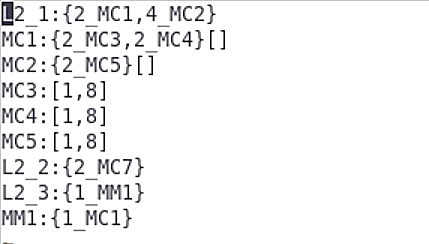


图5.29 克隆层次化器件结构描述文件

对用户来说，门级网表配置文件修改完成之后，只需要运行早期门级网表生成器的脚本predict\_floorplan.tcl，工具就会自动生成用户所需的门级网表，脚本如图5.30所示。但为了方便理解并展示，这里将脚本拆分执行。

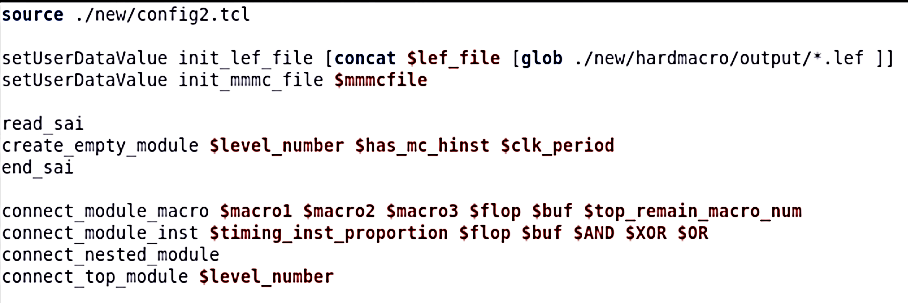


图5.30 predict\_floorplan.tcl

脚本的第一步是读入门级网表配置文件和mmmc文件，之后正式进入设计部分。脚本使用命令“read\_sai”进入SAI模式，“create\_empty\_module”模块规划层次化器件并分配单元，并将器件名称、单元数量及模块名称进行保存，保存的结果如图5.31和5.32所示。

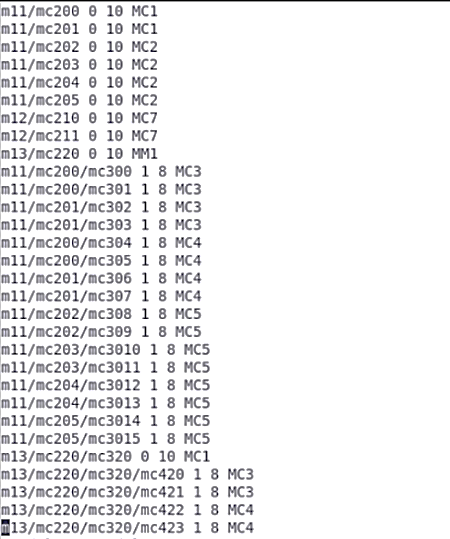
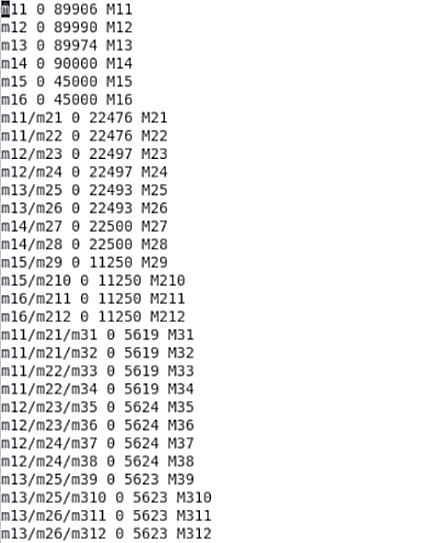
 

图5.31 克隆层次化器件部分结果 图5.32 标准层次化器件部分结果

“create\_empty\_module”模块会根据规划结果生成层次化器件。本次一共需要生成186个标准层次化器件，30个克隆层次化器件，生成器件的部分层次结构图如图5.33和5.34所示。

