Columba 2.0: A Co-Layout Synthesis Tool for Continuous-Flow Microfluidic Biochips

Tsun-Ming Tseng, *Member, IEEE*, Mengchu Li, *Student Member, IEEE*, Daniel Nestor Freitas, Travis McAuley, Bing Li, Tsung-Yi Ho, *Senior Member, IEEE*, Ismail Emre Araci, *Member, IEEE*, and Ulf Schlichtmann, *Member, IEEE*

***Abstract*—Continuous-flow microfluidic large-scale integration (mLSI) shows increasing importance in biological/chemical fields, thanks to its advantages in miniaturization and high throughput. Current mLSI is designed manually, which is time-consuming and error-prone. In recent years, design automation research for mLSI has evolved rapidly, aiming to replace manual labor by computers. However, previous design automation approaches used to design each microfluidic layer separately and over-simplify the layer interactions to various degrees, which resulted in a gap between realistic requirements and automatically-generated designs. In this work, we propose a module model library to accurately model microfluidic components involving layer interactions; and we propose a co-layout synthesis tool, Columba, which generates AutoCAD-compatible designs that fulfill all designs rules and can be directly used for mask fabrication. Columba takes plain-text netlist descriptions as inputs, and performs simultaneous placement and routing for multiple layers while ensuring the planarity of each layer. We validate Columba by fabricating two of its output designs. Columba is the first design automation tool that can seamlessly synchronize with the manufacturing flow.**

***Index Terms*—continuous-flow, microfluidics, microfluidic large-scale integration, mLSI, physical design, mixed integer linear programming.**

I. Introduction

Continuous-flow microfluidic large-scale integration (mLSI) has been widely applied in the last decade. By performing precise control of small reagent volume, mLSI shows advantages in efficiency and accuracy in high-throughput applications [1] [2] [3] [4]. The precise control is supported by layer interactions between multiple microfluidic layers. On mLSI, the mainstream structure consists of two layers: a flow layer and a control layer. Reaction samples and reagents are imported from external instruments along defined channels in the flow layer, and activated by manipulating the pressure in the control channels in the control layer. Figure 1 shows an mLSI design where control channels are indicated with green dye and flow channels are indicated with red dye. Layer interactions happen at valves, which are specific crossings of control and flow channels. As shown in Figure 1(a), valves are formed at those intersections of a control channel with a flow channel where the width of the control channel surpasses a given threshold. The pressure in the wide control segment of a valve results in a shape change of the membrane between the control layer and the flow layer, and thus blocks the fluid transportation in the corresponding flow channel segment. With this mechanism, flexible flow paths can be formed, and thus fluids are guided to their intended directions despite the crossings of flow channels. Valves enable the construction of complex microfluidic components such as rotary mixers as shown in Figure 1(b), and enlarge the scope of microfluidic applications significantly.

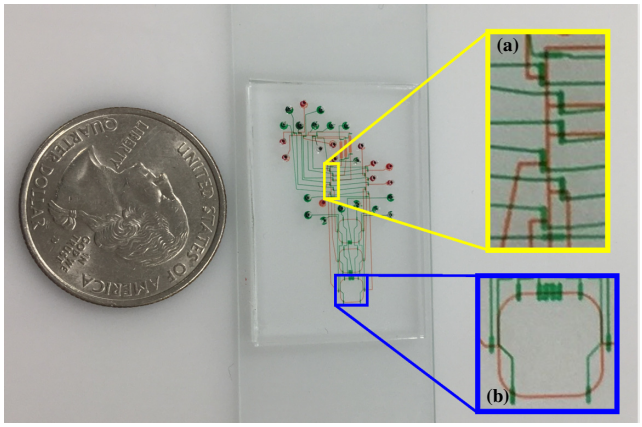


Figure 1: The design of a nucleic acid processor [3] automatically synthesized by the proposed tool Columba. (a) A switch. (b) A rotary mixer.

As the complexity of microfluidic applications increases, more complicated layer interactions need to be treated in the design process. Manual design of highly complex mLSI chips is time-consuming and error-prone, and thus slows down the optimization and implementation processes in microfluidic device development. In recent years, design automation research for mLSI has evolved rapidly and now provides a promising

The preliminary version of this paper was published in the Proceedings of the 53rd Annual Design Automation Conference (DAC), 2016.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes three multimedia GIF format movie clips, which demonstrate fluid routing and mixing on Columba-generated microfluidic designs. This material is 8.3 MB in size.

Tsun-Ming Tseng, Mengchu Li, Bing Li, and Ulf Schlichtmann are with the Chair of Electronic Design Automation, Technical University of Munich (TUM), Arcisstr. 21, Munich 80333, Germany (e-mail: tsun-ming.tseng@tum.de; mengchu.li@campus.lmu.de; b.li@tum.de; ulf.schlichtmann@tum.de).

Daniel Nestor Freitas, Travis McAuley, and Ismail Emre Araci are with Santa Clara University, 500 El Camino Real, Santa Clara, CA 95053 (e-mail: dnfreitas@scu.edu; tmcauley@scu.edu; iaraci@scu.edu).

Tsung-Yi Ho is with National Tsing Hua University, No. 101, Section 2, Kuang-Fu Road, Hsinchu 30013, Taiwan (e-mail: tyho@cs.nthu.edu.tw).

