



ELSEVIER

Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng



Research
Engineering Management—Article

芯片电路设计布局优化的精确算法

张滨琪^{a,*}, 镇璐^{a,*}, Gilbert Laporte^{b,c}

^a School of Management, Shanghai University, Shanghai 200444, China

^b Department of Decision Sciences, HEC Montréal, Montréal, QC H3T 2A7, Canada

^c School of Management, University of Bath, Bath BA2 7AY, UK

ARTICLE INFO

Article history:

Received 29 October 2024

Revised 3 March 2025

Accepted 16 March 2025

Available online 22 March 2025

关键词

装箱

布局优化

芯片设计

线性规划

Benders 分解

摘要

布局优化是芯片设计中的关键阶段,其目的是在有限区域内策略性地放置元器件,以提高空间利用率并减少布线长度。芯片设计企业根据设计规则优化布局,以满足客户需求。虽然混合元器件高度电路在现代芯片设计中广泛应用,但很少有研究同时考虑了元器件不重叠、电源轨道对齐和最小植入面积这些布局约束。因此,本研究通过预处理布局问题中的非线性,建立混合整数线性规划模型,以降低企业合法化布局成本为目标展开研究。此外,本研究基于 Benders 分解设计并实现了一种精确算法,利用对偶理论获得最优割,并迭代求解元器件坐标。本研究通过不同规模的数值实验验证了该算法的性能,并且详细分析芯片区域划分的形状、不同类型元器件的比例、元器件和 bin 的总数对布局的影响,为芯片设计企业提供了一些具有潜在价值的设计见解。

© 2025 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. 引言

在信息化时代,手机、计算机等智能产品的普及高度依赖先进芯片技术。芯片集成将普通设备转化为智能产品,这源于公众对技术创新的持续需求。进入后摩尔定律时代[1]后,芯片设计在有限物理尺寸内同时面临优化性能、功耗与成本的挑战。面向芯片设计的优化不仅对电子技术的进步至关重要,还与国家经济和半导体行业的国际竞争力息息相关。

布局优化在芯片设计自动化中占据核心地位,其质量直接影响后续设计任务。传统的电路设计高度依赖经验和手动操作,通常将元器件高度统一以简化问题[2]。然而,随着自动化的进步,越来越多的布局优化任务通过计算机

算法得以实现,既提升设计效率,又减少人为失误。现代电路设计已转向混合元器件高度(MCH)配置,由于存在布线能力、时序、功耗与可制造性等严格约束[3],布局优化难度显著增加。作为装箱问题(BPP)的一种变体,布局涉及根据设计技术的要求将元器件对齐到合法位置,同时分布在有限的芯片区域内。随着新兴技术的演进,布局约束不断变化,优化方法必须持续适应以符合新的设计规则。例如,元器件尺寸的减小虽然带来集成度的提升,但同时也增加了功耗,所以迫切需要创新方法来提高芯片空间利用率。要在芯片设计中保持竞争力,从业者必须根据技术进步和行业变化调整策略。

基础的布局优化问题可描述为:给定一个高度为 h 、宽度为 w 的芯片布局区域,其中电源导轨水平布线,每两

* Corresponding author.

E-mail address: lzhen@shu.edu.cn (L. Zhen).

条导轨之间的间距为 h_0 。布局优化的目的是在该区域内合法地放置 n 个元器件，同时最小化布线总长度。每个元器件 i （其中 $1 \leq i \leq n$ ）的高度为 h_i 、宽度为 w_i ，其左下角坐标定义为 (x_i, y_i) 。此外，元器件 i 在纵轴上的高度通过 k_i 来描述。元器件的上下边界包含与电源导轨对应的引脚。图1展示了包含三种高度元器件的合法和非法布局示例。

基于上述描述，可以构建一个基础布局模型[4-5]：

$$\text{Minimize } \sum_{1 \leq i, i' \leq n} (x_i - x_{i'})^2 + (y_i - y_{i'})^2 \quad (1-1)$$

约束条件为

$$0 \leq x_i, x_i + w_i \leq w, \quad 1 \leq i \leq n \quad (1-2)$$

$$0 \leq y_i, y_i + h_i \leq h, \quad 1 \leq i \leq n \quad (1-3)$$

$$x_i - x_{i'} \geq w_{i'}, \text{ if } x_i \geq x_{i'} \quad (1-4)$$

$$y_i = k_i h_0, \quad 1 \leq i \leq n, k_i \in \{0, 1, 2, \dots\}. \quad (1-5)$$

目标函数（1-1）最小化布线总长度，布线总长度由任意两个元器件之间水平和垂直坐标的平方差定义。元器件布局的基础合法性约束包括：约束（1-2）和（1-3）共同要求所有元器件放置在指定区域内；约束（1-4）明确规定元器件在水平方向上不得重叠；约束（1-5）规定每个元器件的纵坐标必须与相应的电源导轨对齐。由于决策变量 x_i 和 y_i 的定义域在约束（1-2）和（1-3）中隐含定义，此处不再赘述。

在这一基础问题的框架上，本研究进一步探讨现代技术对布局优化的影响。这些技术带来了新的挑战和机遇，将在第3节中详细讨论。

本研究提出了一种混合整数线性规划（MILP）模型和Benders分解（BD）算法，以应对芯片设计中的实际布局挑战。通过平衡布线长度和芯片区域的利用率，所提出的方法系统地集成了芯片区域的划分、元器件的坐标、元器件引脚与芯片导轨的对齐以及非法元器件的合并优化。针对该非确定性多项式（NP）难问题，开发了一种精确

算法来解决。我们还设计了不同规模的数值实验，通过应用权威竞赛数据作为基准，评估了数学模型和算法的准确性及有效性。此外，我们讨论了芯片区域的纵横比、不同高度元器件的比例、元器件总数和bin总数对芯片设计企业运营管理的影响。我们的发现为指导现代技术下的二维布局提供了宝贵的见解，表明MILP模型和精确算法的结合可以帮助芯片设计企业节省布局成本。

本文的其余部分组织如下：第2节进行文献综述；第3节详述布局优化问题；第4节建立MILP模型；第5节开发精确算法；第6节阐述数值实验；第7节总结研究。

2. 文献综述

本研究聚焦MCH电路布局优化问题，并设计精确算法求解。因此，本节从两个角度回顾了以往的相关文献：布局优化问题和布局优化相关问题的算法设计。

作为芯片设计过程中的一个重要阶段，布局优化根据上游信息在合法区域内放置可用元器件。为降低企业成本，在布局优化过程中需要考虑元器件高度、阈值电压类型和设计规则。Wu和Chu [6]研究包含单、双行高元器件的MCH电路，通过构建最大加权匹配问题生成候选元器件对。他们的方法在鲁棒性与质量方面表现良好，但忽略现代技术且方案现已不合法。

随着芯片设计的不断发展，电压漏极（VDD/VSS）对齐技术吸引了研究人员越来越多的关注。Chow等[7]表明，具有多行高度的元器件在行之间不是独立的，他们首次考虑了MCH电路的合法布局。他们提出的算法在区域元器件移动和缓冲器插入方面特别有效，通过最小化元器件总位移来提高初始布局的质量。Spindler等[8]提出的Abacus方法可以有效处理单行高度元器件的布局，但不

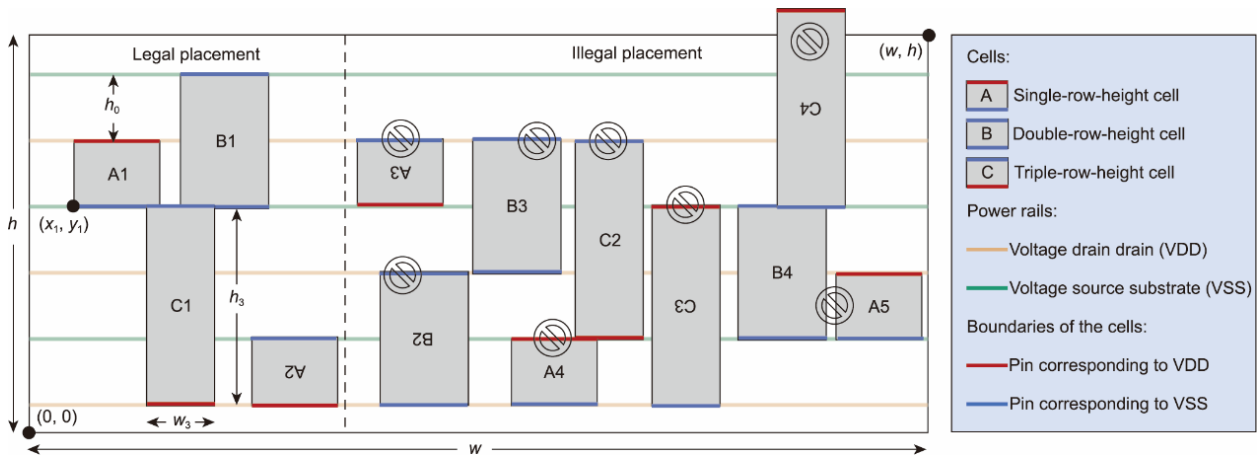


图1. 芯片设计中合法及非法布局。

适用于MCH电路。Wang等[4]通过利用Abacus的特性,设计了一种优化方案和一个特别适合芯片设计的目标函数,解决了对电路类型的限制。与分别优化多行和单行高度元器件布局的方法不同,Lin等[9]同时优化了各种高度的元器件。他们提出了一种包含链式移动方案和由密度驱动布局决定的嵌套动态规划的布局方法,解决了有序多行布局问题。他们的布局比逐个合法化元器件的上述工作实现了更好的合法化解决方案。

随着现代电路设计复杂性的增加,Zhu等[5]考虑了特殊元器件以实现电路各方面性能的平衡。这些特殊元器件对布局设计提出了重大挑战,特别是在增加合法化过程的难度方面。他们通过首先确保元器件满足对齐技术的要求,然后逐个调整非法放置的特殊元器件,优化了现有的布局方案。在器件尺寸逐渐缩小的进程中,最小注入面积(MIA)约束成为MCH电路新挑战。Chen等[10]首创将违反MIA的元器件聚类为具有相同阈值电压(V_{th})的元器件的方法,然后对组合的元器件进行排序以获得理想的元器件顺序。他们的方法在不需要额外面积开销的情况下有效解决了MIA违规问题。

MCH电路中的布局优化是一个高度复杂的问题,我们在表1中总结了一些代表性的工作[4–7,9–10]。在该表中,我们概述了上述研究解决的问题的具体特征。多数研究聚焦常规单/多行高元器件,对特殊元器件关注有限。近年研究日益重视元器件 V_{th} 类型差异,一些研究在现有布局方案的基础上开发方法以实现合法化,这些研究大多以最小化元器件总位移为优化目标。元器件的不重叠约束是布局优化的基本条件,但在考虑复杂的现代设计技术时,实现100%的不重叠可能具有挑战性。现有研究多通过将布局规则(如边缘布局)与启发式算法相结合来获得布局解决方案,然而由于这些规则无法覆盖所有可能的情况,所得布局无法完全保证元器件的不重叠放置。同时满

足不重叠和VDD/VSS对齐约束的布局优化研究被认为是有效的,布局合法化已成为近期研究的重要趋势。由于元器件类型的多样性和内部引脚配置,实现VDD/VSS对齐并非易事。现有研究常将其他类型元器件分解为标准元器件(即单行高度元器件)组合处理,然后组合这些标准元器件以解决对齐问题。然而,这种分解和合并的方法不适用于特殊元器件,因为它们的引脚位于元器件内部,无法轻易映射到标准元器件。MIA约束作为MCH电路布局优化中的新兴研究,相关研究较少,其解决方法也不成熟。

为了帮助芯片设计企业在布局中应对各种类型的元器件,本研究根据元器件的高度和布线方式考虑了四种类型的元器件。元器件的 V_{th} 类型会影响将元器件分配到bin的决策以及违反MIA约束的合并元器件。因此,我们在工作中也考虑了这一因素。与之前专注布线长度优化的研究不同,本研究考虑了布局成本,以提高芯片区域的利用率。这三种先进技术仍值得进一步探索。我们将不重叠约束纳入模型,通过约束强制实现100%的不重叠。通过引入VDD/VSS对齐约束,我们实现了各种元器件引脚与电源导轨的直接对齐。我们提出了一种聚合策略,将具有相同 V_{th} 的元器件水平合并为更大的元器件,从而避免违反MIA约束。在现有研究的基础上,我们通过数学建模系统地呈现了这些先进的设计规则,旨在提供稳定且合法的布局解决方案。

随着芯片设计规模的扩大,布局优化方法不断更新。最初,布局优化问题主要通过基于分区的方法[11]和启发式算法[12]解决。随着这两种技术的应用,布局优化问题的规模呈指数增长。为了获得大规模布局的解决方案,Agnihotri等[13]研究了元器件在较小子区域中的排列方式,并开发了最小切割方法。然而,由于子区域划分的随机性,子区域的解决方案与整体布局的相关性较低,导致解决方案质量不尽如人意。

表1 MCH电路布局优化的代表性工作

Papers	Height type of cell				V_{th} type of cell	Optimization objective (minimize)	Rules		
	S*	D*	T*	Sp			Non-ovlp	VDD/VSS	MIA
Wu and Chu [6]	√	√	–	–	Consistent	The total wirelength	√	–	–
Chow et al. [7]	√	√	√	–	Consistent	The total cell displacement	√	√	–
Wang et al. [4]	√	√	√	–	Consistent	The total cell displacement on the x-axis	√	√	–
Lin et al. [9]	√	√	√	–	Consistent	The half-perimeter wirelength	√	√	–
Zhu et al. [5]	√	√		√	Inconsistent (2)	The total cell displacement	√	√	–
Chen et al. [10]	√	√	√	–	Inconsistent (3)	The cluster area of cells	–	√	√
This study	√	√	√	√	Inconsistent (3)	The placement cost	√	√	√

S*, D*, T*, and Sp represent cells of single, double, triple, and special heights, respectively. “Non-ovlp” represents for non-overlapping. The numbers in parentheses represent the number of different V_{th} types available for cells, for example, “inconsistent (2)” indicates that the study considers two V_{th} types of cells.

