计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(4): 978−988, 2025

流路径驱动的微流控生物芯片任意角度布线算法

潘友林[1,2,3] 郭 帅[1,2,3] 黄 兴[4] 刘耿耿[1,2,3]

[1](福州大学计算机与大数据学院 福州 350116)

[2](大数据智能教育部工程研究中心(福州大学) 福州 350116)

[3](福建省网络计算与智能信息处理重点实验室(福州大学) 福州 350116)

[4](西北工业大学计算机学院 西安 710072)

(panyoulin@163.com)

Any-Angle Routing Algorithm for Microfluidic Biochips Driven by Flow Path

Pan Youlin[1,2,3], Guo Shuai[1,2,3], Huang Xing[4], and Liu Genggeng[1,2,3]

[1](*College of Computer and Data Science*, *Fuzhou University*, *Fuzhou* 350116)

[2](*Engineering Research Center of Big Data Intelligence* (*Fuzhou University*), *Ministry of Education*, *Fuzhou* 350116)

[3](*Fujian Key Laboratory of Network Computing and Intelligent Information Processing* (*Fuzhou University*), *Fuzhou* 350116)

[4](*School of Computer Science*, *Northwestern Polytechnical University*, *Xi'an* 710072)

Abstract Continuous-flow microfluidic biochips (CFMBs) have become a hot research topic in recent years due to their ability to perform biochemical assays automatically and efficiently. For the first time, PathDriver+ takes the requirements of the actual fluid transportation into account in the design process of CFMBs and implements the actual fluid transport and removal, and plans separate flow paths for each transport task, which have been neglected in previous work. However, PathDriver+ does not take full advantage of the flexibility of CFMBs routing because it only considers the optimization of flow channel length for the global routing in the mesh model, except for the detailed routing. In addition, PathDriver+ only considers the X architecture, while the existing work shows that the any-angle routing can utilize the routing resources more efficiently and shorten the flow channel length. To address the above issues, we propose a flow path-driven arbitrary angle routing algorithm, which can improve the utilization of routing resources and reduce the flow channel length while considering the actual fluid transportation requirements. The proposed algorithm constructs a search graph based on constrained Delaunay triangulation to improve the search efficiency of routing solutions while ensuring the routing quality. Then, a Dijkstra-based flow path routing method is used on the constructed search graph to generate a routing result with a short channel length quickly. In addition, in the routing process, channel reuse strategy and intersection optimization strategy are proposed for the flow path reuse and intersection number optimization problems, respectively, to further improve the quality of routing results. The experimental results show that compared with the latest work PathDriver+, the length of channels, the number of ports used, and the number of channel intersections are significantly reduced by 33.21%, 11.04%, and 44.79%, respectively, and the channel reuse rate is improved by 26.88% on average, and the total number of valves introduced at intersections is reduced by 42.01% on average, which demonstrates the effectiveness of the algorithm in this paper.

Key words continuous-flow microfluidic biochips; computer-aided design; physical design; any-angle routing; flow-path planning

收稿日期: 2024-01-29; **修回日期**: 2024-08-12

基金项目: 国家自然科学基金项目(62372109); 福建省杰出青年科学基金项目(2023J06017)

This work was supported by the National Natural Science Foundation of China (62372109) and the Fujian Science Foundation for Distinguished Young Scholars (2023J06017).

通信作者: 刘耿耿(liugenggeng@fzu.edu.cn)

摘　要　连续微流控生物芯片（continuous-flow microfluidic biochips, CFMBs）由于其能够自动高效地执行生化应用，成为近年来的研究热点. PathDriver+将实际的流体运输需求考虑进 CFMBs 设计流程中，并实现了实际的流体运输和去除，并为每个运输任务规划独立的流路径，而这些问题在之前的工作中被忽略了. 但是，由于 PathDriver+仅考虑了网格模型下总体布线的线长优化，而未考虑详细布线，没有充分利用 CFMBs 布线的灵活性. 此外，PathDriver+仅考虑 X 型布线方式，而任意角度布线能够更有效地利用布线资源，从而缩短流通道长度. 针对上述问题，提出了流路径驱动的任意角度布线算法，在考虑实际的流体运输需求的同时，提高布线资源的利用率，减少流通道的长度. 首先基于 Delaunay 三角剖分构建搜索图，从而在保证布线质量的同时，提高布线解的搜索效率. 然后，在构建的搜索图上，使用基于 Dijkstra 的流路径布线方法，以快速生成具有较短线长的布线结果. 在布线过程中针对流通道复用和流通道交叉点数量优化问题，分别提出了通道复用策略和交叉优化策略，以进一步提高布线结果的质量. 实验结果表明，与最新工作 PathDriver+相比，所提算法在布线总线长、流层端口使用数量、通道交叉点数量方面分别降低了 33.21%，11.04%，44.79%，通道复用率平均提高了 26.88 个百分点，交叉点处引入阀门的总数量平均减少了 42.01%，这表明所提算法的有效性和优越性.

关键词　连续微流控生物芯片；计算机辅助设计；物理设计；任意角度布线；流路径规划

中图法分类号　TP391.72

DOI: 10.7544/issn1000-1239.202440058　　**CSTR:** 32373.14.issn1000-1239.202440058

随着物联网[1]和生物化学技术的迅猛发展，连续微流控生物芯片（continuous-flow microfluidic biochips, CFMBs）作为生物化学和信息技术等跨学科问题之间的交叉研究，引起了人们的高度关注[2]. CFMBs 通过精确地操作和控制芯片通道内部的流体样品，以实现生化应用的自动化、小型化和集成化. CFMBs 具有许多优点，例如，样品用量小、反应速度快、检测灵敏度高、实验过程自动化等. 与传统实验方法相比，CFMBs 具有更高的实验效率和可重复性，也能够缩短实验时间和降低实验成本. 由于 CFMBs 具有更高的灵敏度、更小的尺寸和更低的成本，近年来 CFMBs 已被开发用于各种生化测定[3]，如癌症[4]和艾滋病检测[5]. 此外，CFMBs 在蛋白质结晶[6]、免疫测定[7]、DNA 合成[8]等领域都有相应的研究和应用.

CFMBs 通常由 2 层聚二甲基硅氧烷（polydimethylsiloxane, PDMS）材料构成，分别被称为流层和控制层. 利用多层软光刻（multilayer soft lithography, MSL）技术可以在流层和控制层上蚀刻出对应的通道网络[9]. 图 1 为 CFMBs 的结构示意图. 流层中的通道被称为流通道，它通过流层端口与外界相连，用于运输样品和试剂. 而流层端口分为流端口和废液端口. 控制层的通道被称为控制通道，控制通道连接到控制端口，用于传导空气压力. 此外，在控制通道和流通道的重叠区域中形成用作阀门的柔性膜，该柔性膜由从控制端口注入的空气压力控制. 当接受到外部压力时，阀门对应的控制通道向下挤压下方流通道，由此实现阀门的关闭. 而当外部压力撤销时，阀门对应的控制通道恢复原状，下方流通道不再阻塞，由此实现阀门的打开. 以阀门为基本控制单元，可以构建复杂的微流控组件，如多路复用器和混合器[10]，并且可以通过预先定制的分析计划自动执行生化应用[11].