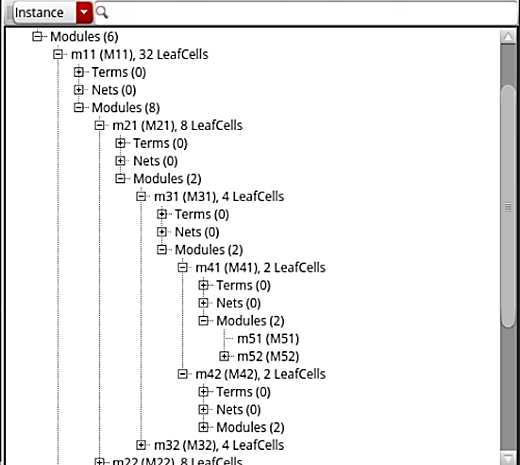
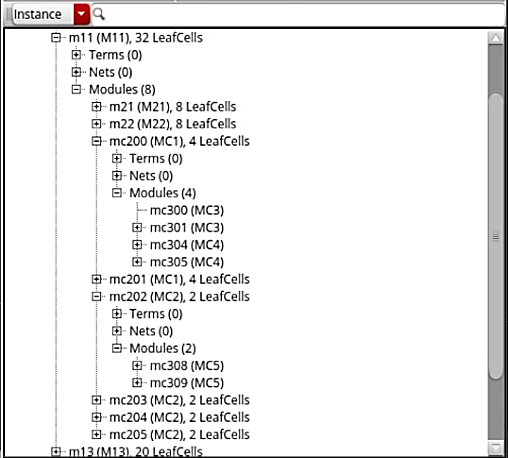
 

图5.33 器件的层次结构（1） 图5.34 器件的层次结构（2）

“create\_empty\_module”模块执行结束后，脚本会运行“end\_sai”命令退出SAI模式进入单元连接阶段。

接下来“connect\_module\_macro”模块会读取宏单元的分配结果在门级网表中添加宏单元并连接。如图5.35所示为模块“connect\_module\_macro”的执行结果，可以看到五十个宏单元和连接它们的触发器已经被成功添加在设计当中。

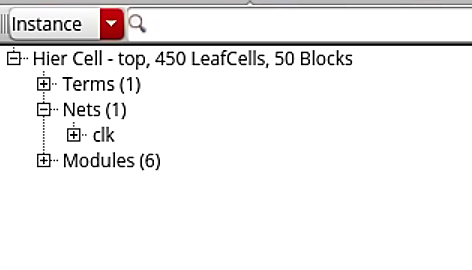


图5.35 connect\_module\_macro的结果

“connect\_module\_inst”模块会读取标准单元的分配结果在门级网表中添加标准单元并连接。如图5.36所示为模块“connect\_module\_inst”的执行结果，图中可以看到一共添加了899890个标准单元，这个数字和目标数相差110。由于添加组合逻辑并控制触发器的比例必然会引入误差，因此模块通过引入精密的计算和多元化的标准单元组合方式将误差控制在0.01%左右。图中可以看到几个顶层的层次化器件包含的标准单元数量，层次化器件m11、m12、m13和m14所包含的标准单元数量的比例在20%左右，m15和m16所包含的标准单元数量的比例为10%左右，误差均不超过0.01%。在图5.37中可以看到本次设计中包含的触发器数量为90344，触发器在标准单元中所占的比例约为10%，误差不超过0.01%。这样的结果十分理想。

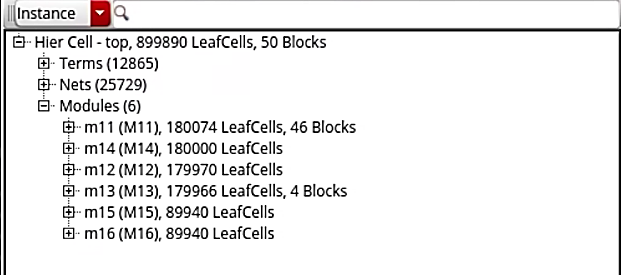


图5.36 connect\_module\_inst的结果



图5.37 触发器的数量

“connect\_nested\_module”模块会将设计中的层次化器件与子器件进行连接，“connect\_top\_module”模块会将顶层的层次化器件与输入输出端口进行连接。如图5.38和5.39所示为顶层添加的端口及顶层添加的连线的部分结果。