芯片尺寸的缩小和设计规模的扩大促使布局算法进一步发展。分析方法在过去十年的学术竞赛中广泛应用,如设计自动化会议[14]、国际物理设计研讨会[15]和国际计算机辅助设计会议[16]。分析方法主要包括二次布局和非线性优化[17]。二次布局构建了一个便于求解的凸优化模型,但由于模型与实际布局差异较大,解决方案质量不符合芯片设计标准。非线性优化的思想是通过添加基于密度的惩罚成本来近似布线长度,从而生成一个非线性目标函数,然后通过非线性优化方法解决布局问题[18]。

然而,现有的布局优化算法仍无法完全满足解决方案质量和计算速度的双重要求,需要进一步开发优化算法,以应对不断增加的布局规模和日益复杂的设计规则。目前,图形处理元器件加速布局设计[19]和强化学习方法[20]引起了研究人员的广泛关注。然而,这两种方法仍处于探索阶段,在面对复杂且不断变化的设计技术时,其性能和稳定性有待进一步提高。因此,我们将注意力转向与芯片布局设计相似的其他问题,试图通过利用这些经典问题的求解方法来探索本研究的解决方案。

基于二维BPP的芯片布局优化更加强调微元(如元器件)的精确定位和连接,以满足电路的功能和性能要求,其约束条件更为复杂和精确。布局优化问题是BPP的一种变体,因此求解BPP的高效技术也可以应用于布局优化问题。自20世纪60年代以来,BPP发展迅速。基于实际背景,出现了许多BPP的变体,如机械加工行业的切割库存问题[21]、造纸生产行业的切割过程优化[22]以及供应链中的集装箱装载问题[23]。鉴于布局优化对解决方案质量的高要求,应用于BPP的精确算法可以作为参考。启发式算法[24]和近似算法[25]可以为精确算法提供初始解决方案。虽然模拟退火被认为是BPP最合适的启发式算法,但BD算法在BPP研究中的应用频率也显著增加[26]。

与BPP相比,切割和装箱问题也旨在优化给定空间

或材料中物体的排列,然而区别在于,切割和装箱问题关注如何将较大物体转化为较小物体,以最大化切割的价值,而BPP则强调如何有效安排各种尺寸的物体。这两个问题都以有效利用空间为共同目标,因此都适用于芯片布局设计。表2对这三个相关问题进行了全面分析。因此,我们总结了切割和装箱问题的求解方法,以适应本研究的求解方法。Serairi和Haouari [27]研究了具有固定方向的二维BPP,该问题在切割和装箱中广泛应用。他们开发了用于建立下界的理论框架,并进行了实证评估,强调了关系在提高装箱效率和解决方案质量中的重要性。Queiroz等[28]通过约束规划算法确保候选解决方案遵守二维装箱约束。他们还探索了界限和有效切割,同时结合了禁忌搜索和模拟退火的启发式方法以迭代生成解决方案。Delorme等[26]总结了近20年来切割和装箱问题的研究,指出最流行的求解方法是MILP和分支定界,并注意到近年来BD已成为求解切割和装箱问题的越来越受欢迎的选择。

综上所述,关于布局优化问题的文献总体趋势是考虑MCH电路和一些基本条件,这种组合适用于大多数芯片设计工作,使其成为研究人员的热门选择。本研究在此基础上,进一步考虑了Vth类型和一些现代电路设计技术对布局的影响。据我们所知,我们是第一个将布局优化问题表述为MILP模型的研究。此外,鉴于BD在BPP与切割和装箱问题中的应用近年来不断增加,本研究提出了一种基于BD的精确算法来解决所提出的问题,寻求速度和解决方案质量之间的平衡。

3. 问题背景

在提出数学模型之前,本节详细介绍了芯片布局的两个要素:第3.1节描述了布局优化问题;第3.2节陈述了若干布局设计准则。

表2 布局优化与其他两个问题的比较分析

Optimization problem	Objective	Constraints	Complexity	Solution methods	Focus point	Potential challenges
Placement in this study	Minimizing placement cost	Basic rules (e.g., cells of the same type clustered in a bin) non-overlap, VDD/VSS, and MIA	Involving electrical and physical characteristics, complex design rules	Mathematical programming, graph theory, etc.	Costs and modern design techniques	High technical complexity, lengthy design times
Bin packing	Maximize box utilization	Item sizes, box capacity (e.g., limit on maximum size and weight)	Relatively simple, focusing mainly on dimensions	Greedy algorithms, approximate algorithms, etc.	Matching item and box sizes	Diverse item types, dynamic changes
Cutting and packing	Maximizing material utilization	Cutting methods, material properties (e.g., strength and thickness limitations)	Includes various cutting methods and material configurations	Linear programming, dynamic programming, etc.	Material cutting and configuration	Resource and material traits affecting effectiveness

3.1. 布局优化问题

芯片布局实际上是一个位置分配问题，确定元器件在芯片区域内的坐标。为了获取元器件的精确坐标，我们假设将矩形芯片区域置于二维笛卡尔坐标系中。如图2所示，坐标系的原点 o 对应芯片区域的左下角坐标， x 轴和 y 轴分别沿芯片区域的宽度和高度方向延伸。 VDD 为芯片提供漏极电压， VSS 为地线。 VDD 和 VSS 导轨之间的间距称为行高，是固定的。元器件的最小宽度称为位点。位点和行高分别是 x 轴和 y 轴的单位长度。换句话说，芯片区域是由 VDD 和 VSS 导轨交错在布局行之间的电路组成。

此外，每个元器件包含两种不同的引脚类型： VDD 和 VSS [6]。当元器件上的 VDD/VSS 引脚与芯片区域中的 VDD/VSS 导轨放置在相应位置时，元器件可以在电路中发挥作用。元器件的形状统一为矩形，其宽度和高度分别是 x 轴和 y 轴上单位长度的倍数。

根据引脚连接和高度的不同，可以区分四种类型的元器件：A、B、C和D。A、B和C型元器件的引脚位于其上下边界，高度分别等于单行、双行和三行高度。D型元器件的高度也是双行高度，但其引脚位于矩形内部[5]，形成更专业的布线配置。

除了这些差异，元器件还可以根据其高、中、低 V_{th} 分为三组。 V_{th} 仅与元器件注入的介质有关，与其高度或布线方式没有固定关联。例如，在图2中，具有单行高度的元器件A1属于低 V_{th} ，而具有相同高度的元器件A2具有标准 V_{th} 。具有多种 V_{th} 类型的元器件可以实现电路在速度和泄漏功率之间的平衡，而具有相同 V_{th} 的元器件聚集可以进一步提高电路的效率[29]。

由于元器件之间的差异，在布局决策中需要考虑诸如相同 V_{th} 类型元器件的聚集、元器件密度和芯片区域利用率等因素，这不可避免地使布局更具挑战性。为了实现元器件数量、类型和坐标之间的合理匹配，本研究采用了一种方法，即在芯片区域上反复执行水平和垂直切割，将完整的芯片区域划分为多个较小的矩形块，称为bin。在布局的初步阶段，确定了放置相同元器件类型的区域，而未指定bin和元器件类型的匹配或每个bin的元器件数量。这种划分有助于更精确的元器件布局，遵守关键的设计约束（如布线），同时通过将复杂的布局区域划分为较小的bin来减少全局搜索所需的计算资源。

为了便于理解切割操作，我们在图3中展示了一个切割示例及其顺序。经过六次交替的水平和垂直切割，芯片区域被划分为七个bin。芯片的功能分区可用于计算bin的面积和数量。每个bin内部的空间决定了其可容纳元器件的上限。将芯片区域划分为bin的目的是确保每个bin中存储的元器件具有相同的 V_{th} 。在确定每个bin对应的 V_{th} 类型后，将给定数量的元器件与bin匹配，以实现相同 V_{th} 元器件的聚集。这种划分机制可以避免元器件在局部电路中过度集中，并实现元器件在芯片区域内的均匀分布。

3.2. 布局设计准则

在布局之前，元器件是相互独立的，其坐标未固定。切割布局的目的是提高空间利用率并减少电路的布线总长度，从而降低电路中的延迟并提高稳定性[30]。在澄清了芯片区域的结构特征、元器件类型和bin划分后，本研究在一些布局设计准则下进行优化。

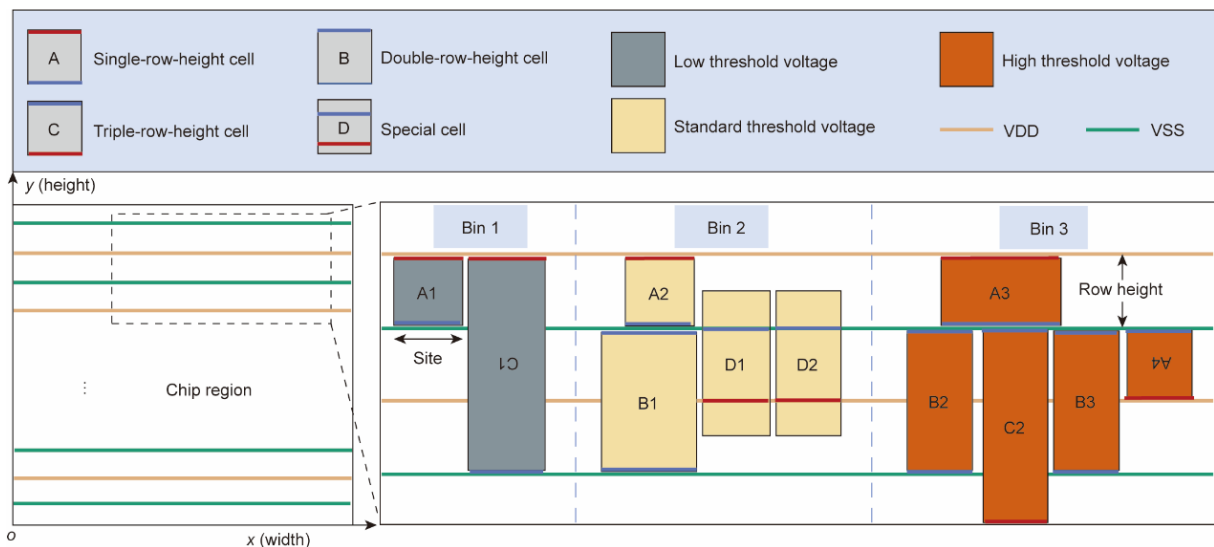


图2. 芯片布局区域和元器件类型。

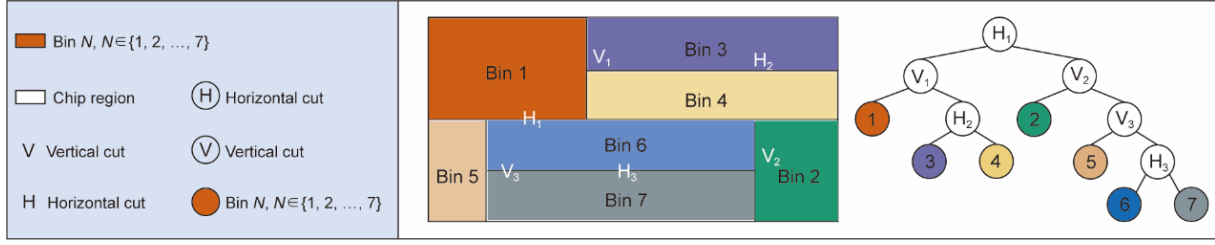


图3. 布局切割示例。

由于布局成本直接影响企业的经济绩效，并与产品质量和客户满意度相关，我们进行了一个旨在降低成本的优化过程。布线长度是衡量布局成本最常用的指标[31]，布线结果也可用于评估布局的质量。我们从两个方面衡量布线长度：布局的紧凑性和布线总长度，将目标从降低成本转变为缩短布线长度。元器件的更密集排列意味着更高的空间利用率和更短的布局布线长度。半周布线长度（HPWL）方法是一种非常准确和快速的布线长度估计方法[32]。在本研究中，我们使用两个指标来评估布局的紧凑性和布线效率。第一个指标是紧凑性度量，使用HPWL方法评估。具体来说，通过计算元器件沿 x 轴和 y 轴的最大距离之和来测量紧凑性。第二个指标是布线总长度，表示为每个元器件到原点 o 的曼哈顿距离的总和。我们将元器件 i 的坐标设为 (x_i, y_i) ，原点 o 的坐标为 $(0,0)$ ，元器件集合为 C ，那么布线总长度可以表示为 $\sum_{i \in C} (x_i + y_i)$ 。

图4演示了HPWL方法，其中芯片区域被划分为三个bin。以bin 1为例， x 轴上的最大距离存在于元器件A5和C1之间，而 y 轴上的最大距离在A5和A1之间。因此，bin 1的HPWL值为10.5。类似地，我们可以计算bin 2和bin 3的HPWL值，最终得到整个布局的总HPWL值为34。