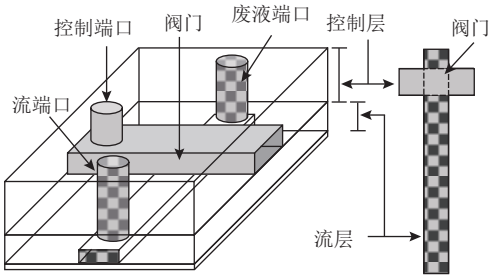


Fig. 1　Structure diagram of CFMBs

图 1　CFMBs 的结构图

为了在 CFMBs 上执行生化应用，样品和试剂需要在组件间运输. 但是，每个运输任务在执行期间都需要一个独占的流路径. 具体而言，为了驱动流体流动，应在路径的起始处设置与外部压力源连接的流端口来提供推力. 此外，由于通道或组件的初始状态不是真空状态，而是充满空气，为了防止液体被空气阻隔，需要在通道的末端连接一个废液端口，用于排出通道中的空气. 因此，流端口、组件、废液端口以及连接它们的流通道构成了运输任务的完整流路径. 以图 2 为例，将存储在 r_1 容器中的试剂输送到混合器时，有效的流路径应为：$f_1 \to r_1 \to c_1 \to$混合器$\to c_2 \to w_1$.

CFMBs 的设计流程通常分为高层次综合阶段和

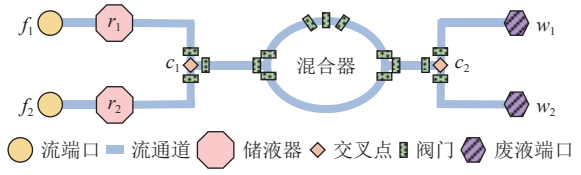


Fig. 2 Schematic diagram of the flow path

图 2 流路径示意图

物理设计阶段. 高层次综合阶段的目标是找到一个绑定方案, 使得每个操作都绑定到一个特定的组件上执行, 然后生成一个调度方案, 使得所有的操作都满足给定的依赖关系, 同时最小化生化应用的完成时间. 在此阶段需要考虑组件之间的流体体积管理以及多余液体、废液的移除等. 而在物理设计阶段中, 需要将所分配的组件放置到芯片上的确切位置, 同时在它们之间建立有效的连接, 构建完整的流路径.

国内外研究学者针对 CFMBs 的架构设计提出了多种方法. 文献 [12] 提出了一种自顶向下的合成方法来生成优化的 CFMBs 架构, 同时最大限度地减少阀门切换的总量. 文献 [13] 使用基于序列对的方法来研究 CFMBs 的物理设计. 而在文献 [14] 中, 物理设计被描述为 SAT 问题, 以生成近似最优解. 文献 [15] 提出了严格约束端口数量下生成 CFMBs 架构的设计流程. 文献 [16] 提出了一种时间感知流通道布线方法来优化 CFMBs 的时序行为. 此外, 文献 [17] 提出了一种考虑清洗优化和通道存储的设计流程, 以生成具有高存储效率的芯片架构. 然而, 这些方法没有考虑运输任务所需的完整流路径, 因而无法满足在 CFMBs 上执行实际流体运输的需求. 为了克服这一缺陷, 文献 [18] 提出了一种路径驱动的 CFMBs 设计流程 PathDriver+, 以将实际流体运输整合到芯片设计流程中. 但 PathDriver+仅考虑了网格模型下总体布线的线长优化, 而未考虑详细布线, 没有充分利用 CFMBs 布线的灵活性.

在 CFMBs 的物理设计阶段中, 布线是一个非常重要的环节, 影响到芯片的性能和成本. 布线的设计需要考虑到 CFMBs 的设计约束、流通道交叉点的优化等问题. 现有的 CFMBs 自动布线算法, 基本上都采用曼哈顿布线方式, 也就是水平垂直布线方式. 文献 [19] 提出了一种避障直角 Steiner 最小树算法以解决 CFMBs 中的流通道布线问题. 完全可编程阀阵列是一种特殊的 CFMBs, 文献 [20] 针对该类型芯片提出了一种流通道动态构造算法. 文献 [21] 提出了一种控制通道布线算法以生成满足长度匹配约束的布线结果. 然而, 文献 [19−21] 研究都没有考虑 CFMBs 的任意角度布线. 随着制造工艺的进步, 人们已经可以在 CFMBs 上蚀刻出任意角度的通道. 基于上述背景, 文献 [22] 提出了第 1 个任意角度布线算法, 以此最大程度缩短流通道的长度. 但该布线算法在布线过程中没有考虑通道复用, 增加了冗余的通道长度和制造成本. 此外, 该工作没有考虑构建完整的流路径, 这使得其生成的布线结果难以满足生化应用中实际的流体运输需求, 降低了芯片的可靠性.

为了克服上述问题, 本文提出了流路径驱动的 CFMBs 任意角度布线算法, 从而更有效地利用布线资源, 提高流路径构建的灵活性, 并利用通道复用策略和交叉优化策略降低 CFMBs 的制造成本. 本文主要贡献有 4 点:

1) 提出了一种流路径驱动的 CFMBs 任意角度布线算法. 该算法将完整流路径的构建考虑到任意角度布线过程中, 以满足实际流体运输的需求.

2) 实现任意角度布线, 提高了布线资源的利用率, 减少流通道长度. 并提出了一种通道复用策略, 通过流通道复用进一步减少流通道长度, 从而降低 CFMBs 的制造成本.

3) 提出了一种交叉优化策略, 在布线过程中尽可能避免流通道交叉, 进而减少阀门数量, 从而降低 CFMBs 的制造成本.

4) 实验结果表明, 本文算法与最新同类型工作相比, 在流通道长度、流通道交叉点数量、流层端口使用数量等方面取得了更好的结果.

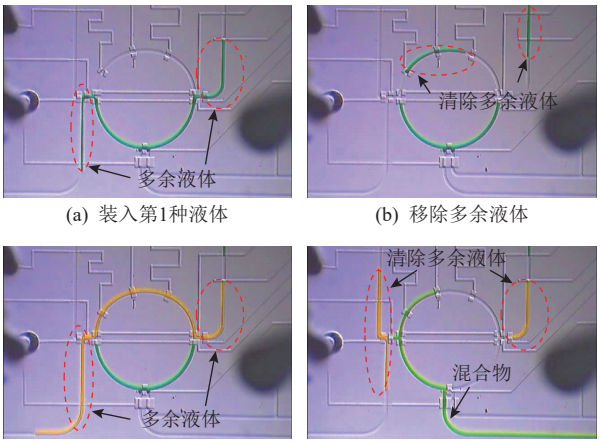
1 问题描述

1.1 设计约束

在 CFMBs 上执行生化应用需要精确控制样品和试剂在组件间的运输. 为了满足这一需求, 每个运输任务在执行期间都需要独占一条完整的流路径, 包括流端口、组件、废液端口以及连接它们的流通道. 此外, 在执行生化应用的过程中还会产生一些不再需要的液体. 例如, 装入后的留在通道中的多余液体、完成生化应用后组件中存在的废弃液体, 这都需要通过独立的流路径来排出, 如图 3 所示.