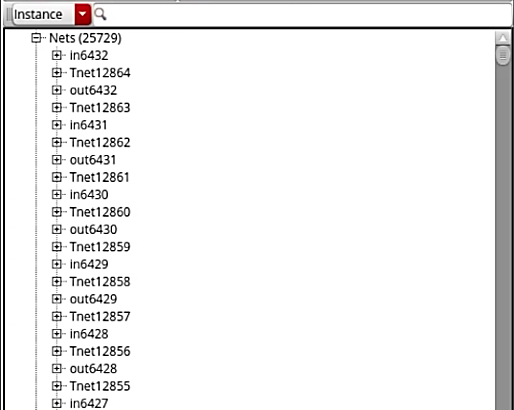
 

图5.38 顶层添加的端口的部分结果 图5.39 顶层添加的连线的部分结果

脚本运行完毕后使用命令“saveNetlist”可以将门级网表保存下来。最后，本节使用了论文研究的混合尺寸布图方法进行布图，使用的命令是“place\_design -concurrent\_macros”，产生的布图结果如图5.40和图5.41所示。第一张图片是布图结果的缩略图；第二张图片是布图结果的细节图，图中红黄绿色的部分为单元之间的连线。图中我们可以清晰地看到宏单元和标准单元的布图情况。

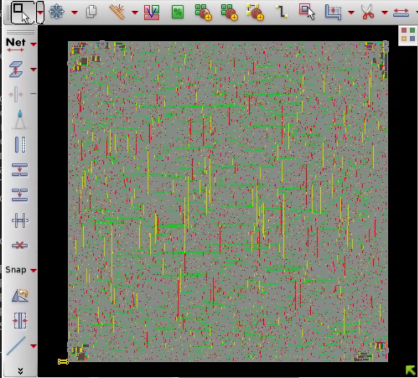
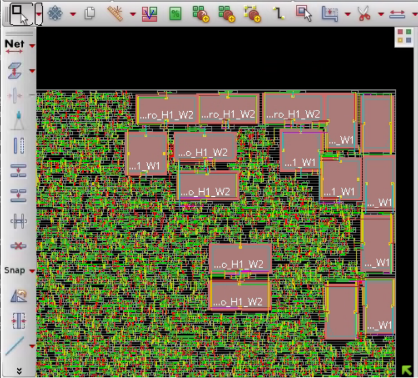
 

图5.40 自动布图结果缩略图 图5.41 自动布图结果细节图

### 5.8.2 SAI工具的结果呈现

为了使结果对比更具说服力，这里用SAI工具生成和早期门级网表生成器相同结构的门级网表，并且使用了相同的库文件和mmmc文件，SAI工具的使用脚本如图5.42所示。