基于上述方法，布线总长度的计算可以表示为

$$\sum_{z \in Z} \sum_{l \in L} \left[\max_{\substack{i, i', i'', i''' \in C \\ i \neq i', i' \neq i'', i'' \neq i'''}} (x_i \mu_{i,z,l} + y_i \mu_{i',z,l} - x_{i'} \mu_{i'',z,l} - y_{i''} \mu_{i''',z,l}) \right], \text{ 其}$$

中 Z 表示bin集合； L 表示元器件Vth类型集合； z 和 l 分别是上述两个集合的索引； $\mu_{i,z,l}$ 是一个二进制变量，如果元器件 i 的类型 l 在bin z 中，则为1，否则为0。然而，这个表达式的计算复杂度为 $O(|Z| + |L| + 4|C|)$ 。考虑到芯片布局的可扩展性，我们决定修改上述布线长度计算公式以提高求解速度。

我们将芯片区域视为一个整体，直接计算整个电路的HPWL。HPWL方法适用于小规模布局的电路；当布局规模较大时，HPWL比实际布线长度短。为了避免产生这种误差，我们将HPWL乘以一个校正因子。考虑到布线长度与元器件数量成正比，但与bin数量和Vth类型数量成反比，我们在测试后将校正因子设为 $\sqrt{|C|/(1.5|Z| + |L|)}$ 。修改后的HPWL公式为 $\sqrt{|C|/(1.5|Z| + |L|)} \max_{\substack{i, i', i'', i''' \in C \\ i \neq i', i' \neq i'', i'' \neq i'''}} (x_i + y_i - x_{i'} - y_{i''})$ ，其计算复杂度为 $O(|C|)$ ，显著提高了求解效率。

在图4所示的示例中，修改后的HPWL'约为32.9，与原始HPWL的差距在4%以内。有关HPWL校正因子有效性的实验详见附录A中的第S1节。

在确定元器件的坐标时，需要考虑基本布局条件和现代电路设计技术的双重要求。基本布局条件决定了元器件

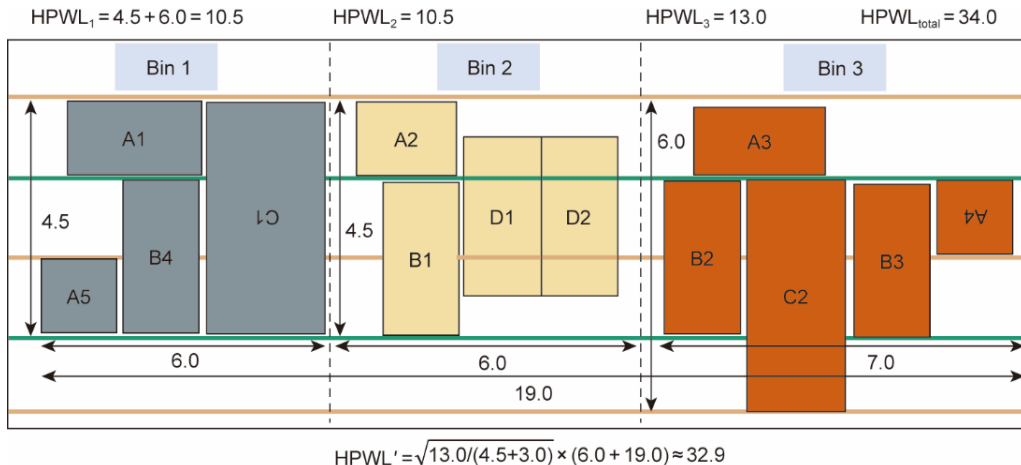


图4. 使用HPWL方法描述布线长度的示例。

的分配方案,现代电路设计技术确保了元器件的排列是合法的。在初始布局阶段,元器件被放置在芯片区域内,然后进行细致的匹配过程,将其坐标与相应的bin对齐。以合理的方式将元器件与bin匹配并确保bin中的每个元器件具有相同的 V_{th} 是一项挑战。同样重要的是确保每个元器件仅属于一个bin。值得注意的是,bin的容量是有限的,bin内的元器件数量不能超过容量。基本布局条件决定了各种元器件与bin之间的位置和数量关系,而现代电路设计技术对优化提出了更高的要求。

元器件在bin内不是随机放置的,它们的坐标必须遵守设计技术。本研究考虑了三个约束:元器件不重叠、VDD/VSS对齐和解决MIA违规。不重叠约束是合法化的基本条件[33],要求两个元器件的坐标在 x 轴或 y 轴上至少有一个不同。这一约束在实际应用中很有意义,因为重叠的元器件会增加布局的层数,复杂化优化工作。

VDD/VSS对齐约束对元器件的纵坐标提出了更具体的要求。图5展示了四种类型元器件在VDD/VSS对齐约束下的合法和非法条件。合法性的标准是元器件的引脚必须与相同类型的导轨匹配。由于A型和C型元器件具有相同的布线方式,它们可以通过垂直翻转放置在行之间,被统称为奇数行高元器件。B型元器件有两个相同类型的引脚,垂直翻转对它们没有意义[34]。D型元器件的布线方式类似于A型和C型元器件,因为它们的引脚在内部,其坐标位于两条相邻导轨的中间。总之,我们将元器件分为三种类型:单行高度(A和C)、双行高度(B)和特殊(D)。我们进一步根据布线方式确定元器件纵坐标与行高之间的关系。

随着元器件尺寸的不断缩小,芯片设计的复杂性和精细度增加。然而,更小的元器件并不总是更可取的。MIA约束规定了元器件的最小允许面积,违反此约束的元器件需要进一步处理。由于元器件的面积是给定的,解决

MIA违规有两种方法:一种是插入相同 V_{th} 的填充物以从根本上增加元器件的面积;另一种是聚集过小的元器件并将它们组合以遵守MIA约束。前一种方法增加了设计布局成本并导致空间浪费,而后一种方法经济且已被证明是有效的[10]。根据邻近原则,我们将违反MIA约束的非法元器件组合成一个新实体,确保组合后的面积超过MIA约束规定的下限。本研究基于聚合的思想扩展了元器件的宽度,并在水平方向上解决了MIA违规问题。

4. 考虑现代电路设计技术的建模

该模型包含各种复杂的设计技术,确定元器件的坐标,并通过使用整个电路的布线长度评估布局质量。我们在附录A中的第S2节中展示了以下符号及其与布局优化的关系。

4.1. 假设

为了优化布局,我们开发了一个MILP模型,考虑了元器件不重叠、VDD/VSS对齐和MIA约束。在构建模型之前,需要解释一些基本假设。

(1) 交错的VDD/VSS导轨:芯片区域中VDD和VSS导轨的交错旨在最小化地弹并提高信号完整性。这种技术还有助于更好的热分布,减少密集设计中的热问题。

(2) 元器件的高度规格:待布局的元器件高度最高为三行高度,其中特殊元器件的高度为两倍行高,但其布线方式与双行高度元器件不同。

(3) 元器件尺寸对布局的影响:每个元器件的物理尺寸直接影响其在布局中占用的面积,因为较大的元器件需要更多空间以避免干扰和信号延迟。每个bin的宽度和高度必须大于任何单个元器件的宽度和高度,以确保布局的稳定性和各种元器件的性能。

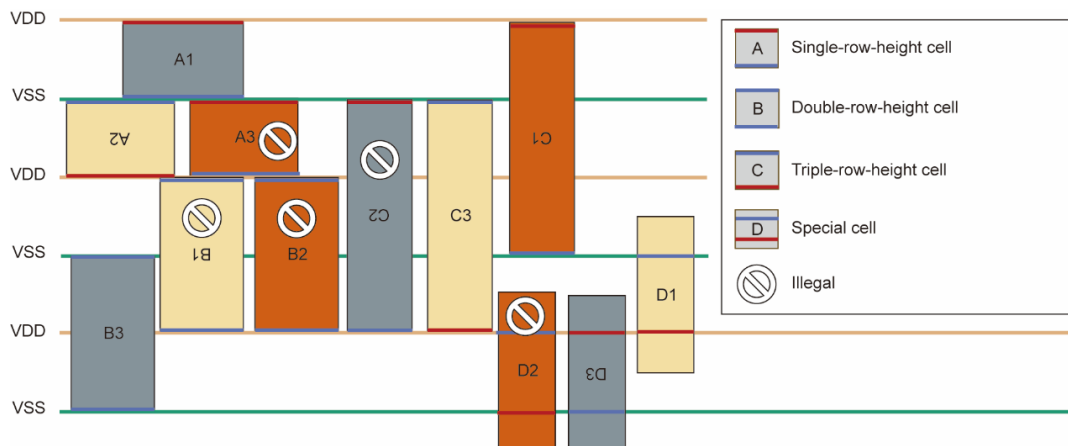


图5. VDD/VSS对齐约束下的合法和非法条件。

(4) 芯片区域分割为 **bin**: 将芯片区域分割为 **bin**, 每个 **bin** 都有面积和元器件容量的上限, 这对于有效的资源分配和高效的布局管理至关重要。

(5) 长度单位的设置: 在本研究中, 元器件的尺寸在纳米级, 而布局区域的尺寸在微米级。因此, 本研究全程采用微米作为长度单位。

4.2. 符号

我们定义了以下三种类型的符号来构建 MILP 模型: 集合用大写字母表示, 索引为相应的小写字母; 为了突出决策的内容, 变量用希腊字母表示; 上标用于进一步解释这些符号。

4.2.1. 索引和集合

- C 元器件的集合, 索引为 i ;
- C^s 单行高度元器件集合, $C^s \subset C$;
- C^d 双行高度元器件集合, $C^d \subset C$;
- C^t 三行高度元器件集合, $C^t \subset C$;
- C^{sp} 特殊双行高度元器件集合, $C^{sp} \subset C$;
- Z **bin** 的集合, 索引为 z ;
- L 元器件 Vth 类型集合, 索引为 l ;
- J 导线集合, 索引为 j ;
- J^0 与奇数行高元器件对应的导线集合, $J^0 \subset J$;
- J^1 与双行高度元器件对应的导线集合, $J^1 \subset J$;
- J^2 与特殊元器件对应的导线集合, $J^2 \subset J$ 。

4.2.2. 参数

- w 芯片区域的宽度;
- h 芯片区域的高度;
- w^0 x 轴上的单位长度(点位);
- h^0 y 轴上的单位长度(行高);
- w_i 元器件 i 的宽度;
- h_i 元器件 i 的高度;
- w_z^{bin} **bin** z 的宽度;
- h_z^{bin} **bin** z 的高度;
- f 切割系数;
- $(x_z^{\text{bin}}, y_z^{\text{bin}})$ **bin** z 的左下角坐标;
- w^{minw} 所有元器件的统一合法宽度;
- $k_{i,j}$ 导线 j 在 y 轴上的位置, 即元器件 i 引脚的放置处;
- M 一个足够大的正整数。

4.2.3. 决策变量

- (x_i, y_i) 连续变量, 元器件 i 左下角坐标;
- $\theta_{z,l}$ 整数变量, **bin** z 中 l 类型的元器件数量;
- $\mu_{i,z,l}$ 二进制变量, 如果类型为 l 的元器件 i 在 **bin** z 中, 则为 1, 否则为 0;
- $\omega_{z,l}$ 二进制变量, 如果类型为 l 的元器件位于 **bin** z 中, 则为 1, 否则为 0;
- $\alpha_{i,i'}$ 二进制变量, 如果元器件 i 位于元器件 i' 的左侧, 则为 1, 否则为 0;
- $\beta_{i,i'}$ 二进制变量, 如果元器件 i 位于元器件 i' 的下方, 则为 1, 否则为 0;
- $\rho_{i,l,j}$ 二进制变量, 如果类型为 l 的元器件 i 的引脚在导线 j 上, 则为 1, 否则为 0;
- $\gamma_{i,i'}$ 二进制变量, 如果元器件 i 和元器件 i' 组合成新的元器件, 则为 1, 否则为 0;
- τ_i 二进制变量, 如果元器件 i 的宽度小于 w^{minw} , 则为 1, 否则为 0。

4.3. 模型构建

使用上述符号定义的混合整数非线性规划 (MINLP) 模型如下:

$$[M0] \text{ Minimize } \left[\sqrt{|C|/(1.5|Z| + |L|)} \max_{\substack{i, i', i'', i''' \in C \\ i \neq i', i'' \neq i'''}} (x_i + y_{i'} - x_{i''} - y_{i'''}) \right] + \sum_{i \in C} (x_i + y_i) \quad (1)$$

目标 (1) 从两个方面最小化布局成本。如前所述, 布局的紧凑性是评估成本的第一个指标。修改后的 HP-WL' 不是根据最短直线距离计算的, 而是根据最长的曼哈顿距离, 这与我们从二维坐标角度的建模一致。第二个指标用于评估布局的布线总长度, 计算为所有元器件到原点的曼哈顿距离的总和。在实际布线中, 并非所有元器件都直接连接到原点, 因此这个计算结果是布线总长度的上界。然而, 由于在布局阶段无法获得具体的布线细节, 且该指标有效地捕捉了布线总长度的变化, 因此被认为是一种合理的评估方法。总之, 我们的模型通过提高芯片区域的利用率和最小化布线总长度来最小化布局成本。为了确保结果的稳健性, 我们在正文中没有为目标函数中的两个指标进行具体的权重分配。然而, 考虑到芯片设计公司在实际生产中可能对布局的紧凑性和布线总长度有不同的优先级, 我们在附录 A 中的第 S3 节中进行了额外的实验来分析这些方面。