为了满足生化应用的实际流体运输需求, 在设计芯片时就要规划大量的不同用途的流路径. 主要包括 3 种流路径:

1)流体运输路径. 主要完成在组件间的流体运输. 对应的流通道应为流端口→源组件→目标组件→废液端口.



(a) 装入第1种液体 (b) 移除多余液体

(c) 装入第2种液体并混合 (d) 移除混合物和多余液体

Fig. 3 Snapshots of the mixing operation using CFMBs[18]

图 3 使用 CFMBs 进行混合操作的快照[18]

2)多余液体移除. 在经过流体运输任务后，组件两端中会存在一些多余的液体，需要及时排出. 对应的流通道为流端口→缓存位置→废液端口.

3)废液移除和外部输入. 把生化应用完成后产生的废液及时排出以及将液体从外界输入. 对应的流通道为流端口→目标组件→废液端口.

这些流路径被集成在一个有限的区域内，形成了一个高度集成的芯片架构. 为了从根本上提高 CFMBs 的性能，同时降低相应的制造成本，在设计 CFMBs 时要尽可能地减少通道长度，并减少交叉点和端口[18]. 此外，在 CFMBs 的物理设计阶段，布线方式直接影响通道的长度. 现有的制造工艺允许在 CFMBs 上蚀刻任意角度的通道，该技术能够提高 CFMBs 在生化应用中的性能.

1.2 问题模型

综上所述，流路径驱动的 CFMBs 物理设计任意角度布线问题描述如下：

1)输入芯片的尺寸；各个组件和端口的相关信息，包括位置和尺寸；需要布线的流路径.

2)输出满足设计规则的任意角度布线方案.

3)目标为在构建完整流路径的同时，最大限度地降低整体芯片成本，包括流通道的总长度、芯片中流层端口使用数量、流通道的交叉点数量等.

2 算法设计

2.1 算法流程

本文提出的流路径驱动的任意角度布线算法在输入待布线芯片的相关信息后，就可以输出满足设计规则的任意角度布线方案. 所生成的布线方案不仅构建了完整的流路径，而且通过通道复用策略和交叉优化策略，最大限度地降低了芯片的成本. 图 4 显示了本文算法的总体流程.

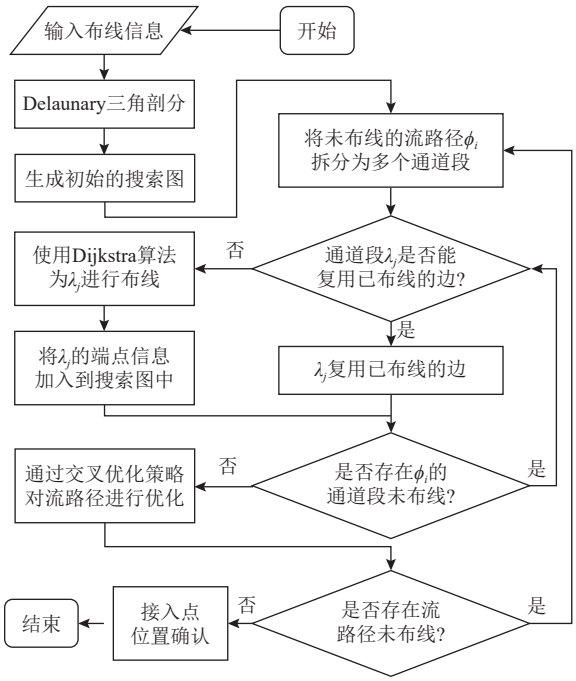


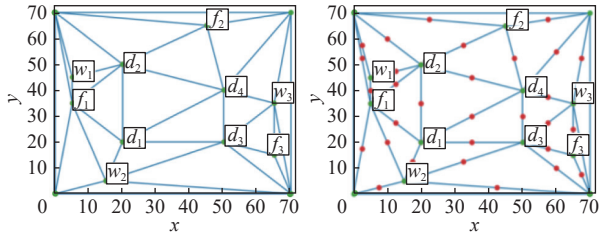
Fig. 4 Flow chart of the any-angle routing algorithm driven by the flow path

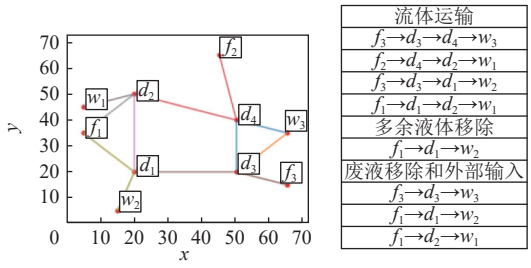
图 4 流路径驱动的任意角度布线算法流程图

首先，根据输入的芯片尺寸以及各个组件和端口的相关信息，对布线区域执行 Delaunay 三角剖分(constrained Delaunay triangulation, CDT). 之后，遍历 CDT 图中的所有三角形，找到每个三角形的每条边的中点，将它们连接起来构成初始的搜索图(search graph, SG). 然后基于生成的搜索图，对于每一条待布线的流路径完成以下过程：1)将流路径拆分为多个通道段；2)对于每个通道段，根据通道复用策略判断是否可复用已布线的边，若有可复用的边则复用该边，若没有可复用的边则将源端和目标端所在三角形的中线加入到初始的搜索图中，并通过 Dijkstra 最短路径算法在搜索图中找到一组最短路径；3)通过交叉优化策略对流路径进行优化；4)重复步骤 2)和步骤 3)直到所有的流路径完成布线. 最后，确定每一条流路径与组件或端口的接入点位置即可结束算法.

图 5 展示了本文算法在上述步骤中生成的结果. 布线区域大小为70×70，$f_1 \sim f_3$代表流端口，$w_1 \sim w_3$代表废液端口，$d_1 \sim d_4$代表组件. 图 5(a)为 Delaunay 三角剖分后生成的 CDT 图. 将图 5(b)中三角形边的中点互相连接即可生成相应的搜索图. 其中，三角形边的中点对应搜索图的顶点. 需注意的是，在芯片边

界上的点被排除在外. 此外, 一个完整的搜索图还必须添加布线源端和目标端以及以它们为顶点的三角形的中线作为额外的点和边. 图 5(c)左侧为通过在搜索图上执行 Dijkstra 最短路径算法得到的最终布线结果, 右侧为对应的 3 种流路径.





(c) 最终的布线结果图以及对应的流路径

Fig. 5 Results generated during the routing process

图 5 布线过程生成的结果

2.1.1 Delaunay 三角剖分

为了能够实现任意角度布线, 需要一个有效的数据结构来表示 CFMBs 的布线区域, 包括布线边界和布线通道. 根据上述要求, 该数据结构必须具有以下属性: 1)只能使用 1 组线段来构造; 2)具有指导布线路径探索的能力; 3)布线通道与芯片边界必须区分. 为了满足这些需求, 采用 Delaunay 三角剖分[23]生成的 CDT 图结构来表示布线区域.