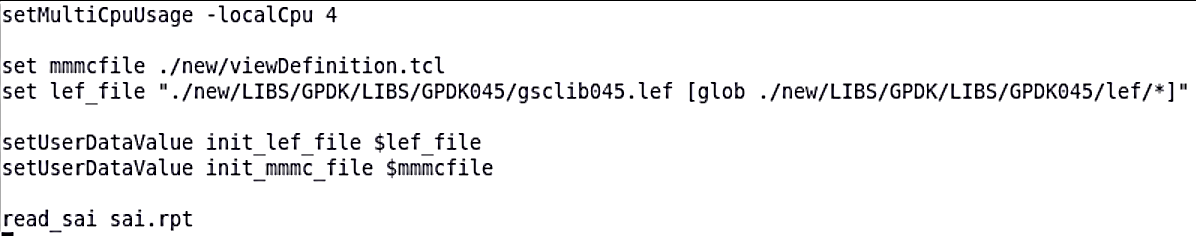


图5.42 SAI工具使用脚本

脚本最复杂的部分在SAI文件“sai.rpt”中，SAI文件的完整内容可在附录1中查看。文件使用了最新的2.0版本，文件中设置了名为DFFQX4和BUFX4的触发器和缓冲器两种参考单元，器件名称需要在库文件中查找。文件的最后一部分是层次化器件的基本信息，在这个设计中一共包含216个层次化器件，其中有186个标准层次化器件和30个克隆层次化器件。首先需要设置器件名称和模块名称，之后分配标准单元。为了控制顶层层次化器件的标准单元数量比例和触发器比例，需要计算出每一个层次化器件包含的标准单元数量和触发器数量，在计算和分配过程中还要考虑克隆器件的特殊性，这些都需要人工完成。由于门级网表中单元与器件数量多，树状结构复杂，因此手工计算十分复杂。

填写SAI文件时，需要使用命令“create\_module”为每一个层次化器件都编辑一次，同时还要计算子器件数量，非常复杂。生成的结果如图5.43和5.44所示。从图中可以看到：

1、设计中不包含宏单元，这是因为库文件中没有宏单元的信息。

2、手工计算的标准单元数量误差很大。

3、设计中没有任何连接关系。

4、SAI工具会在层次化器件下建立一个新的器件以保存标准单元，无形中破坏了用户设定的层次化器件树状结构。

5、由于设计中只有两种单元，因此没有复杂的组合逻辑。

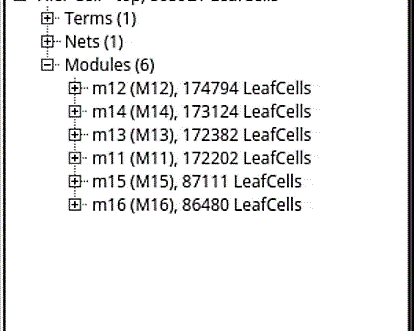
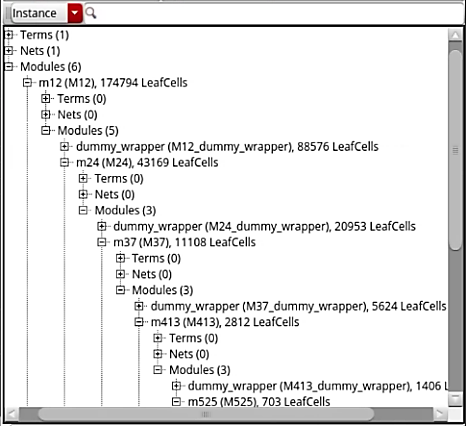
 

图5.43 SAI工具设计结果（1） 图5.44 SAI工具设计结果（2）

SAI工具生成的门级网表的布图结果以及细节如图5.45和图5.46所示（由于它们没有连接关系，因此打开连接显示也看不到任何的连线）。

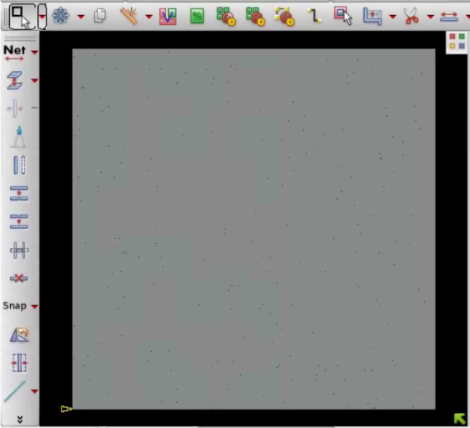
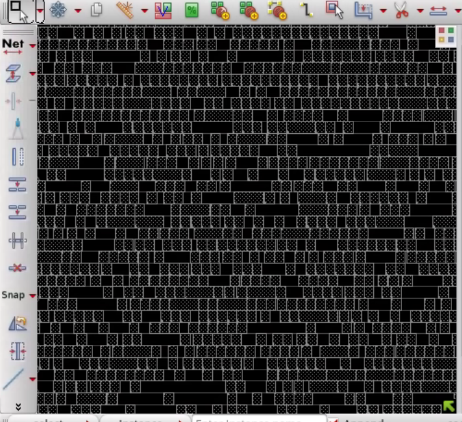
 