约束条件为

$$0 \leq x_i \leq w - w_i \quad \forall i \in C \quad (2)$$

$$0 \leq y_i \leq h - h_i \quad \forall i \in C \quad (3)$$

$$\sum_{l \in L} \theta_{z,l} \leq \frac{|C|}{f w^0 h^0} \quad \forall z \in Z \quad (4)$$

$$\theta_{z,l} = \sum_{i \in C} \mu_{i,z,l} \quad \forall z \in Z, l \in L \quad (5)$$

$$\sum_{l \in L} \sum_{z \in Z} \mu_{i,z,l} = 1 \quad \forall i \in C \quad (6)$$

$$\sum_{l \in L} \omega_{z,l} = 1 \quad \forall z \in Z \quad (7)$$

$$\theta_{z,l} \leq M \omega_{z,l} \quad \forall z \in Z, l \in L \quad (8)$$

$$\sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} \mu_{i,z,l} \geq \sum_{z \in Z} \sum_{l \in L} \theta_{z,l} \omega_{z,l} \quad (9)$$

$$x_z^{\text{bin}} \leq x_i \mu_{i,z,l} + x_z^{\text{bin}} (1 - \mu_{i,z,l}) \leq x_z^{\text{bin}} + w_z^{\text{bin}} - w_i \quad \forall i \in C, z \in Z, l \in L \quad (10)$$

$$y_z^{\text{bin}} \leq y_i \mu_{i,z,l} + y_z^{\text{bin}} (1 - \mu_{i,z,l}) \leq y_z^{\text{bin}} + h_z^{\text{bin}} - h_i \quad \forall i \in C, z \in Z, l \in L \quad (11)$$

约束 (2) ~ (11) 是关于基本布局条件的。约束 (2) 和约束 (3) 确保元器件放置在芯片区域内。约束 (4) 确保每个 bin z 中所有 Vth 类型的元器件总数不超过由元器件总数、元器件尺寸和切割系数共同确定的密度阈值，以防止布局设计过于拥挤或过于稀疏。约束 (5) 计算 bin z 中 l 类型的元器件的实际数量，这是关于两个变量 $\theta_{z,l}$ 和 $\mu_{i,z,l}$ 的连接约束。约束 (6) 确保元器件 i 只能匹配一个类型 l 和 bin z 。约束 (7) 要求 bin z 内的元器件类型 l 一致。约束 (8) 规定了 bin 中可以放置的元器件类型，而约束 (9) 进一步将 bin 与元器件关联。约束 (10) 和约束 (11) 确保类型 l 的元器件 i 的坐标应对应于 bin z 。

约束 (12) ~ (27) 对应现代电路设计技术。

$$x_{i'} - x_i + M(1 - \alpha_{i,i'}) \geq w_i \quad \forall i, i' \in C, i \neq i' \quad (12)$$

$$y_{i'} - y_i + M(1 - \beta_{i,i'}) \geq h_i \quad \forall i, i' \in C, i \neq i' \quad (13)$$

$$\alpha_{i,i'} + \alpha_{i',i} + \beta_{i,i'} + \beta_{i',i} \geq 1 \quad \forall i, i' \in C, i \neq i' \quad (14)$$

$$\alpha_{i,i'} + \alpha_{i',i} \leq 1 \quad \forall i, i' \in C, i \neq i' \quad (15)$$

$$\beta_{i,i'} + \beta_{i',i} \leq 1 \quad \forall i, i' \in C, i \neq i' \quad (16)$$

约束 (12) ~ (16) 确保元器件 i 和 i' 在 x 或 y 坐标上至少有一个差异，从而保证这两个元器件在二维坐标系中不重叠。

$$y_i = \sum_{l \in L} \sum_{j \in J^0} k_{i,j} \rho_{i,l,j} \quad \forall i \in C^s \cup C^d \cup C^t \quad (17)$$

$$\sum_{l \in L} \sum_{j \in J} \rho_{i,l,j} = 1 \quad \forall i \in C \quad (18)$$

$$\sum_{l \in L} \sum_{j \in J^0} \rho_{i,l,j} = 1 \quad \forall i \in C^s \cup C^t \quad (19)$$

$$\sum_{l \in L} \sum_{j \in J^2} \rho_{i,l,j} = 1 \quad \forall i \in C^d \quad (20)$$

$$y_i + \frac{1}{4} h_i = \sum_{l \in \{1,2\}} \sum_{j \in J^2} k_{i,j} \rho_{i,l,j} \quad \forall i \in C^{\text{sp}} \quad (21)$$

$$\sum_{l \in L} \sum_{j \in J^2} \rho_{i,l,j} = 1 \quad \forall i \in C^{\text{sp}} \quad (22)$$

$$\sum_{l \in L} \sum_{j \in J} \left(\sum_{i \in C^s \cup C^d} \rho_{i,l,j} + \sum_{i \in C^d} \rho_{i,l,j} + \sum_{i \in C^{\text{sp}}} \rho_{i,l,j} \right) = |C| \quad (23)$$

约束 (17) ~ (23) 是关于将元器件上的引脚与 VDD 和 VSS 导轨匹配的。约束 (17) 规定了非特殊元器件的纵坐标，并根据其类型将它们分配到相应的导轨。允许的 j 值集合（表示导线索引）随元器件类型而变化。对于可以翻转放置的单行高度和三行高度元器件，允许的 $k_{i,j}$ 范围更广，以适应这种灵活性。约束 (18) 表示对于一个元器件，必须有且只有一个 Vth 类型，并且其引脚只能与一条导线相关联。约束 (19) 声明单行、三行高度元器件 i 应与类型 l 匹配，并对应于导线 j 。约束 (20) 规定双行高度元器件 i 与类型 l 和导线 j 之间的关系。约束 (21) 定义特殊元器件的纵坐标，建立这些元器件内部引脚与布局区域内电源导轨的对应关系。约束 (22) 表示特殊元器件 i 的纵坐标应与类型 l 和导线 j 对应。约束 (23) 规定所有类型元器件在所有可能导线上的总和应等于待放置的元器件数量，确保没有元器件被遗漏。值得注意的是，由于每个元器件上的引脚位置是预先确定的，元器件的放置方向（原始或翻转）可以从其纵坐标和相应的导线分配中推断出来。

$$w_i + \tau_i \geq w^{\text{minw}} \quad \forall i \in C \quad (24)$$

$$\sum_{i' \in C, i \neq i'} \gamma_{i,i'} = \tau_i \quad \forall i \in C \quad (25)$$

$$w_i + w_{i'} \geq \gamma_{i,i'} w^{\text{minw}} \quad \forall i, i' \in C, i \neq i' \quad (26)$$

$$x_{i'} - x_i - M(2 - \gamma_{i,i'} - \alpha_{i,i'}) \leq w_i \quad \forall i, i' \in C, i \neq i' \quad (27)$$

约束 (24) ~ (27) 是关于解决 MIA 违规的。约束 (24) 比较元器件 i 的宽度和 w^{minw} 以确定 MIA 违规。由于第 4.1 小节中的假设 (5)，确保了约束 (24) 的有效性。约束 (25) 表示如果元器件 i 的宽度小于 w^{minw} ，它将与其它元器件组合。约束 (26) 要求组合元器件的宽度超过 w^{minw} 。约束 (27) 表示组合元器件在 x 轴上的坐标是连续的。

$$0 \leq x_i \leq w \quad \forall i \in C \quad (28)$$

$$0 \leq y_i \leq h \quad \forall i \in C \quad (29)$$

$$0 \leq \theta_{z,l} \leq |C| \quad \forall z \in Z, l \in L \quad (30)$$

$$\mu_{i,z,l}, \omega_{z,l} \in \{0, 1\} \quad \forall i \in C, z \in Z, l \in L \quad (31)$$

$$\alpha_{i,i'}, \beta_{i,i'}, \gamma_{i,i'}, \tau_i \in \{0, 1\} \quad \forall i, i' \in C, i \neq i' \quad (32)$$

$$\rho_{i,l,j} \in \{0, 1\} \quad \forall i \in C, l \in L, j \in J \quad (33)$$

约束 (28) ~ (33) 定义了决策变量的定义域。

4.4. MILP 模型的构建

上述模型包含几个难以直接使用求解器求解的部分，如目标函数中最小化最大值的结构和约束 (9) ~ (11) 中的非线性部分。如果不解决这些问题，模型的计算工作量将增加，进而导致设计周期延长。

4.4.1. $M0$ 中目标函数的预处理

对于目标函数 (1) 中的第一部分, 即 $\sqrt{|C|/(1.5|Z|+|L|)} \max_{\substack{i,i',i'',i''' \in C \\ i \neq i', i'' \neq i'''}} (x_i + y_{i'} - x_{i''} - y_{i'''})$, 我们定义一个额外的非负决策变量 δ 。随后, 公式 (1) 可以通过添加以下约束来构建:

$$\delta \geq x_i + y_{i'} - x_{i''} - y_{i'''} \quad \forall i, i', i'', i''' \in C, i \neq i', i'' \neq i''', z \in Z, l \in L \quad (34)$$

使用 δ 表示芯片区域内的最大曼哈顿距离, 将最小化最大值的双重目标转化为单目标, 允许模型通过 CPLEX (IBM ILOG CPLEX Optimization Studio) 求解。修订后的目标函数见第 4.4.3 小节。

4.4.2. $M0$ 中约束的线性化

约束 (9) 中的非线性是由决策变量的乘积引起的。令 $\pi_{z,l}$ 表示 $\theta_{z,l}\omega_{z,l}$, 约束 (9) 可以修改为方程 (35) ~ (39)。线性化过程在附录 A 推导 1 中描述。

$$\pi_{z,l} = \theta_{z,l}\omega_{z,l} \quad \forall z \in Z, l \in L \quad (35)$$

$$\pi_{z,l} \geq \omega_{z,l} \quad \forall z \in Z, l \in L \quad (36)$$

$$\pi_{z,l} \geq \theta_{z,l} - |C| + |Z| - 1 + (|C| - |Z| + 1)\omega_{z,l} \quad \forall z \in Z, l \in L \quad (37)$$

$$\pi_{z,l} \leq (|C| - |Z| + 1)\omega_{z,l} \quad \forall z \in Z, l \in L \quad (38)$$

$$\pi_{z,l} \leq \theta_{z,l} + \omega_{z,l} - 1 \quad \forall z \in Z, l \in L \quad (39)$$

类似地, 我们使用 $\sigma_{i,z,l}$ 和 $\varphi_{i,z,l}$ 替换约束 (10) 和约束 (11) 中的双线性表达式, 这两个约束可以分别替换为方程 (40) ~ (44) 和方程 (45) ~ (49)。这两个线性化过程详细记录在附录 A 的推导 2 和推导 3 中。

$$\sigma_{i,z,l} = x_i \mu_{i,z,l} \quad \forall i \in C, z \in Z, l \in L \quad (40)$$

$$\sigma_{i,z,l} \geq 0 \quad \forall i \in C, z \in Z, l \in L \quad (41)$$

$$\sigma_{i,z,l} \geq x_i - w + w_i + (w - w_i)\mu_{i,z,l} \quad \forall i \in C, z \in Z, l \in L \quad (42)$$

$$\sigma_{i,z,l} \leq (w - w_i)\mu_{i,z,l} \quad \forall i \in C, z \in Z, l \in L \quad (43)$$

$$\sigma_{i,z,l} \leq x_i \quad \forall i \in C, z \in Z, l \in L \quad (44)$$

$$\varphi_{i,z,l} = y_i \mu_{i,z,l} \quad \forall i \in C, z \in Z, l \in L \quad (45)$$

$$\varphi_{i,z,l} \geq 0 \quad \forall i \in C, z \in Z, l \in L \quad (46)$$

$$\varphi_{i,z,l} \geq y_i - h + h_i + (h - h_i)\mu_{i,z,l} \quad \forall i \in C, z \in Z, l \in L \quad (47)$$

$$\varphi_{i,z,l} \leq (h - h_i)\mu_{i,z,l} \quad \forall i \in C, z \in Z, l \in L \quad (48)$$

$$\varphi_{i,z,l} \leq y_i \quad \forall i \in C, z \in Z, l \in L \quad (49)$$

这种通过放宽可行域边界的线性化方法使模型能够从非线性转换为线性, 并获得全局最优解的下界。与引入大 M 进行线性化相比, 这种方法使用的计算空间更少, 求解速度更快。

4.4.3. 混合整数规划模型 $M1$

基于这些新的决策变量和约束, 模型 $M0$ 可以重新表述为 MILP 模型 $M1$ 。

$$[M1] \text{ Minimize } \sqrt{|C|/(1.5|Z|+|L|)} \delta + \sum_{i \in C} (x_i + y_i) \quad (50)$$

约束条件为

式 (2) ~ (8) 以及式 (12) ~ (49)。

第 4.4 节的核心内容总结在图 6 中, 描述了线性化过程。线性化的目的是使模型能够直接由 CPLEX 求解, 并便于应用精确算法。在初始模型 $M0$ 中, 有两个部分需要预处理: 非凸目标函数和约束 (9) ~ (11) 中的非线性表达式。通过添加新的决策变量 δ , 实现从 Min-Max 问题到最小化问题的转换。通过引入新的辅助变量来替换原始决策变量的乘积, 并根据决策变量的定义域定义差异变量, 我们可以推导出线性化约束。随后, 推导出新的 MILP 模型 $M1$, 可以轻松求解。对于更实际的中等或大规模实例, CPLEX 无法在指定设计周期内得到解决方案。因此, 在下一节中设计了一种合适的算法来提高求解效率。