基于给定的平面点集进行操作, 三角剖分能够生成对应的三角形集合, 如图 6 所示. 具体而言, 给定一个平面点集 $P=\{p_1,p_2,\cdots,p_n\}$, 三角剖分的目标是得到一个三角形集合 $T=\{t_1,t_2,\cdots,t_m\}$, 并满足: 1)所有的三角形的顶点恰好组成集合 P; 2)任意 2 个三角形的边不相交; 3)所有三角形的合集构成 P 的凸包[24].

Delaunay 三角剖分是一种带约束的三角剖分, 它

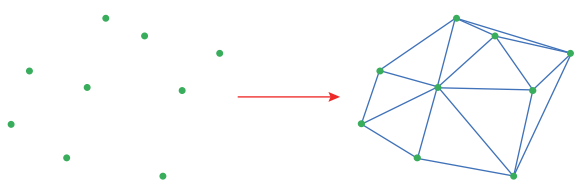
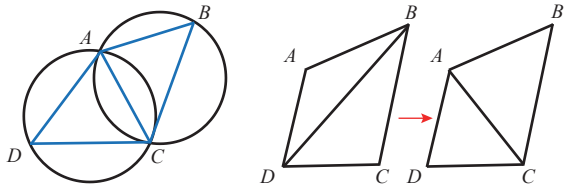


Fig. 6 Schematic diagram of triangulation

图 6 三角剖分示意图

有 2 个重要的特性: 1)空圆特性. 它使得平面点集 P 中没有点处于 T 中任意一个三角形外接圆的内部. 2)最大化最小角特性. 在平面点集 P 对应的三角剖分方案中, Delaunay 三角剖分所形成的三角形的最小角最大. 空圆特性与最大化最小角特性分别如图 7(a)与图 7(b)所示.



(a) 空圆特性 (b) 最大化最小角特性

Fig. 7 Properties of triangulation

图 7 三角剖分性质

本文生成 CDT 图的过程如算法 1 所示.

算法 1. CDT 图生成算法.

输入: 端口, 组件以及边界对应的顶点集合 P;

输出: CDT 图.

① 生成空集合 S_1;

② 取出 P 中任意 3 个不共线的点构成初始三角形 t_0 并放入 S_1 中;

③ for p_i in P

④ 生成空集合 S_2;

⑤ for t_i in S_1

⑥ if p_i 在 t_j 的外接圆上

⑦ 将 t_j 的 3 条边加入到 S_2;

⑧ 将 t_j 从 S_1 中移除;

⑨ end if

⑩ end for

⑪ 移除 S_2 中重复的边;

⑫ for e_k in S_2

⑬ 将 e_k 的 2 个端点连接到 p_i 上, 获得新三角形 t';

⑭ if t' 与 S_1 中的三角形满足 Delaunay 性质

⑮ 将 t' 加入到 S_1;

⑯ end if

⑰ end for

⑱ end for

⑲ 输出 S_1.

2.1.2 构造搜索图

在路径搜索阶段, 采用 Dijkstra 最短路径算法求解布线路径. 为了实现 Dijkstra 最短路径算法, 文献 [22]定义了搜索图. 搜索图是由待布线线网的源端与目标端、CDT 图三角形边的中点、源端和目标端对应

的三角形的中点相互连接组成的无向图. 具体的搜索图的构造过程为: 1)搜索图的顶点是由待布线线网的源端和目标端, 以及除设计边界外的所有三角形边的所有中点组成; 2)三角形的每条中线都对应1条搜索图的边, 除非违反设计规则, 或者中线的3个端点中至少有1个端点不是搜索图的顶点; 3)当且仅当三角形的中线的顶点与源端或者目标终端相对应时, 应将三角形的中线添加到搜索图中.

为了在搜索图上找到最短路径, 由于路径的端点是三角形的顶点, 因此需要附加几条边. 直观地说, 三角形的中线可以作为额外的边. 因此, 当寻找一对给定源端和目标端之间的通道时, 可以利用Dijkstra最短路径算法, 找到搜索图上任意一对源端和目标端之间的最短路径. 生成搜索图的过程如算法2所示.

算法 2. SG图生成算法.

输入: CDT图;

输出: 搜索图SG.

① $SG=\varnothing$;

② 将CDT图中所有不在芯片边界上的边的中点加入到SG中;

③ 将源端和目标端加入到SG中;

④ for p_i in SG

⑤ if p_i位于CDT图的三角形t_j

⑥ 将p_i与t_j的中点相连, 并将生成的边加入到SG;

⑦ end if

⑧ end for

⑨ 输出SG.

2.1.3 基于Dijkstra的流路径布线方法

为了高效地构建生化应用所需的流路径, 本文提出了基于Dijkstra的流路径布线方法. Dijkstra算法是求解单源最短路径的经典算法, 它可以求解从起点到其他点的最短路径. 它的基本思想是从起点出发, 依次扩展到离起点最近的顶点, 然后更新其相邻节点的距离, 直到所有顶点都加入到确定的最短路径集合中.

具体来说, Dijkstra算法维护2个集合: 一个是已经找到最短路径的点的集合; 另一个是还没有找到最短路径的点的集合. 该算法从起点开始, 将起点加入到已确定的最短路径集合中, 然后遍历所有与起点相邻的节点, 更新它们的距离值. 这个过程一直持续到所有顶点都被加入到已确定最短路径的集合中. 在更新节点的距离值时, Dijkstra算法通过比较当前距离值与新的距离值的大小来决定是否更新距离值和前驱节点.

在2.1.2节中, 得到了一个搜索图, 接下来就需要在搜索图中使用Dijkstra算法来求出源端到目标端的最短路径. 在布线开始前, 先将待布线的流路径拆分为多个通道段. 在给定搜索图和待布线的通道段的源端和目标端后, 基于Dijkstra的流路径布线方法过程如算法3所示.

算法 3. 基于Dijkstra的流路径布线方法.

输入: 搜索图SG, 源端p_s, 目标端p_t;

输出: p_s到p_t的最短路径.

① $Q=\varnothing$;

② for p_i in SG

③ 加入到Q;

④ $dist(p_i)=\infty$; /*$dist(p_i)$表示p_i到源点p_s的距离*/

⑤ $prev(p_i)=\varnothing$; /*$prev(p_i)$表示p_i的前驱节点*/

⑥ end for

⑦ $dist(p_s)=0$, 并从Q中删除p_s;

⑧ while $Q\neq\varnothing$

⑨ 从Q取出与p_s最近的点p_i;

⑩ for p_j in 与p_i相邻点的集合

⑪ $dist(p_i,p_j)=dist(p_i)+len(p_i,p_j)$; /*计算从$p_s$经过$p_i$到达$p_j$的路径长度, 其中$len(p_i,p_j)$表示从$p_i$到$p_j$的边长度. */

⑫ if $dist(p_i,p_j)<dist(p_j)$

⑬ $dist(p_j)=dist(p_i,p_j)$;

⑭ $prev(p_j)=p_i$;

⑮ end if

⑯ end for

⑰ end while

⑱ 根据$prev$的值输出到目标端p_t的最短路径.