图5.45 SAI布图结果 图5.46 SAI布图结果细节图

### 5.8.3对比与总结

经过上面的设计后，这里对SAI工具和早期门级网表生成器的功能以及使用方式进行了对比，总结的表格如表5.3所示：

表5.3 功能对比表



首先，两个工具都支持定制早期门级网表，并且它们都能支持不同的工艺和不同的类型。但SAI工具要求用户熟悉相关命令及使用规则，还需要花费大量的时间编写和修改SAI文件，并且产生的结果误差大。经过优化后，用户只需要修改门级网表配置文件即可控制门级网表的结构，结果精确。填写参数时还可以利用辅助工具，更加简单方便。同时成功避免了SAI工具破坏预定树状结构的问题。

在控制层次化器件树状结构的问题上，SAI工具需要手工规划并使用命令分别生成每一个层次化器件，复杂且耗时。经过优化后，用户只需要填写门级网表配置文件并提供结构描述文件即可。实现了自动生成层次化器件树状结构的功能。

在控制器件单元数量比例和触发器比例的问题上，SAI工具需要人工计算每一个层次化器件所包含的单元数量，复杂且误差大。经过优化后，早期门级网表生成器会自动帮助控制，简单精确。

除了工具使用的便利度得到提高外，早期门级网表生成器也具有一些独特的创新点，一共包括九个部分。第一，在标准单元和宏单元的分布上模拟了真实设计的分布结构。第二，提供给用户两种层次化器件定制方案，这两种方式都可以实现自动生成，提高了工具的自动化程度。第三，使用了自研的排序生成算法帮助用户自动生成克隆层次化器件。第四，在门级网表中为标准单元、宏单元以及层次化器件添加了连接关系。第五，将标准单元连接成了计算器，实现了在门级网表中添加组合逻辑。第六，工具提供了UPF文件自动生成的功能。第七，可以帮助用户定制宏单元，摆脱了库文件的束缚。第八，工具提供了从库文件中查找标准单元名称的功能。第九，将所有顶层层次化器件连接到顶层端口上，模拟了真实的设计。

除此以外，在生成门级网表的耗时方面，早期门级网表生成器从修改参数到门级网表的生成一共花费了25分钟，SAI工具从编辑文件到门级网表的生成一共花费了74分钟，效率提升了66.2%。

## 5.9 本章小结

本章主要介绍了早期门级网表生成器的实现原理和实现过程中使用到的部分算法，并与SAI工具的结果进行了对比。根据SAI工具的不足，早期门级网表生成器做出了如下优化：早期门级网表生成器在数据准备阶段为用户提供了文件选配和设计参数控制的开关，并且提供单元查找器、宏单元生成器以及UPF自动生成器三个辅助工具，提高了用户对设计的控制能力，将结果的误差控制在0.01%以内，同时极大的提高了便利性。早期门级网表生成器在树状结构生成阶段提供了两种生成方式，满足了用户不同的需求。同时通过自研的算法，解决了一直以来层次化器件无法自动生成的问题，提高了工具的自动化程度。在分配单元时高度模拟了真实设计的结构，提高了生成的门级网表的真实度。早期门级网表生成器在单元连接阶段给生成的门级网表加入了多种连接关系和组合逻辑，提供了控制时序单元比例的功能，使生成的门级网表更加贴近芯片项目综合后的门级网表。这些功能都是SAI工具不完善或不具备的。并且经过验证，早期门级网表生成器的使用效率提升了66.2%。

# 第六章 总结与展望

## 6.1 总结

随着集成电路的不断发展，后端物理设计工作和EDA验证工作的难度不断增加，后端物理设计工程师为了提高PPA，不得不在早期阶段进行布图规划；EDA验证工程师在验证过程中需要很多复杂的门级网表进行验证工作。针对上述需求，论文设计了一款自动布图规划工具，它由一个早期门级网表生成器和一个混合尺寸布图方法组成。早期门级网表生成器能够自动的帮助用户生成一个具有连接关系和组合逻辑的高级门级网表，高度自动化的工具提高了工作效率和工作质量。混合尺寸布图方法能够实现自动且同时的摆放宏单元和标准单元，节约了时间成本的同时还提高了布图质量。论文的主要工作和成果如下：