5. 精确算法

考虑到现代电路设计技术, 布局优化问题面临重大挑战。在设计周期中, 商业求解器 (如 CPLEX) 在数据规模扩大时几乎无法计算结果, 而在实际生产中, 元器件数量极大, 需要开发一种为该模型量身定制的算法。BD 以求解大规模优化问题而闻名, 广泛应用在供应链[35]、材料运输[36]等领域。我们将迭代优化的思想融入布局模型中, 寻求解决方案质量和计算速度之间的平衡。

5.1. 经典 BD

BD 的主要思想是根据求解难度将决策变量分为两类, 从而将复杂问题分解为主问题 (MP) 和子问题

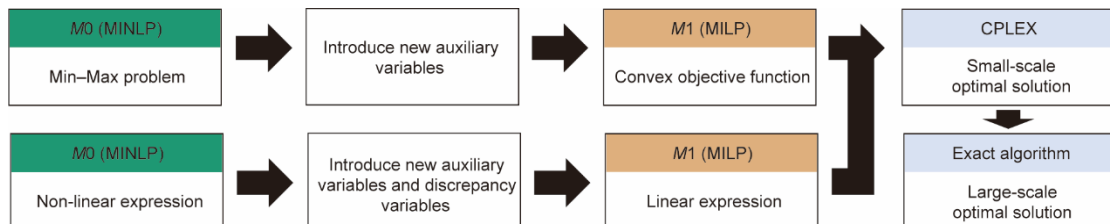


图 6. 模型预处理的流程。

(SP)。通过迭代求解这两个小模型，我们可以比直接求解原始模型更容易地获得结果。

在本研究中，决定元器件、bin、Vth和导轨之间相对位置关系的决策变量包括 $\theta_{z,l}$ 、 $\mu_{i,z,l}$ 、 $\omega_{z,l}$ 、 $\alpha_{i,i'}$ 、 $\beta_{i,i'}$ 、 $\gamma_{i,i'}$ 、 τ_i 以及 $\rho_{i,l,j}$ 。上述间接确定元器件坐标的变量统一放置在MP中。然而，与元器件坐标直接相关的核心变量 x_i 和 y_i 被分配到SP中以做出最终决策。MP中的二进制和整数变量被代入SP中，然后从SP中计算最优割，并添加到MP中，这极大地提高了求解效率。结合两级解耦的思想，我们迭代优化原始问题的上、下界以获得最优解。

5.1.1. SP的构建

SP根据MP获得的相对位置关系确定元器件的最终坐标。元器件的纵坐标受VDD/VSS对齐约束的影响，在决定元器件坐标的过程中可能存在一定的随机性，因此确保SP有解至关重要。首先，SP为原始问题提供了上界，表述如下：

$$[\text{SP}] \text{ Minimize } \sqrt{|C|(1.5|Z|+|L|)} \delta + \sum_{i \in C} (x_i + y_i) \quad (51)$$

约束条件为

$$\text{式 (2)、(3)、(28)、(29)、(34)、(41)、(44)、(46)、(49);}$$

$$x_z^{\text{bin}} - x_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) \leq x_i \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (52)$$

$$x_i \widehat{\mu_{i,z,l}} \leq x_z^{\text{bin}} + w_z^{\text{bin}} - w_i - x_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) \quad \forall i \in C, z \in Z, l \in L \quad (53)$$

$$y_z^{\text{bin}} - y_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) \leq y_i \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (54)$$

$$y_i \widehat{\mu_{i,z,l}} \leq y_z^{\text{bin}} + h_z^{\text{bin}} - h_i - y_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) \quad \forall i \in C, z \in Z, l \in L \quad (55)$$

$$x_{i'} - x_i + M(1 - \widehat{\alpha_{i,i'}}) \geq 0 \quad \forall i, i' \in C, i \neq i' \quad (56)$$

$$y_{i'} - y_i + M(1 - \widehat{\beta_{i,i'}}) \geq 0 \quad \forall i, i' \in C, i \neq i' \quad (57)$$

$$y_i = \sum_{l \in L} \sum_{j \in J^0} k_{i,j} \widehat{\rho_{i,l,j}} \quad \forall i \in C^u \cup C^d \cup C^t \quad (58)$$

$$y_i + \frac{1}{4} h_i = \sum_{l \in \{1,2\}} \sum_{j \in J^1} k_{i,j} \widehat{\rho_{i,l,j}} \quad \forall i \in C^{\text{sp}} \quad (59)$$

$$x_{i'} - x_i - M(2 - \widehat{\gamma_{i,i'}} - \widehat{\alpha_{i,i'}}) \leq w_i \quad \forall i, i' \in C, i \neq i' \quad (60)$$

$$\sigma_{i,z,l} = x_i \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (61)$$

$$\sigma_{i,z,l} \geq x_i - w + w_i + (w - w_i) \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (62)$$

$$\sigma_{i,z,l} \leq (w - w_i) \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (63)$$

$$\varphi_{i,z,l} = y_i \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (64)$$

$$\varphi_{i,z,l} \geq y_i - h + h_i + (h - h_i) \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (65)$$

$$\varphi_{i,z,l} \leq (h - h_i) \widehat{\mu_{i,z,l}} \quad \forall i \in C, z \in Z, l \in L \quad (66)$$

SP中的所有变量都是连续的，模型是线性的。带帽的

符号表示MP中的决策变量。为了确保迭代过程的顺利进行，我们首先设计了一个初始解生成规则。该规则要求按高度对所有元器件进行排序，然后从底部到顶部将元器件放置到布局区域中。使用这种方法，获得一个初始布局解，然后传递到SP中进行进一步优化。SP求解更精细的布局，并将元器件位置传递到MP中进行全局调整。这个过程在每次迭代中重复，每轮生成一个新的割。我们基于对偶理论推断最优割，并用它来收缩可行域，从而提高布局质量。

根据SP的约束，我们引入对偶变量 P_i^n ，其中 $n \in \{1, 2, \dots, 21\}$ 。这些对偶变量及其对应的约束通过加权和被纳入SP的原始目标函数中，生成一个增强的目标函数。这种表述通过将约束直接集成到目标中简化了问题。所得拉格朗日松弛表达式如下：

$$\begin{aligned} \text{Minimize } & \delta - \sum_{i \in C} P_i^1 x_i + \sum_{i \in C} P_i^2 (w - w_i - x_i) - \\ & \sum_{i \in C} P_i^3 y_i + \sum_{i \in C} P_i^4 (h - h_i - y_i) + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^5 \left[x_z^{\text{bin}} - x_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) - \sigma_{i,z,l} \right] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^6 \left[x_z^{\text{bin}} + w_z^{\text{bin}} - w_i - x_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) - \sigma_{i,z,l} \right] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^7 \left[y_z^{\text{bin}} - y_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) - \varphi_{i,z,l} \right] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^8 \left[y_z^{\text{bin}} + h_z^{\text{bin}} - h_i - y_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) - \varphi_{i,z,l} \right] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^9 (M \widehat{\mu_{i,z,l}} - \sigma_{i,z,l}) - \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{10} (x_i + \sigma_{i,z,l}) + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{11} M (1 - \widehat{\mu_{i,z,l}}) - \sigma_{i,z,l} - x_i + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{12} [-M (1 - \widehat{\mu_{i,z,l}}) - \sigma_{i,z,l} - x_i] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{13} (M \widehat{\mu_{i,z,l}} - \varphi_{i,z,l}) - \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{14} (y_i + \varphi_{i,z,l}) + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{15} [M (1 - \widehat{\mu_{i,z,l}}) - \varphi_{i,z,l} - y_i] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{16} [-M (1 - \widehat{\mu_{i,z,l}}) - \varphi_{i,z,l} - y_i] + \\ & \sum_{i \in C} \sum_{i' \in C_{\text{left}} \setminus \{i\}} P_{i,i'}^{17} [M (\widehat{\alpha_{i,i'}} - 1) - x_{i'} + x_i] + \\ & \sum_{i \in C} \sum_{i' \in C_{\text{left}} \setminus \{i\}} P_{i,i'}^{18} [M (\widehat{\beta_{i,i'}} - 1) - y_{i'} + y_i] + \\ & \sum_{i \in C^u \cup C^d \cup C^t} P_i^{19} \left(\sum_{l \in L} \sum_{j \in J^0} k_{i,j} \widehat{\rho_{i,l,j}} - y_i \right) + \\ & \sum_{i \in C^s} P_i^{20} \left(\sum_{l \in \{1,2\}} \sum_{j \in J^1} k_{i,j} \widehat{\rho_{i,l,j}} - \frac{1}{4} h_i - y_i \right) + \\ & \sum_{i \in C} \sum_{i' \in C_{\text{left}} \setminus \{i\}} P_{i,i'}^{21} [w_i + M (2 - \widehat{\gamma_{i,i'}} - \widehat{\alpha_{i,i'}}) - x_{i'} + x_i] \end{aligned} \quad (67)$$

5.1.2. MP的构建

在求解SP后，我们获得一个可行解，并可以基于对

偶理论推断最优割。我们遵循 Adulyasak 等[37]的思想，将所有场景合并为一个 SP，并在每次迭代中仅添加一个最优割，以避免求解过多 SP。大多数研究首先基于 SP 获得对偶子问题（DSP），然后通过最优割法将 DSP 与 MP 连接起来。由于本研究涉及大量决策变量，并且在对偶过程中 max 函数和求和的合并具有挑战性，因此我们选择拉格朗日松弛来确定最优割。根据拉格朗日松弛展开，我们引入一个新变量 ζ 来表示预期成本。与元器件相对位置对应的 MP 可以重新表述如下：

$$[\text{MP}] \text{ Minimize } \zeta \quad (68)$$

约束条件为

式 (4) ~ (9)，(14) ~ (16)，(18) ~ (20)，(22) ~ (26) 和 (30) ~ (33)；

$$\begin{aligned} \zeta \geq & \sum_{i \in C} P_i^2 (w_i - w_i) + \sum_{i \in C} P_i^4 (h_i - h_i) + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^5 \left[x_z^{\text{bin}} - x_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) + x_z^{\text{bin}} + \right. \\ & \left. w_z^{\text{bin}} - w_i - x_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) \right] + \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^7 \left[y_z^{\text{bin}} - \right. \\ & \left. y_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) + y_z^{\text{bin}} + h^{\text{bin}} - h_i - y_z^{\text{bin}} (1 - \widehat{\mu_{i,z,l}}) \right] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^9 M \mu_{i,z,l} + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{11} M (1 - \mu_{i,z,l}) + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{12} [-M (1 - \mu_{i,z,l})] + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{13} M \mu_{i,z,l} + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{15} M (1 - \mu_{i,z,l}) + \\ & \sum_{i \in C} \sum_{z \in Z} \sum_{l \in L} P_{i,z,l}^{16} [-M (1 - \mu_{i,z,l})] + \\ & \sum_{i \in C} \sum_{i' \in C_{\text{left}(i)}} P_{i,i'}^{17} M (\widehat{\alpha_{i,i'}} - 1) + \\ & \sum_{i \in C} \sum_{i' \in C_{\text{right}(i)}} P_{i,i'}^{18} M (\widehat{\beta_{i,i'}} - 1) + \\ & \sum_{i \in C^a \cup C^d \cup C^c} P_i^{19} \sum_{l \in L} \sum_{j \in J^a} k_{i,j} \widehat{\rho_{i,l,j}} + \\ & \sum_{i \in C^b} P_i^{20} (\sum_{l \in \{1,2\}} \sum_{j \in J^2} k_{i,j} \widehat{\rho_{i,l,j}} - \frac{1}{4} h_i) + \\ & \sum_{i \in C} \sum_{i' \in C_{\text{left}(i)}} P_{i,i'}^{21} [w_i + M (2 - \widehat{\gamma_{i,i'}} - \widehat{\alpha_{i,i'}})] \end{aligned} \quad (69)$$

约束 (68) 表示对所有 MP 解都有效的最优割。

算法的核心在于确保在每次迭代中传递的变量在 SP 和 MP 中都是有效和可行的。具体来说，当 MP 确定元器件的相对位置时，约束相对宽松，而 SP 在可行域内进一步细化元器件的精确位置坐标。因此，MP 用于提供元器件的相对位置，而 SP 确保这些位置满足布局约束，最终提供精确的布局坐标。

在每次迭代中，新产生的最优割是算法进展的关键。通过求解 SP，我们获得一个可行解，并通过对偶理论推断割的优化性。这个最优割为后续迭代提供了关键指导，

确保每一轮在可行域内提高解决方案质量，从而接近最优解。基于我们的算法，MP 和 SP 可以高效协作，使每次迭代都能收缩可行域并逐步优化布局，最终求解元器件的最优坐标。

5.2. 算法框架

总之，BD 迭代地将 MP 中的变量值传递到 SP，然后 SP 将最优割返回到 MP。这种迭代思想逐渐缩小了解空间，最终获得最终解。BD 的伪代码在附录 A 的第 S5 节中展示，框架如下：