2.1.4 接入点

在CFMBs设计过程中, 组件或端口都被建模为矩形, 但在寻找流路径时是按照组件或者端口的中心点坐标来计算的. 所以当确定好布线路径后, 为了得到更确切的布线结果, 还需要计算出每条流路径与组件或者端口的接入点坐标. 根据2.1.3节, 已经得到了一组连续线段构成的流路径, 要计算每条路径与组件或者端口边缘的交点, 就要考虑线段与矩形的位置关系. 根据提供的组件或端口的中心点坐标和尺寸, 就可以得出每个组件4个顶点的坐标. 然后, 计算线段与矩形区域的交点, 如图8所示即为4种不同的情况下选择的接入点.

本文确定接入点过程的算法如算法4所示.

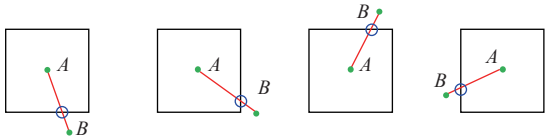


Fig. 8 Pins selected for different scenarios

图 8 不同情况下选择的接入点

算法 4. 接入点确定算法.

输入: 通过 Dijkstra 算法得到的一组边 E, 每个组件或端口的中心点坐标以及长度和宽度;

输出: 流路径和设备或端口的接入点坐标.

① for 任意组件或端口对应的中心点 p_i

② if E 中存在以 p_i 为端点的边

③ 记 p_i 的坐标为 (x_i,y_i), p_i 对应的组件长度和宽度分别为 l_i 和 w_i, 另一个端点记为 p_j;

④ if $x_i == x_j$

⑤ if $y_i < y_j$

⑥ 接入点为 $(x_i, y_i + w_i/2)$;

⑦ else

⑧ 接入点为 $(x_i, y_i - w_i/2)$;

⑨ end if

⑩ else

⑪ $k = (y_j - y_i)/(x_j - x_i)$;

⑫ $b = y_i - k \times x_i$;

⑬ 由式(1)确定接入点;

⑭ end if

⑮ end if

⑯ end for

式(1)如下所示:

$$loc_i = \begin{cases} \left(x_i - \dfrac{l_i}{2}/2,\ k\left(x_i - \dfrac{l_i}{2}\right) + b\right), \text{if } y_i - \dfrac{w_i}{2} \leqslant \\ k\left(x_i - \dfrac{l_i}{2}\right) + b \leqslant y_i + \dfrac{w_i}{2}, \\ \left(x_i + \dfrac{l_i}{2}/2,\ k\left(x_i + \dfrac{l_i}{2}\right) + b\right), \text{if } y_i - \dfrac{w_i}{2} \leqslant \\ k\left(x_i + \dfrac{l_i}{2}\right) + b \leqslant y_i + \dfrac{w_i}{2}, \\ \left(\left(y_i - \dfrac{w_i}{2} - b\right)/k,\ y_i - w_i\right), \text{if } x_i - \dfrac{l_i}{2} \leqslant \\ \left(y_i - \dfrac{w_i}{2} - b\right)/k \leqslant x_i + \dfrac{l_i}{2}, \\ \left(\left(y_i + \dfrac{w_i}{2} - b\right)/k,\ y_i + w_i\right), \text{if } x_i - \dfrac{l_i}{2} \leqslant \\ \left(y_i + \dfrac{w_i}{2} - b\right)/k \leqslant x_i + \dfrac{l_i}{2}. \end{cases} \tag{1}$$

2.2 优化策略

2.2.1 通道复用

为了能够尽可能地减少通道的总长度, 进而降低芯片的成本, 在本文算法中还考虑了通道复用策略, 其主要思想是记录在布线过程中已经布线的通道段, 并在后续布线过程中尽可能复用这些通道段. 在布线开始前, 首先建立一个空图, 记为复用图. 在布线过程中, 首先判断待布线的通道段是否在复用图中, 若存在, 则直接复用该边, 若不存在, 则通过 Dijkstra 算法得到的最短边, 并且将这条新的边加入到复用的图中. 上述通道复用过程如算法 5 所示.

算法 5. 通道复用过程.

输入: 流路径拆分生成的通道段的集合;

输出: 考虑通道复用的布线结果.

① 建立一个空的复用图 G;

② for 每条待布线的通道段 ϕ_i

③ if 待布线的通道段 ϕ_i 能够复用 G 中的通道

④ 复用该通道;

⑤ else

⑥ 利用基于 Dijkstra 的流路径布线方法为 ϕ_i 布线, 并将其加入到 G 中;

⑦ end if

⑧ end for

例如, 如图 5(c)所示, 第 1 条要布线的流路径为 $f_1 \to d_1 \to d_3 \to w_2$, 第 2 条要布线的路径为 $f_1 \to d_1 \to d_4 \to w_1$, 那么在对第 2 条流路径布线时, 如果检查 $f_1 \to d_1$ 可以直接复用, 而后续的 $d_1 \to d_4 \to w_1$ 不可复用, 就通过 Dijkstra 算法求出最短路径, 并加入到复用图中.

2.2.2 交叉优化

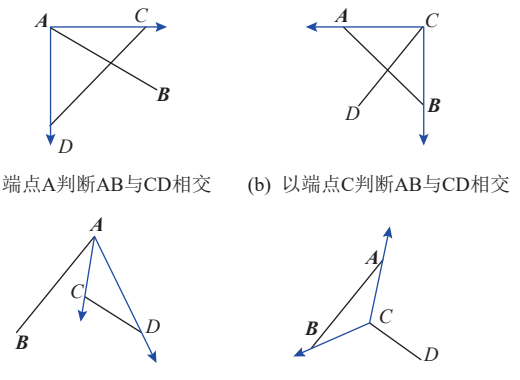
由于布线区域的限制, 多条流路径在布线时不可避免会出现交叉的情况. 为了保证流体在运输过程中不会出现交叉污染, 在交叉点需要引入阀门来控制流向. CFMBs 上的交叉点数量的增加导致需要引入的阀门数量也会增加, 进而增加了芯片设计的复杂度以及芯片的制造成本, 因此需要尽可能地避免交叉点的出现. 可以将一条完整的流路径视为一系列连续的通道段. 因此要判断 2 条流路径中的边是否存在交叉, 需要逐一判断这 2 条流路径对应的通道段是否存在交叉, 这增加了交叉点判断的复杂度. 为了快速判断任意 2 个通道段是否存在交叉点, 本文提出了基于排斥测试和跨立测试的交叉点判断方法, 并基于该方法进一步提出了交叉优化策略以减少交叉点数量.