（1）介绍了SAI的使用模式、流程以及命令。同时分析研究了低功耗的基本原理，详细分析了功耗的组成及来源，介绍了几种常见的低功耗技术。

（2）研究了混合尺寸布图方法中的算法及流程，包括目标函数以及为解决布图结果中的线长问题和拥塞问题而引入函数的派系模型和静电密度模型，最后介绍了混合尺寸布图方法的流程。最后，分析对比了论文研究的混合尺寸布图方法和传统布图方法的布图结果，混合尺寸布图方法产生的结果WNS平均减少了33.35%，TNS平均减少了30.78%，max transition 的DRV数量平均减少了55.65%，单元密度平均减少了0.63%，泄漏功耗平均减少了8.29%，线长平均减少了8.06%，最大热点值平均减少了28.92%，总热点值平均减少了34.51%，TAT平均减少了25.7%，证明了混合尺寸布图方法在布图质量和耗时方面更具优势。

（3）研究并改进了电源线模型。电源线优化模型具有计算结果更准确、保存的有效信息更多、可以补偿或标注电源线密度差异等特点。在电源线分布不均的极端情况下，使用了电源线优化模型的布图算法的WNS可以额外减少33.34%，TNS可以额外减少19.45%，max transition 的DRV数量可以额外减少14.28%，单元密度可以额外减少0.24%，线长可以额外减少10.93%，最大热点值可以额外减少26.68%，总热点值可以额外减少47.58%。

（4）设计并实现了早期门级网表生成器。早期门级网表生成器是基于SAI工具，融合了DB ECO和一些算法而来，实现了添加连接关系、添加组合逻辑、自动生成层次化器件、模拟真实的设计分配单元以及约束时序单元比例等功能。除此以外，它还简化了SAI工具的使用难度，提高了自动化程度。

（5）分析对比了早期门级网表生成器和SAI工具，使用了早期门级网表生成器和SAI工具复刻同一种结构的门级网表，最后对比了生成的门级网表，证明了早期门级网表生成器具有生成的门级网表质量更高、自动化程度更高、使用更方便、误差更小等优点。

本论文来源于笔者在上海楷登电子科技有限公司实习期间的项目，论文实验数据均来源于上海楷登电子科技有限公司。

## 6.2 展望

由于时间等多方面因素，论文尚有一些不足之处。

（1）通过选取大量的设计不断的进行实验，我们发现在少部分实验中混合尺寸布图方法处理宏单元之间的间距结果不太理想，这种情况往往需要人工干预。并且混合尺寸布图方法更适合宏单元数量多的设计使用，实验发现，当宏单元的数量大于100时，混合尺寸布图方法的优化效果尤为明显，当宏单元的数量小于30时，混合尺寸布图方法相比于传统布图方法在布图质量上并无明显优势。

（2）早期门级网表生成器虽然在门级网表中加入了连接关系和组合逻辑，但连接关系和组合逻辑不能根据设计的不同而变化。本论文研究的早期门级网表生成器虽然满足了EDA验证和早期布图规划的基本需求，但如果想在早期布图规划方向大规模商用还需要在上述方向进行改进。

目前，将人工智能与后端相结合已经成为了EDA工具发展的趋势。也许在人工智能的帮助下，混合尺寸布图方法能够实现更高质量的布图，人工智能也能够根据不同种类芯片的设计规律，帮助早期门级网表生成器在不同的芯片中加入不同的组合逻辑，引入不同的连接关系。随着集成电路的快速发展，越来越多的新技术也会被应用在芯片设计当中，这也会造成芯片设计的难度不断增加。因此在芯片项目早期阶段的规划会越来越受重视，相信在不久的将来也会涌现出越来越多优秀的早期布图规划方案。