步骤 1：初始化。设置上界 $UB = \infty$ ，下界 $LB = -\infty$ ，迭代次数 $I_n = 0$ ，迭代上限 $I_{\text{max}} = 50$ ，可接受间隙 $g = 0.01(w + h)$ 。计算初始指导、SP 和 MP 的目标值，并将它们分别设为 $\mathcal{F}(o)$ 、 $\mathcal{F}(sp)$ 和 $\mathcal{F}(mp)$ 。

步骤 2：初始解设置。根据从高到低的排列顺序和沿芯片区域边缘放置元器件的思想，我们设计了一组初始可行解。由于 $\mathcal{F}(o) > LB$ ，更新 LB ， $\mathcal{F}(o) \leftarrow LB$ 。

步骤 3：求解 SP。将初始指导的相对位置信息代入 SP 以获得元器件的初始可行坐标。如果 $\mathcal{F}(sp) < UB$ ，更新 $\mathcal{F}(sp) \leftarrow UB$ ， $I_n + 1 \leftarrow I_n$ 。

步骤 4：推断本轮迭代的最优割。如果 $UB - LB \leq g$ ，停止迭代；否则，转到步骤 5。

步骤 5：求解 MP。将通过拉格朗日松弛获得的最优割代入 MP 以获得相对位置信息。如果 $\mathcal{F}(mp) > LB$ ，更新 LB ， $\mathcal{F}(mp) \leftarrow LB$ 。如果 $I_n \geq 1$ ，转到步骤 2。

步骤 6：迭代终止。存在以下任何一种情况：① $UB - LB \leq g$ ， $I_n = I_{\text{max}}$ ，算法终止并输出最优解。

5.3. 算法收敛性

在本研究中，我们的算法保证收敛。具体来说，我们在迭代中获得新的最优割，算法最终将收敛到最优解。

原始的最佳解为 (x^*, y^*) ，其中 x 和 y 分别表示整数向量和非负实向量。根据经典 BD 原理，SP 有以下表达式：

$$g(\tilde{y}) = \text{Minimize } c^T x \quad (70)$$

约束条件为

$$Ax = b - B\tilde{y} \quad (71)$$

$$x \geq 0 \quad (72)$$

其中， A 是一个矩阵，代表整数变量线性约束的系统； B 是一个矩阵，对应于涉及实向量 $\tilde{y}$ 的对偶约束中的系数； b 是一个常数向量；向量 c 代表 SP 目标函数中的系数。

由于在第 4.3 节中对目标和约束的处理，SP 的解是单调的。存在一个足够大的 N ，使得对于所有 $n \geq N$ ，SP 的

最优解满足 $\mathbf{c}^T \mathbf{x}_n \leq \mathbf{c}^T \mathbf{x}_{n-1}$ 。这确保了算法的每次迭代都产生一个至少与前一个同样好的解。芯片区域限制了 SP 的可行解数量，从而限制了 SP 的解空间。因此，原始问题的最优解对应于优化 SP 的解。

总之，由于 SP 的决策受芯片区域的限制，SP 的目标函数具有上界，使 SP 成为一个有界优化问题。由于 MP 的约束比 SP 更宽松，并且 MP 和 SP 都受同一芯片区域的限制，因此 MP 也是有界的。由于 MP 和 SP 都是有界的，根据优化理论，原始问题有最优解。如果一个有界优化问题没有最优解，那么该问题的求解空间将是无限的，这与有界的定义相矛盾。综上所述，由于本研究中的 MP 和 SP 都受芯片区域的限制，原始问题的最优解可以通过我们的算法有效确定。

6. 数值实验

在以下实验中，我们使用来自布局竞赛“2014 年国际物理设计研讨会 (ISPD) 详细布线驱动布局”的真实数据来测试所提出的 BD 算法的性能[38]。我们的真实数据也被 Zhu 等[5]、Chen 等[32]和 Chen 等[10]相关研究广泛使用。本研究首先设置了各种规模的实例组，然后通过与 CPLEX 的比较进行一些对比分析，以验证 BD 算法的有效性，并进行一些敏感性分析，以提供布局优化的一些潜在有用的见解。每个实例在一台配备 128 GB RAM 和 3.40 GHz Intel Xeon E5-2643 v3 CPU 的工作站上求解。算法在 Visual Studio 2020 中使用 C# 编程语言实现，CPLEX 版本为 12.6.1。

6.1. 参数设置

为确保实验的有效性，我们使用竞赛数据作为基准，这些数据具有权威性。我们采用 Chen 等[32]的方法来划分各种元器件的数量，以形成一个 MCH 布局实例。在此基础上，我们考虑了内部有导线的特殊元器件，并将其比例设置为双行高度元器件的 30% [5]。由于部分元器件高度的调整，芯片区域可能无法容纳所有元器件，因此应相应调整这些元器件的宽度，以确保元器件的面积保持不变。修改后的实例组在最大限度上保留了基准本身的特征，确保了布局的合理性[7]。实例的特征已在表 3 中列出，以供参考。

6.2. 算法性能

对于小规模实例，我们使用三种方法进行实验：第一种方法是直接使用 CPLEX 求解模型 $M1$ ，这为后两种

表 3 实例数据

Group ID	No. of cells	Proportion of various cells	No. of bins	No. of wires	No. of Vth
ISG1	4	1:1:1:1	2	5	0:4:0
ISG2	10	6:2:1:1	4	10	1:8:1
ISG3	20	12:5:1:2	8	30	2:16:2
ISG4	35	23:7:2:3	12	50	3:28:4
ISG5	55	40:9:2:4	16	65	5:44:6
ISG6	70	46:14:4:6	20	80	6:56:8

The order in proportion of various cells is single-row-height cells: double-row-height cells: triple-row-height cells: special cells. The order in the column of “No. of Vth” is high voltage threshold: standard voltage threshold: low voltage threshold.

方法提供了最优解作为基准；第二种方法是通过删除约束 (2)、约束 (3) 和约束 (12) ~ (16) 并放宽模型 $M1$ 中的二进制变量 $\omega_{z,l}$ 使其连续，获得最优解值的下界；第三种方法是使用所提出的 BD 算法迭代求解 MP 和 SP。这些方法获得的计算时间和目标值用于评估算法的性能。我们将每个实例的最大求解时间设置为 1 h。

根据表 4，BD 算法计算的小规模实例的最小布局成本与 CPLEX 相同。这一比较表明，BD 算法是模型求解的可靠方法，它可以获得布局优化问题的最优解。在 ISG1 和 ISG2 中，BD 算法的求解时间长于 CPLEX，但随着实例组规模的增加，BD 算法在求解速度上的优势变得更加明显。在 ISG3 中，BD 算法能够在 81 s 内给出布局解，而 CPLEX 需要约 15 min。由于放宽了一些变量和约束，LB 获得的结果是 CPLEX 目标值的下界，与最优解的最大差距为 3.52%。虽然 CPLEX 在小规模实例上表现尚可，但随着实例组规模的增加，它无法在有效时间内提供芯片布局解。我们得出结论，BD 算法对于求解大规模实例是有用的，因为该算法在小规模实例上的准确性已得到确认。

为了进一步评估算法在不同规模实例上的性能。我们设计了三组大规模实验，每组包括五个代表性实例。BD 算法、LB 方法和 CPLEX 的详细计算结果总结在表 5 中。值得注意的是，在 ISG4 实验中，虽然 CPLEX 偶尔能够在预定义的时间限制内解决问题，但我们的 BD 算法显示出高的效率，所有求解时间均保持在 270 s 以下。随着实例规模的进一步增加，BD 算法始终在 16 min 内提供可行的布局解，凸显了其卓越的求解速度和适应性。此外，在小规模实验中，我们观察到随着实例复杂性的增加，LB 解与最优解之间的差距扩大。然而，BD 算法与 LB 方法之间的最大差距控制在 13.96% 以内，这不仅验证了 BD 算法在保持解决方案质量方面的稳健性，还间接强调了其在更大规模问题上优于 LB 方法的优势。基于上述分析，我们得出结

表4 算法在小规模实例下的表现

Instances (group and ID)	CPLEX		LB		BD algorithm		
	F_{CPLEX}	t_{CPLEX}	F_{LB}	GAP_1	F_{BD}	t_{BD}	GAP_2
ISG1							
1-1	6.00	0.51	5.98	0.33%	6.00	2.14	0.00%
1-2	6.00	0.49	5.99	0.17%	6.00	1.63	0.00%
1-3	6.00	0.58	5.98	0.17%	6.00	1.84	0.00%
1-4	6.00	0.53	5.97	0.34%	6.00	1.81	0.00%
1-5	6.00	0.34	5.99	0.17%	6.00	1.35	0.00%
Average	—	—	—	0.30%	—	—	0.00%
ISG2							
2-1	21.48	4.35	20.86	2.97%	21.48	11.95	0.00%
2-2	23.16	2.49	132.87	0.65%	23.16	10.14	0.00%
2-3	23.07	2.08	113.84	1.36%	23.07	10.02	0.00%
2-4	22.43	3.63	107.88	2.23%	22.43	11.39	0.00%
2-5	23.38	3.41	121.86	1.08%	23.38	11.25	0.00%
Average	—	—	—	1.66%	—	—	0.00%
ISG3							
3-1	232.08	890.62	226.54	2.65%	232.08	78.34	0.00%
3-2	202.94	828.17	193.52	4.87%	202.94	72.49	0.00%
3-3	219.36	847.65	212.75	4.14%	219.36	73.98	0.00%
3-4	221.57	853.23	213.69	2.65%	221.57	75.62	0.00%
3-5	243.41	901.84	235.18	3.50%	243.41	80.03	0.00%
Average	—	—	—	3.52%	—	—	0.00%

F_{CPLEX} , F_{LB} , and F_{BD} represent the objective value of solution obtained by CPLEX, LB, and the BD algorithm, respectively. t_{CPLEX} and t_{BD} are the calculation time of CPLEX and the BD algorithm in seconds, respectively. $\text{GAP}_1 = (F_{\text{CPLEX}} - F_{\text{LB}})/F_{\text{LB}}$; $\text{GAP}_2 = (F_{\text{BD}} - F_{\text{CPLEX}})/F_{\text{CPLEX}}$.

表5 算法在大规模实例下的表现

Instances (group and ID)	CPLEX		LB		BD algorithm		
	F_{CPLEX}	t_{CPLEX}	F_{LB}	GAP_1	F_{BD}	t_{BD}	GAP_2
ISG4							
4-1	612.21	—	584.20	4.79%	612.21	259.48	0.00%
4-2	567.37	3509.79	523.81	8.32%	567.37	228.95	0.00%
4-3	622.33	—	595.78	4.46%	622.33	268.31	0.00%
4-4	607.85	—	572.92	6.10%	607.85	237.60	0.00%
4-5	522.17	3382.05	485.14	7.63%	522.17	203.20	0.00%
Average	—	—	—	6.26%	—	—	0.00%
ISG5							
5-1	—	—	1084.35	11.09%	1204.56	457.02	—
5-2	—	—	995.47	13.44%	1129.31	410.39	—
5-3	—	—	1126.14	10.88%	1248.63	463.76	—
5-4	—	—	1219.07	9.03%	1329.13	517.13	—
5-5	—	—	1158.62	9.10%	1264.01	470.50	—
Average	—	—	—	10.71%	—	—	—
ISG6							
6-1	—	—	1561.57	15.26%	1799.84	909.60	—
6-2	—	—	1509.65	12.26%	1694.66	840.25	—
6-3	—	—	1420.29	11.23%	1579.72	797.09	—
6-4	—	—	1540.93	14.49%	1764.18	901.55	—
6-5	—	—	1402.21	16.59%	1634.90	832.20	—
Average	—	—	—	13.96%	—	—	—

F_{CPLEX} , F_{LB} , and F_{BD} represent the objective value of solution obtained by CPLEX, LB and the BD algorithm, respectively. t_{CPLEX} and t_{BD} are the calculation time of CPLEX and the BD algorithm in seconds, respectively. $\text{GAP}_1 = (F_{\text{BD}} - F_{\text{LB}})/F_{\text{LB}}$; $\text{GAP}_2 = (F_{\text{BD}} - F_{\text{CPLEX}})/F_{\text{CPLEX}}$. CPLEX cannot solve the instances within 3600 seconds except ISG4-2 and 4-5.