判断通道交叉过程一共分为 2 步: 第 1 步称为排斥测试. 该步判断 2 条通道段在任意坐标轴的投影是否相交, 若在任意坐标轴上的投影没有相交, 则 2 条

通道段不可能相交. 第 2 步称为跨立测试. 对于 2 条线段, 若每条线段的 2 个端点分别在另一线段所在直线的两侧, 则称这 2 条线段相互跨立, 可利用向量积判断 2 条通道段是否相互跨立. 下面具体阐述这 2 个步骤:

1)排斥测试. 若 2 条通道段在横坐标轴和纵坐标轴的投影都不重合, 则 2 条通道段不存在相交. 以横坐标轴举例, 若 2 条通道段中任意一条通道段的 2 个端点中最大的横坐标大于另一条通道段的 2 个端点中最小的横坐标, 则 2 条通道段的投影在横坐标轴存在重合, 否则, 2 条通道段的投影在横坐标轴不存在重合, 2 条通道段不存在交叉. 2 条通道段则需通过跨立测试判断是否存在交叉.

2)跨立测试. 若排斥测试中 2 条通道段的投影在横坐标轴存在重合, 则需要通过跨立测试判断是否存在交叉点. 假设 2 个向量 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$, 记 $|\boldsymbol{X},\boldsymbol{Y}|$ 为 $\boldsymbol{X}$ 与 $\boldsymbol{Y}$ 的向量积. 如图 9 所示, 以判断 $\boldsymbol{AB}$ 跨立 CD 举例, 记 $\boldsymbol{AB}$ 的 2 个端点构成的向量为 $\boldsymbol{AB}$, 在 $\boldsymbol{AB}$ 的 2 个端点中任意选 1 个端点, 将它与 CD 的 2 个端点相连得到 2 个向量 $\boldsymbol{L}_1$ 和 $\boldsymbol{L}_2$, 若此时向量 $\boldsymbol{AB}$ 在向量 $\boldsymbol{L}_1$ 和 $\boldsymbol{L}_2$ 的中间或 $\boldsymbol{L}_1$ 和 $\boldsymbol{L}_2$ 的边上, 则 $\boldsymbol{AB}$ 跨立 CD; 若此时向量 $\boldsymbol{AB}$ 不在向量 $\boldsymbol{L}_1$ 和 $\boldsymbol{L}_2$ 的中间, 则 $\boldsymbol{AB}$ 没有跨立 CD. 当 2 条通道段相互跨立时就认为这 2 条通道段出现通道交叉.



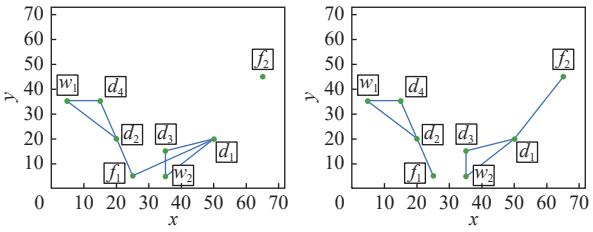
(a) 以端点A判断AB与CD相交 (b) 以端点C判断AB与CD相交

(c) 以端点A判断AB与CD不相交 (d) 以端点C判断AB与CD不相交

Fig. 9 Schematic diagram of straddle test

图 9 跨立测试示意图

在布线过程中, 根据上述方法可以判断出是否会出现通道交叉, 若没有交叉则直接布线. 若出现交叉, 则需要遍历所有可替代路径, 直到找到具有更少交叉点数量的替代路径; 若无法找到则选择原路径. 本文通过设定一个长度阈值来筛选路径, 以平衡交叉点数量与通道长度, 超过该阈值的路径将不作为可选的替代路径. 例如, 图 10(a)为未经交叉优化策略处理的流路径, 其中 d_3 到 w_2 和 f_1 到 d_1 的 2 段流通道出现了一个交叉点. 为了尽可能地优化交叉点数量, 需要替换其中一条通道段. 若选择用 d_3 到 w_1 的通道段替换 d_3 到 w_2 的通道段, 则通道长度增加, 且 d_3 到 w_1 和 d_4 到 d_2 的 2 条通道段还是会出现交叉点. 若选择用 f_2 到 d_1 的通道段替换 f_1 到 d_1 的通道段, 则通道长度没有增加, 且不会出现交叉, 如图 10(b)所示.



(a) 未使用减少交叉策略 (b) 使用减少交叉策略

Fig. 10 Example of reducing channel intersections

图 10 减少通道交叉示例

3 实验分析

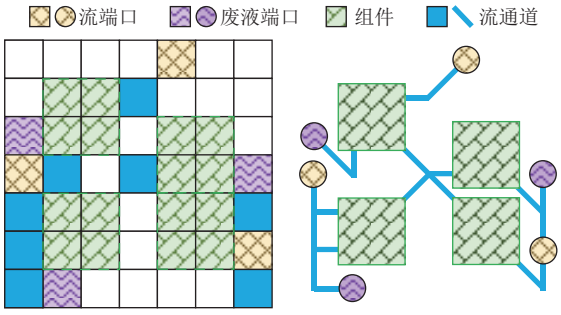
使用 Python 实现了所提出的流路径驱动 CFMBs 任意角度布线算法, 并在具有 2.40 GHz CPU 和 8 GB 内存的个人计算机上进行了测试. 本文共使用了 8 个测试用例[18], 其中聚合酶链反应(PCR)、体外诊断(IVD)、蛋白质分离(ProteinSplit)对应实际的生化应用, 而 Synthetic1, Synthetic2, Synthetic3, Synthetic4, Synthetic5 是 5 个人工合成测试用例. 这 8 个测试用例的详细信息如表 1 所示.

Table 1 Details of Test Cases Used in the Experiments

表 1 实验使用的测试用例信息

测试用例	操作数量	流路径数量	提供的组件和端口数量
PCR	7	17	10
IVD	12	18	8
ProteinSplit	14	25	8
Synthetic1	10	22	12
Synthetic2	15	24	9
Synthetic3	20	24	8
Synthetic4	25	38	12
Synthetic5	30	45	14

为了验证本文算法的有效性, 将本文算法与流路径驱动的最新工作 PathDriver+进行对比. PathDriver+基于虚拟网格模型实现总体布线, 而未考虑虚拟网格内部流通道的详细布线. 图 11(a)展示了 PathDriver+生成的总体布线结果. 为了保证对比的公平性, 将 PathDriver+生成的布线结果按顺序连接端口网格、通



(a) PathDriver+布线结果 (b) 转换后的布线结果

Fig. 11 Schematic diagram of PathDriver+ routing result transformation

图 11 PathDriver+的布线结果转换示意图

道网格及组件网格的中心以获得对应的详细布线结果, 如图 11(b)所示.

为了验证本文算法的有效性, 首先进行 Pathdriver+和本文算法的布线结果在流层端口使用数量、流通道交叉点数量和流通道长度的对比. 表 2、表 3 和表 4 分别展示了 2 种算法在流层端口使用数量、流通道交叉点数量和流通道长度的对比结果.