论, 所提出的BD算法在大规模实例中表现出卓越的效率和精度, 为复杂优化问题提供了有效且可靠的解决方案。

6.3. 敏感性分析

在本节中, 我们考察了几个重要参数的影响: 芯片区域的纵横比、不同高度元器件的数量比、元器件总数和bin总数。根据敏感性分析, 我们得出了一些可能有用的设计见解, 以指导芯片布局优化。

6.3.1. 实验设计

本节的敏感性分析涉及表6中列出的指标, 以理解后续实验。我们将从三个方面描述这些指标: 指标对应的主题、属性和具体内容。芯片区域的形状受其纵横比影响, 这也在实验中考虑。元器件可以根据高度进行划分, 实验中变化的是各种元器件之间的比例和数量。芯片区域与bin数量之间的关系取决于切割系数: 切割次数越多, 芯片区域划分越细, bin数量越多。

表6 敏感性实验中的指标

Subject	Property	Concrete content
Chip	Shape of chip region	Proportion of height to width
Cell	Types of height	Single-row-height
		Double-row-height
		Triple-row-height
		Special
Bin	Degree of cutting	The number of bins

我们设计了六组实验来讨论每个指标对布局成本的影响。表7列出了非实验因素、因变量和自变量。非实验因素与自变量相关, 但在每个实例中保持不变。自变量对应表6中的指标, 敏感性实验的目的是探索它们对成本的影响。因变量与本研究的优化目标一致。除了第三组额外考虑了最大空间利用率外, 其他组都关注布局成本。每组实验涵盖了六个不同规模的实例, 可以全面探索变量之间的关系。

表7 敏感性实验考虑的因素和变量

Group	Non-experimental factors	Independent variables	Dependent variables
1	Area of chip region, the total number of cells and the proportion of cells with various heights	Aspect ratio of chip region	Placement cost
2	Total height of cells and combination of cells with various heights	Quantity ratio of cells with varying heights	Placement cost
3	Area of chip region, the types of cells and their respective proportions	Total number of cells	Placement cost and the maximum space utilization rate
4	Area of chip region, the total number of cells and the proportion of cells with various heights	Total number of bins	Placement cost

6.3.2. 芯片区域纵横比的影响

为便于建模, 我们将元器件、bin和芯片区域的形状设置为矩形。元器件的宽度和高度根据其功能和其他属性设置, bin和芯片区域的纵横比($w:h$)与芯片本身相关。虽然芯片的形状是固定的, 但布局过程中bin的确定大多是经验性的, 没有明确的规则。因此, 以下实验探讨了芯片区域纵横比对布局成本的影响, 旨在为bin的划分提供有参考价值的建议。

附录A中的表S1提供了六组实例(G1~G6)的芯片区域宽度和高度的实验数据。为使表格清晰, 宽度和高度分别用 w 和 h 表示。在相同规模的实例组下进行了五次不同纵横比的实验, 以全面探索芯片区域纵横比对布局成本的影响。以G1为例, 如果将芯片区域固定为36, 我们可以获得一个宽和高均为6的正方形区域, 以及($w:h$) = (12:3)、(9:4)、(4:9)和(3:12)的矩形区域。元器件总数随芯片区域的扩大而增加, 各种高度的元器件平均分配以确保实验的有效性。

图7展示了芯片区域纵横比对不同实例组成本的影响。在面积固定的同一组中, 当($w:h$) = (3:12)时, 布局成本最低, 成本随着($w:h$)的减小而增加。当芯片区域的宽度和高度相差较大时, 线性阵列意味着更确定的设计方案。然而, 随着芯片区域的宽度和高度更加接近, 区域更接近正方形, 提供了更多的设计选择。由于元器件的宽度都设置在较小范围内, 高度固定为行高的倍数, 因此元器件宽度的差异小于高度。这使得元器件的垂直排列比水平排列更紧凑, 浪费的空间更少。随着组规模的增加, ($w:h$)较高的芯片区域的成本增加缓慢。

总之, ($w:h$)较高的芯片区域对布局成本更有效。在实际芯片设计中, 建议在布局分区中增加bin的纵横比, 以便布局能够实现更高的质量。

6.3.3. 不同高度元器件数量比的影响

在意识到芯片区域纵横比与布局成本之间的关系后, 我们继续调查成本是否受不同高度元器件数量比的影响。

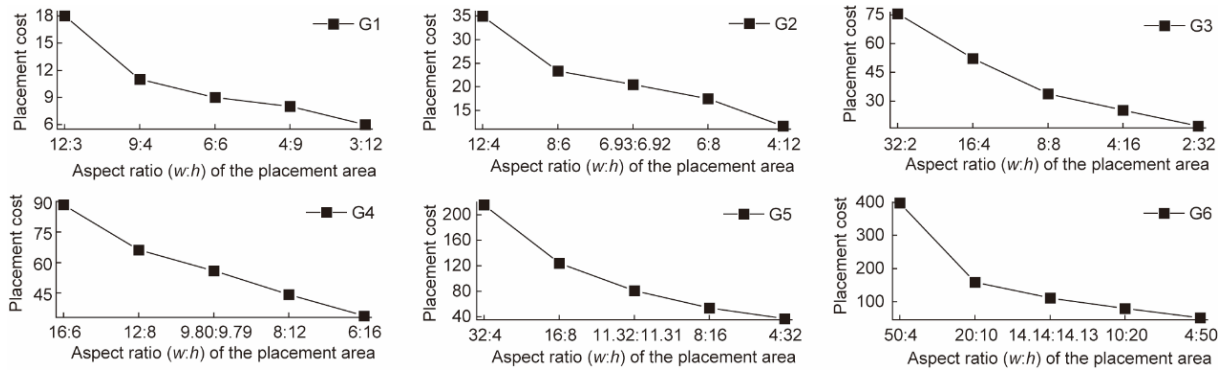


图7. 芯片区域纵横比($w:h$)的敏感性分析。

直接在实验中调整各种元器件的数量是没有意义的,因为这忽略了元器件的总高度,并且不同高度的元器件需要不同的布线方式。

我们设置了六组实例,并在同一组中保持元器件的总高度不变。为保证实验的可行性,确定能够容纳元器件自身的芯片区域面积的下限至关重要。以G1为例,10个元器件的总面积应小于给定的芯片区域。固定元器件的总高度后,可以根据其高度安排元器件,并相应分配各种元器件的数量。每组包括七个实验,具体实验数据如附录A中的表S2所示。

图8展示了不同高度元器件数量比对不同实例组布局成本的影响。在总元器件高度固定的同一组中,当元器件总数最少时,成本最低,成本与元器件总数成正相关。这是由于元器件数量较少意味着三行高度元器件的比例较高,这削弱了其他元器件对布局的影响。当元器件总数适中时,双行高度元器件往往产生最佳的布局质量。高度类型更均匀的元器件意味着元器件之间的兼容性更高,从而提高了芯片区域的利用率。受特殊元器件对齐规则的影响,存在两个半行残余空间,使得结果不够紧凑。

成本与芯片面积和元器件总高度成正比。不同高度元

器件的数量比影响成本,当较高元器件的比例较高时,成本较低。对于芯片设计公司,安排高度类型统一的元器件更具经济意义。控制元器件的高度类型也可以有效提高布局的利用率。

6.3.4. 元器件总数的影响

在芯片区域足够大时,添加更多元器件会增加布局成本。在研究元器件数量与布局成本之间的关系时,固定芯片区域面积的实例是重要的,因为芯片布局的目标是在有限的区域内放置尽可能多的元器件。

对于相同规模的实例,我们保持芯片区域的面积不变。在本实验中,仅考虑单行和双行高度元器件,每种元器件的比例相等,以简化各种元器件的分配。我们确定了芯片区域可以容纳的最大元器件数量,确保每个实验都是合理的。每组中最大规模的元器件总数是给定芯片区域内可以容纳的最高值。关于芯片区域和不同类型元器件数量的实验数据在附录A的表S3中提供。在同一组中,元器件数量均匀增加。为保持表格简洁,两种高度的元器件数量分别用s和d表示。

图9展示了元器件总数对布局成本和最大空间利用率的影响。在芯片区域相同的同一组中,布局成本随着元器件总数的增加而增加。尽管在设置中考虑的实例组是可行的,但仍有一些情况无法求解。这表明考虑现代设计规则的布局比仅考虑排列可行性的布局更复杂,空间利用率更低。尽管元器件总数均匀增加,但成本的上升是不均匀的。

由于本实验包括无法求解的实例,我们还比较了组之间的最大空间利用率。通过将可能容纳元器件的最大面积除以芯片区域的面积,得出最大空间利用率。如图9右上角的子图所示,最大空间利用率随着实例规模的增长呈现快速上升的趋势,在G3达到局部峰值,随后短暂下降后再次上升。这种波动的趋势源于MIA约束与布局面积变化之间的平衡。初始的上升趋势可归因于在较小面积中

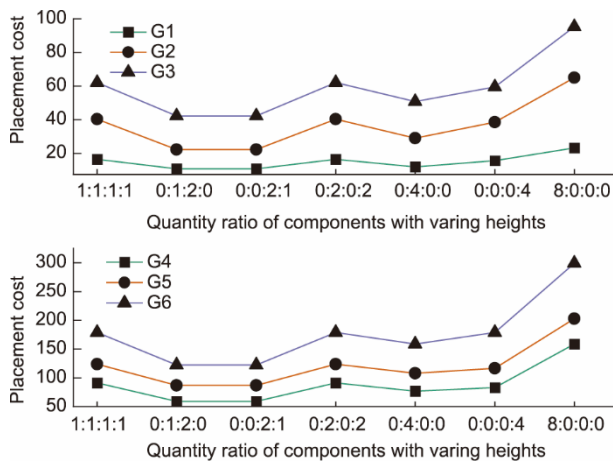


图8. 元器件数量比的敏感性分析。

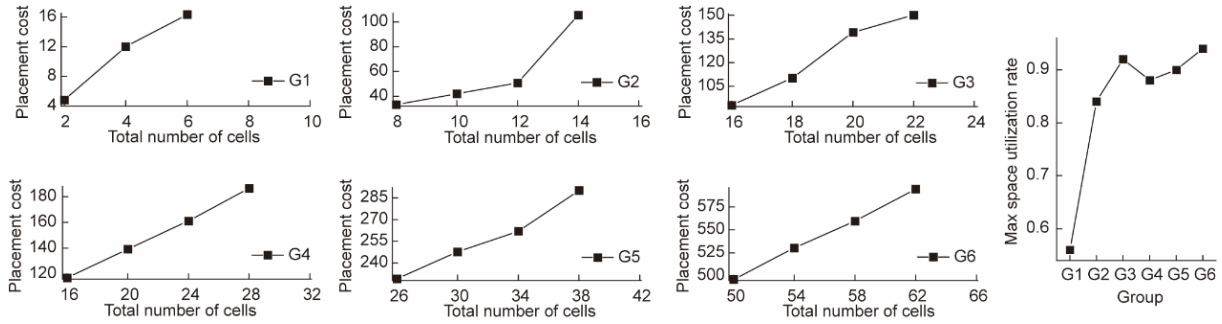


图9. 元器件总数的敏感性分析。

MIA 约束对元器件布局的影响更为显著，这些约束促进了小面积元器件的聚集，使有限空间内能够容纳更多元器件，从而提高空间利用率。然而，随着布局面积的扩大，元器件的布局选择变得更加灵活。这种更大的自由度有利于早期元器件的布局，但使后续元器件的布局变得复杂，导致最大空间利用率暂时下降。对于大规模芯片布局问题，MIA 约束值保持不变，元器件聚集往往在多个局部空间中发生。在这种情况下，布局顺序对自由度的影响逐渐减小，整体空间利用率得以进一步提高。然而，由于元器件之间的不重叠约束，在实例规模的增加超过一定阈值后，空间利用率最终趋于饱和。

总体而言，布局成本随着元器件总数的增加而增加，且增加幅度会有所波动。能够容纳的元器件数量不仅与芯片区域的面积有关，还与设计技术有关。此外，元器件总数与空间利用率之间存在正相关关系，凸显了如何从另一个角度审视布局优化。对于芯片设计公司，通过提高单个元器件的性能并减少元器件总数，可以在保持或提高芯片功能的同时显著节省成本。除了根据功能需求规划元器件总数外，公司还可以通过考虑布局成本、空间利用率以及设计约束与芯片性能之间的平衡来战略性地确定元器件配置。通过采用先进策略和优化布局区域，公司可以实现更高的效率和更低的成本，特别是在大规模芯片设计中，布局约束和资源利用发挥着关键作用。

6.3.5. bin总数的影响

将芯片区域划分为bin的目的是细化芯片区域，以解决元器件与其各种类型匹配的复杂问题。bin的数量与芯片区域的面积成正比，还与元器件的数量和类型有关。bin的划分在布局优化中起着至关重要的作用，其数量可能会影响成本。

为此，我们构建了六组实例（G1~G6），除了第一组（G1）只有四种情况外，其余每组各有五种情况。由于G1中的元器件总数相对较少，第五种情况缺乏数据。每组的初始两个bin均匀分配为一个和两个。G1和G6的bin

划分相当，中间四组（G2~G5）的bin划分相同。元器件的类型均为单行高度，芯片区域的面积和元器件总数在同一组内保持不变，以防止其他因素对布局优化的影响。附录A中的表S4提供了bin划分的全面信息。

图10展示了bin数量对所有实例组布局成本的影响。在芯片区域和元器件总数固定的同一组中，成本随着bin数量的增加而增加，且增加逐渐放缓。这是由于随着bin数量的增加，元器件排列变得不那么紧凑。每个bin的边缘都有空白区域以防止元器件重叠，但也降低了空间利用率。这个结果与第6.3.3节中的发现相似，即减少bin内元器件的数量可以节省成本。由于芯片区域是固定的，bin的划分也有限，空白区域存在上限，因此成本的增加率也呈现放缓趋势。

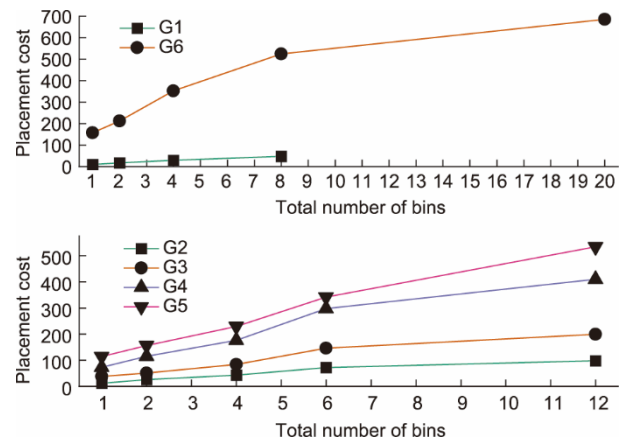


图10. bin总数的敏感性分析。

总体而言，布局成本与bin数量成正相关。对于芯片设计公司，减少bin的划分可以有效节省成本。为了满足客户需求并保持竞争优势，建议公司掌握bin划分的技术并将其与对齐技术相结合，这对公司的业务发展和市场地位至关重要。