Table 2 Comparison of Our Algorithm and PathDriver+[18] in the Number of Used Fluidic Ports

表 2 本文算法与 PathDriver+[18] 在流层端口使用数量的对比

测试用例	PathDriver+[18]	本文算法	改进比例/%
PCR	6	6	0.00
IVD	4	3	25.00
ProteinSplit	4	4	0.00
Synthetic1	8	7	12.50
Synthetic2	4	4	0.00
Synthetic3	4	4	0.00
Synthetic4	7	5	28.57
Synthetic5	9	7	22.22
平均			11.04

Table 3 Comparison of Our Algorithm and PathDriver+[18] in the Number of Intersections

表 3 本文算法与 PathDriver+[18] 在交叉点数量的对比

测试用例	PathDriver+[18]	本文算法	改进比例/%
PCR	4	0	100.00
IVD	2	1	50.00
ProteinSplit	2	1	50.00
Synthetic1	5	0	100.00
Synthetic2	3	3	0.00
Synthetic3	2	2	0.00
Synthetic4	4	3	25.00
Synthetic5	3	2	33.33
平均			44.79

Table 4 Comparison of Our Algorithm and PathDriver+[18] in the Total Channel Length

表 4 本文算法与 PathDriver+[18] 在流通道总长度的对比

测试用例	流通道总长度/mm		改进比例/%
	PathDriver+[18]	本文算法	
PCR	179.85	108.41	39.72
IVD	117.43	47.18	59.82
ProteinSplit	137.78	108.39	21.33
Synthetic1	229.25	114.23	50.17
Synthetic2	159.85	144.61	9.53
Synthetic3	144.32	109.91	23.84
Synthetic4	227.35	147.10	35.30
Synthetic5	187.35	138.63	26.00
平均			33.21

不同于 PathDriver+基于虚拟网格模型实现 X 型布线, 本文算法基于 Delaunay 三角剖分图实现任意角度布线, 这为流路径的端口提供更多连接方式, 增加了流层端口复用的概率. 因此, 虽然 PathDriver+是通过整数线性规划实现布线的, 在 X 型布线结果中流层端口使用数量方面已经接近最优解, 但是与 PathDriver+相比, 本文提出的任意角度布线算法依旧在流层端口使用数量上取得了 11.04% 的优化. 此外, 得益于任意角度布线以及本文提出的交叉优化策略和通道复用策略, 与 PathDriver+相比, 本文算法在流通道交叉点数量和流通道长度上分别取得了 44.79% 和 33.21% 的优化. 这表明本文算法能够在考虑流体实际运输需求的情况下, 生成高质量的布线结果.

此外, 本文中还考虑了通道复用率这一指标, 在不引入冲突的情况下, 流通道之间是可以共享的. 提高通道复用率可以降低通道的总长度, 分别计算了 PathDriver+与本文算法的通道复用率, 其结果如图 12 所示. 与 PathDriver+对比, 本文算法的通道复用率平均增加了 4.68 个百分点, 平均增长率为 26.88 个百分点.

由于需要在通道交叉处放置 2~4 个阀门, 以引导流体流向正确的方向, 同时避免运输任务之间的冲突. 阀门的引入会增加芯片设计的成本, 也会提高芯片设计的复杂性. 因此交叉点处引入的阀门数量也是一个重要的评价指标. 图 13 展示了微流控芯片中引入阀门数量的对比结果. 通过计算分别得到 PathDriver+与本文算法引入的阀门数量, 由于在本文算法中, PCR 和 Synthetic1 的布线均无交叉出现, 所以阀门数量为 0. 与 PathDriver+对比, 本文算法引入

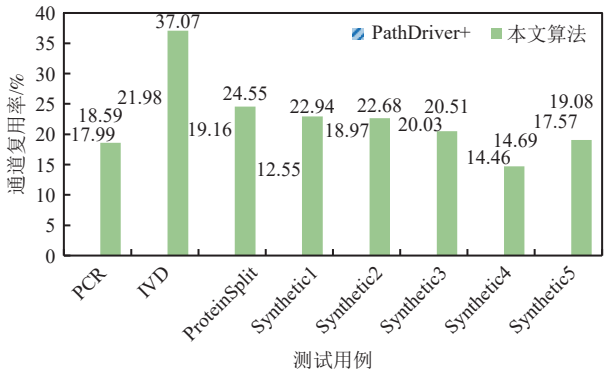


Fig. 12 Comparison of our algorithm and PathDriver+[18] in the channel reuse rate

图 12 本文算法与 PathDriver+[18] 在通道复用率的对比

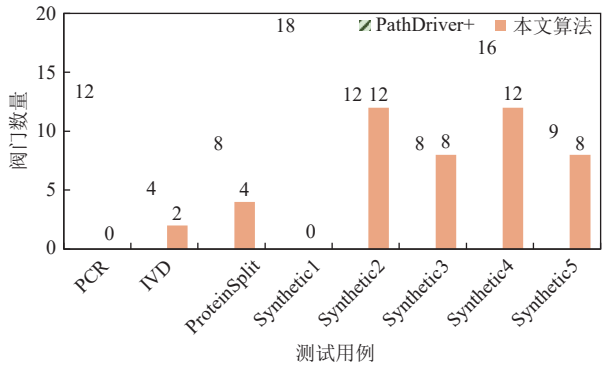


Fig. 13 Comparison of our algorithm and PathDriver+[18] in the number of valves

图 13 本文算法与 PathDriver+[18] 在阀门数量的对比

阀门的总数量平均减少了 42.01%, 这降低了芯片控制系统的复杂性和制造成本.

PathDriver+将 X 型布线问题公式转化为整数线性规划问题, 并由求解器求解. 由于求解整数线性规划问题的复杂性, 因此本文限制求解器的运行时间为 900 s. 表 5 展示了 PathDriver+与本文算法在 CPU 时间方面的对比. 在所有测试用例上, 本文算法均能够在 11 s 内获得布线结果, 这表明本文算法能够在较短时间内生成高质量的布线结果.

Table 5 Comparison of our Algorithm and Pathdriver+[18] in the CPU Time

表 5 本文算法与PathDriver+[18]在CPU时间的对比

s

测试用例	PathDriver+[18]	本文算法
PCR	900.38	8.64
IVD	900.74	7.52
ProteinSplit	900.24	8.69
Synthetic1	900.20	9.92
Synthetic2	900.90	7.95
Synthetic3	900.48	8.77
Synthetic4	900.39	9.94
Synthetic5	900.96	10.81

4 结 论

针对现有工作未能同时考虑实际流体运输需求和任意角度布线, 本文提出了一种流路径驱动的 CFMBs 任意角度布线算法. 本文算法通过 Delaunay 三角剖分图对布线区域进行划分, 并通过基于 Dijkstra 的流路径布线方法构建完整流路径, 使得生成的 CFMBs 布线结果满足生化应用的流体运输需求. 此外, 本文还提出了通道复用策略和交叉优化策略以降低 CFMBs 的制造成本. 通道复用策略通过优先选择已存在的流通道, 从而提高通道复用率, 减少布线的总长度, 降低了 CFMBs 的制造成本. 交叉优化策略通过减少交叉点数量, 从而减少了阀门的数量, 进一步降低了芯片设计架构的复杂性和芯片制造的成本. 与最新工作 PathDriver+的实验结果对比, 本文算法中引入阀门的数量平均减少了 42.01%, 通道的复用率提高了 26.88 个百分点, 并且在流层端口使用数量、交叉点数量和流通道总长度方面分别取得了 11.04%, 44.79%, 33.21% 的优化, 这表明了本文算法的有效性和优越性.

作者贡献声明: 潘友林负责提出初步方案、论文撰写和修改; 郭帅负责算法设计和实验验证; 黄兴负责论文思路构建、理论指导和论文修改; 刘耿耿负责论文方案分析、理论指导、论文润色和修改.

参 考 文 献

[1] Zhang Xiaodong, Zhang Chaokun, Zhao Jijun. State-of-the-art survey on edge intelligence[J]. Journal of Computer Research and Development, 2023, 60(12): 2749−2769 (in Chinese)
(张晓东, 张朝昆, 赵继军. 边缘智能研究进展[J]. 计算机研究与发展, 2023, 60(12): 2749−2769)

[2] Liu Genggeng, Liu Yufan, Pan Youlin, et al. Three-stage rapid physical design algorithm for continuous-flow microfluidic biochips considering actual fluid manipulations[J]. Electronics, 2024, 13(2): 332

[3] Hu Xu, Chen Zhen, Chen Zhisheng, et al. Architectural synthesis of continuous-flow microfluidic biochips with connection pair optimization[J]. Electronics, 2024, 13(2): 247

[4] Weigum S E, Floriano P N, Redding S W, et al. Nano-bio-chip sensor platform for examination of oral exfoliative cytology[J]. Cancer Prevention Research, 2010, 3(4): 518−528

[5] Watkins N N, Hassan U, Damhorst G, et al. Microfluidic CD4+ and

CD8+ T lymphocyte counters for point-of-care HIV diagnostics using whole blood[J]. Science Translational Medicine, 2013, 5(214): 214ra170

[6] Anderson M J, Hansen C L, Quake S R. Phase knowledge enables rational screens for protein crystallization[J]. Proceedings of the National Academy of Sciences, 2006, 103(45): 16746−16751

[7] Kartalov E P, Zhong J F, Scherer A, et al. High-throughput multi-antigen microfluidic fluorescence immunoassays[J]. BioTechniques, 2006, 40(1): 85−90

[8] Huang Yanyi, Castrataro P, Lee C, et al. Solvent resistant microfluidic DNA synthesizer[J]. Lab on a Chip, 2007, 7(1): 24−26

[9] Wang Qin. Research on control-fluidic physical codesign algorithm for microfluidic biochips[D]. Beijing: Tsinghua University, 2018 (in Chinese)
(王钦. 微流控生物芯片流体与控制协同物理设计算法研究[D]. 北京: 清华大学, 2018)

[10] Thorsen T, Maerkl S J, Quake S R. Microfluidic large-scale integration[J]. Science, 2002, 298(5593): 580−584

[11] Chen Zhisheng, Huang Xing, Guo Wenzhong, et al. Physical synthesis of flow-based microfluidic biochips considering distributed channel storage[C]//Proc of the 22nd Design, Automation & Test in Europe Conf & Exhibition. Piscataway, NJ: IEEE, 2019: 1525−1530

[12] Tseng K, You S, Liou J, et al. A top-down synthesis methodology for flow-based microfluidic biochips considering valve-switching minimization[C]//Proc of the 22nd Int Symp on Physical Design. New York: ACM, 2013: 123−129

[13] Wang Qin, Zou Hou, Yao Hailong, et al. Physical co-design of flow and control layers for flow-based microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, 37(6): 1157−1170

[14] Grimmer A, Wang Qin, Yao Hailong, et al. Close-to-optimal placement and routing for continuous-flow microfluidic biochips[C]// Proc of the 22nd Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2017: 530−535

[15] Huang Xing, Ho T, Guo Wenzhong, et al. MiniControl: Synthesis of continuous-flow microfluidics with strictly constrained control ports[C/OL]//Proc of the 56th Design Automation Conf. New York: ACM, 2019[2024-06-14]. https://ieeexplore.ieee.org/document/8806859

[16] Huang Xing, Ho T, Chakrabarty K, et al. Timing-driven flow-channel network construction for continuous-flow microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(6): 1314−1327

[17] Huang Xing, Guo Wenzhong, Chen Zhisheng, et al. Flow-based microfluidic biochips with distributed channel storage: Synthesis, physical design, and wash optimization[J]. IEEE Transactions on Computers, 2021, 71(2): 464−478

[18] Huang Xing, Pan Youlin, Zhang G L, et al. PathDriver+: Enhanced path-driven architecture design for flow-based microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(7): 2185−2198

[19] Lin C, Liu C, Chen I, et al. An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips[C/OL]//Proc of the 51st Design Automation Conf. New York: ACM, 2014[2024-06-14]. https://ieeexplore.ieee.org/document/6881468

[20] Tseng T, Li Bing, Ho T, et al. Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping[C/OL]//Proc of the 52nd Design Automation Conf. New York: ACM, 2015[2024-06-14]. https://ieeexplore.ieee.org/document/7167327

[21] Yao Hailong, Ho T, Cai Yici. PACOR: Practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips[C/OL]//Proc of the 52nd Design Automation Conf. New York: ACM, 2015[2024-06-14]. https://ieeexplore.ieee.org/document/7167328

[22] Yang Kailin, Yao Hailong, Ho T, et al. AARF: Any-angle routing for flow-based microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(12): 3042−3055

[23] Chew L P. Constrained delaunay triangulations[C]//Proc of the 3rd Annual Symp on Computational Geometry. New York: ACM, 1987: 215–222

[24] Kong Dehui, Ma Chunling. An algorithm for constructing the convex hull and the triangulation of a set of nodes in a plane[J]. Journal of Computer Research and Development, 2000, 37(7): 891−896 (in Chinese)
(孔德慧, 马春玲. 一种平面点集凸包与三角网格综合生成的算法[J]. 计算机研究与发展, 2000, 37(7): 891−896)

Pan Youlin, born in 1996. PhD candidate. His main research interest includes design automation of microfluidic biochips.

潘友林, 1996 年生. 博士研究生. 主要研究方向为微流控生物芯片的设计自动化.

Guo Shuai, born in 1999. Master candidate. His main research interest includes design automation of microfluidic biochips.

郭　帅, 1999 年生. 硕士研究生. 主要研究方向为微流控生物芯片的设计自动化.

Huang Xing, born in 1991. PhD, professor, PhD supervisor. His main research interest includes design automation for microfluidic biochips and integrated circuits.

黄　兴, 1991 年生. 博士, 教授, 博士生导师. 主要研究方向为微流控生物芯片和集成电路的设计自动化.

Liu Genggeng, born in 1988. PhD, professor, PhD supervisor. Senior member of CCF. His main research interest includes design automation for microfluidic biochips and integrated circuits.

刘耿耿, 1988 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为微流控生物芯片和集成电路的设计自动化.