6.3.6. 芯片设计公司成本优化的管理见解

基于所提供的内容，以下是一些管理见解。

(1) 纵横比优化：管理者在规划布局时应考虑设计空间的纵横比。优化的纵横比（如3：12）可以降低成本并提升设计灵活性。在宽度和高度之间取得平衡可实现更高效的空间利用。

(2) 元器件数量和成本管理：减少设计中的元器件总数可以显著降低布局成本。使用较少、较高的元器件（如三行高度元器件）可提高整体布局效率。这表明管理层应专注于优化元器件高度和数量，以增强兼容性和空间利用。

(3) 复杂性和可行性：随着元器件数量的增加，布局成本不成比例地增加，凸显了需要仔细规划以避免阻碍有效空间利用的复杂性。管理者应在元器件数量与设计可行性之间取得平衡，以保持效率并最小化成本。

(4) bin管理：bin的数量会影响成本和空间利用率。管理者应优化bin划分以减少降低利用率的过多间距。减少每个bin中的元器件数量也可以是控制成本的策略。了解bin大小和排列之间的权衡可以实现更好的资源分配和成本管理。

通过应用这些见解，管理者可以做出更明智的决策，提高效率、降低成本并改善芯片的整体设计效果。

7. 结语

我们研究了在芯片制造中使用一些现代布局技术的背景下，芯片布局的优化问题，并提出了一种支持元器件分配、bin划分、bin与元器件类型匹配以及元器件合法化决策过程的方法论。本文构建的模型可以满足芯片布局工程师对各种功能元器件的设计要求，并从节省成本和提高空间利用率的角度优化设计。我们提出了一种综合优化模型，考虑了设计细节，设计了一种精确算法以提供合理的解决方案，并基于敏感性分析的结果得出了一些潜在有用的管理见解。以下是本文的核心成就总结。

从问题建模的角度来看，我们制定了一个MILP模型用于布局，并处理了布局决策中的复杂非线性部分。在开发模型时，我们旨在最小化布局成本，同时满足不同类型元器件的实际设计要求。除了满足二维布局不重叠的基本需求外，模型还纳入了技术设计中的三个实际约束。本文可能是第一个在芯片制造中使用一些现代电路设计技术的背景下，为布局问题提出数学规划模型的研究。

从算法设计的角度来看，我们设计并实现了一种基于Benders分解的精确算法，在求解阶段将上述芯片布局问题分解为MP和SP。MP中的决策涉及元器件和bin之间的位置关系、按类型分类的元器件数量以及非法元器件的

合并。SP使用从MP传输的数据来确定元器件布局坐标。此外，我们使用对偶理论获得最优割，实现了基于MP和SP之间交互的迭代求解过程。解决方案的最优性通过数值实验得到了验证。

从设计见解的角度来看，我们利用布局优化的真实数据并进行敏感性分析，得出了芯片区域形状和元器件类型对布局设计成本的影响等潜在有用的设计见解。基于实验结论，我们进一步分析了元器件性能改进的方向以及对齐技术对区域划分的影响，提供了实用的设计见解。

本研究也存在局限性。本研究中提出的模型仅使用了部分已发布的现代设计技术，未使用一些高科技企业开发的最新技术。如果熟悉最前沿相关技术的相关从业者能够提炼一些新技术并进一步改进本研究提出的模型，我们的研究将更加深入。以上都可以是未来可能的研究方向。

CRedit authorship contribution statement

Binqi Zhang: Writing-original draft, Software, Investigation. **Lu Zhen:** Writing-original draft, Supervision, Methodology, Funding acquisition, Conceptualization. **Gilbert Laporte:** Writing-review & editing, Supervision, Project administration, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

致谢

本研究得到了国家自然科学基金(72025103、72394360、72394362、72361137001)以及上海市科学技术委员会项目(23JC1402200)的支持。

Appendix A. Supplementary data

Supplementary data to this article can be found online at <https://doi.org/10.1016/j.eng.2025.03.020>.

References

- [1] Anderson J, Kayraklioglu E, Imani H, Shen C, Miscuglio M, Sorger V, et al. Virtualizing a post-Moore's Law analog mesh processor: the case of a photonic PDE accelerator. *ACM Trans Embed Comput Syst* 2023;22(2):38.
- [2] Wang J, Wong A, Lam E. Standard cell layout with regular contact placement. *IEEE Trans Semicond Manuf* 2004;17(3):375–83.
- [3] Zhu Z, Chen J, Peng Z, Zhu W, Chang Y. Generalized augmented lagrangian and its applications to VLSI global placement. In: *Proceedings of the 55th ACM/ESDA/IEEE Design Automation Conference*; 2018 Jun 24 – 28; FranciscoSan, CA, USA. New York City: IEEE Press; 2018. p. 1–6.
- [4] Wang C, Wu Y, Chen J, Chang Y, Kuo S, Zhu W, et al. An effective legalization algorithm for mixed-cell-height standard cells. In: *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference*; 2017 Jan 16–17; Tokyo, Japan. New York City: IEEE; 2017. p. 450–5.
- [5] Zhu Z, Huang Z, Yang P, Zhu W, Chen J, Zhou H. Mixed-cell-height legalization considering complex minimum width constraints and half-row fragmentation effect. *Integr VLSI J* 2020;71:1–10.
- [6] Wu G, Chu C. Detailed placement algorithm for VLSI design with double-row height standard cells. *IEEE Trans Comput Aided Des Integrated Circ Syst* 2016; 35(9):1569–73.
- [7] Chow J, Pui W, Young F. Legalization algorithm for multiple-row height standard cell design. In: *Proceedings of the 53rd Annual Design Automation Conference*; 2016 Jun 5–9; Austin, TX, USA. New York City: IEEE; 2016. p. 1–6.
- [8] Spindler P, Schlichtmann U, Johannes F. Abacus: fast legalization of standard cell circuits with minimal movement. In: *Proceedings of the 2008 ACM International Symposium on Physical Design*; 2008 Apr 13–16; Portland, OR, USA. New York City: Association for Computing Machinery (ACM); 2008. p. 47–53.
- [9] Lin Y, Yu B, Xu X, Gao J, Viswanathan N, Liu W, et al. MrDP: multiple-row detailed placement of heterogeneous-sized cells for advanced nodes. *IEEE Trans Comput Aided Des Integrated Circ Syst* 2018;37(6):1237–50.
- [10] Chen J, Lin Z, Xie Y, Zhu W, Chang Y. Mixed-cell-height placement with complex minimum-implant-area constraints. *IEEE Trans Comput Aided Des Integrated Circ Syst* 2022;41(11):4639–52.
- [11] Breuer M. A class of min-cut placement algorithms. In: *Proceedings of the 14th Design Automation Conference*; 1977 Jun 20–22; OrleansNew, LA, USA. New York City: Association for Computing Machinery (ACM); 1977. p. 284–90.
- [12] Wang M, Yang X, Sarrafzadeh M. DRAGON2000: standard-cell placement tool for large industry circuits. In: *Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design*; 2000 Nov 5–9; JoseSan, CA, USA. New York City: IEEE; 2000. p. 260–3.
- [13] Agnihotri A, Ono S, Madden P. Recursive bisection placement: Feng Shui 5.0 implementation details. In: *Proceedings of the 2005 International Symposium on Physical Design*; 2005 Apr 3–6; San Francisco, CA, USA. New York City: Association for Computing Machinery (ACM); 2005. p. 230–2.
- [14] Viswanathan N, Alpert C, Sze C, Li Z, Wei Y. The DAC 2012 routability-driven placement contest and benchmark suite. In: *Proceedings of the 2012 49th ACM/EDAC/IEEE Automation Conference*; 2012 Jun 3–7; FranciscoSan, CA, USA. New York City: Association for Computing Machinery (ACM); 2012. p. 774–82.
- [15] Yutsis V, Bustany I, Chinnery D, Shinnerl J, Liu W. ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement. In: *Proceedings of the 2014 International Symposium on Physical Design*; 2014 Mar 30–Apr 2; Petaluma, CA, USA. New York City: Association for Computing Machinery (ACM); 2014. p. 161–8.
- [16] Kim M, Hu J, Li J, Viswanathan N. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. In: *Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design*; 2015 Nov 2–6; FranciscoSan, CA, USA. New York City: Association for Computing Machinery (ACM); 2015. p. 921–6.
- [17] Hao R, Cai Y, Zhou Q. Intelligent and kernelized placement: a survey. *Integr VLSI J* 2022;86:44–50.
- [18] Lu J, Chen P, Chang C, Lu S, Huang D, Teng C, et al. ePlace: electrostatics-based placement using fast Fourier transform and Nesterov's method. *ACM Trans Des Autom Electron Syst* 2015;20(2):1–34.
- [19] Lin Y, Jiang Z, Gu J, Li W, Dhar S, Ren H, et al. Dreamplace: deep learning toolkit-enabled GPU acceleration for modern VLSI placement. *IEEE Trans Comput Aided Des Integrated Circ Syst* 2021;40(4):748–61.
- [20] Yue S, Songhori E, Jiang J, Boyd T, Goldie A, Mirhoseini A, et al. Scalability and generalization of circuit training for chip floorplanning. In: *Proceedings of the ISPD 2022: International Symposium on Physical Design*; 2022 Mar 27–30; Seoul, Republic of Korea. New York City: Association for Computing Machinery (ACM); 2022. p. 65–70.
- [21] Silva E, Melega G, Akartunali K, de Araujo S. Formulations and theoretical analysis of the one-dimensional multi-period cutting stock problem with setup cost. *Eur J Oper Res* 2023;304(2):443–60.
- [22] Pierini L, Poldi K. Optimization of the cutting process integrated to the lot sizing in multi-plant paper production industries. *Comput Oper Res* 2023; 153: 106157.
- [23] Coutinho J, Reis M, Neves D, Bernardo F. Robust optimization and data-driven modeling of tissue paper packing considering cargo deformation. *Comput Ind Eng* 2023;175:108898.
- [24] Cai S, Deng J, Lee L, Chew E, Li H. Heuristics for the two-dimensional irregular bin packing problem with limited rotations. *Comput Oper Res* 2023; 160:106398.
- [25] Martinovic J, Selch M. Mathematical models and approximate solution approaches for the stochastic bin packing problem. *Comput Oper Res* 2021;135: 105439.
- [26] Delorme M, Iori M, Martello S. Bin packing and cutting stock problems: mathematical models and exact algorithms. *Eur J Oper Res* 2016;255(1):1–20.
- [27] Serairi M, Haouari M. A theoretical and experimental study of fast lower bounds for the two-dimensional bin packing problem. *Oper Res* 2018; 52(2): 391–414.
- [28] Queiroz T, Hokama P, Schouery R, Miyazawa F. Two-dimensional disjunctively constrained knapsack problem: heuristic and exact approaches. *Comput Ind Eng* 2017;105:313–28.
- [29] Tseng K, Chang Y, Liu C. Minimum-implant-area-aware detailed placement with spacing constraints. In: *Proceedings of the 2016 ACM/EDAC/IEEE Design Automation Conference*; 2016 Jun 5–9; Austin, TX, USA. New York City: Association for Computing Machinery (ACM); 2016. p. 1–6.
- [30] Farghadan A, Mohammadzadeh N. Quantum circuit physical design flow for 2D nearest-neighbor architectures: quantum circuit design flow. *Int J Circuit Theory Appl* 2017;45(7):989–1000.
- [31] Li W, Lin Y, Li M, Dhar S, Pan D. UTPlaceF 2.0: a high-performance clock-aware FPGA placement engine. *ACM Trans Des Autom Electron Syst* 2018; 23(4):1–23.
- [32] Chen J, Chang Y, Wu Y. Mixed-cell-height detailed placement considering complex minimum-implant-area constraints. *IEEE Trans Comput Aided Des Integrated Circ Syst* 2021;40(10):2128–41.
- [33] Lu J, Zhuang H, Chen P, Chang H, Chang C, Wong Y, et al. EPlace-MS: electrostatics-based placement for mixed-size circuits. *IEEE Trans Comput Aided Des Integrated Circ Syst* 2015;34(5):685–98.
- [34] Lin Z, Xie Y, Zou P, Wang S, Yu J, Chen J. An incremental placement flow for advanced FPGAs with timing awareness. *IEEE Trans Comput Aided Des Integrated Circ Syst* 2022;41(9):3092–103.
- [35] Belieres S, Hewitt M, Jozefowicz N, Semet F. Meta partial benders decomposition for the logistics service network design problem. *Eur J Oper Res* 2022;300(2):473–89.
- [36] Zhang Q, Wang SA, Zhen L. Yard truck retrofitting and deployment for hazardous material transportation in green ports. *Ann Oper Res* 2022;17:103322.
- [37] Adulyasak Y, Cordeau J, Jans R. Benders decomposition for production routing under demand uncertainty. *Oper Res* 2015;63(4):851–67.
- [38] International Symposium on Physical Design (ISPD). ISPD 2014 detailed routing-driven placement contest [Internet]. Petaluma: International Symposium on Physical Design 2014; 2014 Mar 16 [cited 2025 Mar 3]. Available from: https://www.ispd.cc/contests/14/ispd2014_contest.html.