The work of Tsung-Yi Ho was supported in part by the Ministry of Science and Technology of Taiwan, under Grant MOST 102-2221-E-007-149-MY3 and 104-2220-E-007-021 and in part by the Technical University of Munich—Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763.

TCAD, DOI: 10.1109/TCAD.2017.2760628
http://ieeexplore.ieee.org/document/8060543/

solution to this problem.

The physical design for mLSI can be modeled as two interacting single-layered problems. The single-layered problem is a classic problem in the design automation field. It has a relatively smaller problem space compared with multi-layered problems, but needs to satisfy more design constraints such as architecture planarity. Different from multi-layered designs, where routing paths are usually allowed to overlap as long as they can be assigned to different layers, in single-layered designs, overlapping of on-chip components is either completely prohibited or penalized with significant costs, which greatly shrinks the solution space and raises the design difficulty. So far, new algorithms are still constantly proposed to solve the classic single-layered electronic circuit problems [5] [6].

As for mLSI, control layer design and flow layer design can be treated as two single-layered problems, where the control layer is prohibited from channel crossing, and the flow layer allows channel crossings but penalizes them with extra costs. However, existing solutions for single-layered electronic circuits cannot solve the microfluidic design problem, since the control and flow layers interact with each other through valves. Valves are implemented to guide the fluid direction in the flow layer, and are connected to pressure sources by the control channels in the control layer, which means that the control layer design depends on the placement of the corresponding to-be-controlled flow channels. Therefore, independent design for the flow layer is very likely to harm the routability of control channels.

In addition, there are two design factors in continuous-flow microfluidics that should be treated with caution: pressure sharing and netlist planarization.

Pressure sharing arises from the concern that the number of inlets/outlets on a chip is limited. Inlets/Outlets are punch holes on continuous-flow microfluidics used for pressure or fluid communication with external instruments. Compared with microfluidic channels which have a standard width of 0.1mm, a standard inlet/outlet has a size of 1mm^2 and thus occupies much more chip area [7]. In particular, complex applications require complicated valve actuation, which can result in a large number of control inlets/outlets that is beyond the manufacturing capability. A common approach to deal with this problem is pressure sharing [8] [9], which finds groups of valves that can share the same pressurizing sequence, without affecting the original functionality. This approach enables valves in the same group to share one single control inlet, but requires extra routing efforts to support the interconnection.

A netlist describes the logical connections among microfluidic components. Mathematical research demonstrates that not all logical connections can be transformed as physical connections in a 2D plane without introducing any crossing (i.e., can be drawn as a planar graph) [10]. Therefore, we enable channel crossings in the flow layer by introducing valves at cross points to guide the flow direction. As introducing additional valves results in additional routing burden on the control layer, it is essential to synchronize the two layers whenever flow channels cross.

Due to the above-mentioned design difficulties, existing design automation work only demonstrated partial solutions, which either focused on singled-layered placement or routing [8] [9], or simplified the problem by omitting inlets/outlets [11] [12] and assuming a naturally planar netlist [13]. An alternative is to apply homogeneous designs [14] [15]. However, though the homogeneous structure alleviates design difficulty, new challenges arise such as fluid routing, contamination owing to resource reuse, and assay scheduling prolongation. So far, no previous work was able to generate complete designs for mLSI, and thus could not synchronize with the manufacturing flow.

In this paper, we propose a module model library to accurately model microfluidic components involving layer interactions; and we propose a co-layout synthesis tool named ***Columba***, which generates AutoCAD-compatible designs that fulfill all design rules and can be directly used for mask fabrication. Columba takes plain-text netlist descriptions as inputs, and performs simultaneous placement and routing for multiple layers while ensuring the planarity of each layer. We validate the ability of Columba by fabricating two of its output designs. Columba is the first design automation tool that can seamlessly synchronize with the manufacturing flow.

II. Module Model for Microfluidics

On mLSI, operations depend on valve actuation, which brings layer interaction for fluid control. Therefore, we treat valves as the constituent elements of compound functional modules such as mixers, reaction chambers and switches, and propose a library to accurately model the inner-structure as well as inter-communication constraints of these modules.

We use the term *module model* to indicate an initial physical architecture for a specific type of microfluidics. The explicit design of each microfluidic component is refined from its corresponding module model. Figure 2 shows our module models proposed for mixers, switches, reaction chambers, and inlets/outlets. In our library, each module model is drawn as a bounding box with pins on its boundary for inter-communication. We allow the rotation of the bounding boxes to support more placement and routing solutions. Inside of the bounding boxes, flow channels are drawn as blue lines, control channels are drawn as green lines, and valves are drawn as orange rectangles. Channels drawn by dashed lines can be extended to satisfy different application requirements. In practice, the width of normal control channels is smaller than the width of valves, but we omit this difference in our module models for the sake of design convenience.

Our module model library enables us to model the complicated valve behavior as pin selection problems. Each valve is connected to one or two control pins of its corresponding modules. The selection of pins specifies the explicit inner-structure of a module. If a valve or a channel segment is not connected to any selected pin, it will be removed from the eventual design. Details of the selection are discussed in the following module descriptions.

Mixers are the common platforms for highly efficient mixing operations. Figure 2(a) shows our module model for mixers. The model consists of a ring-shaped flow channel segment, six valves for fluid control, two groups of valves forming peristaltic pumps, and several corresponding control

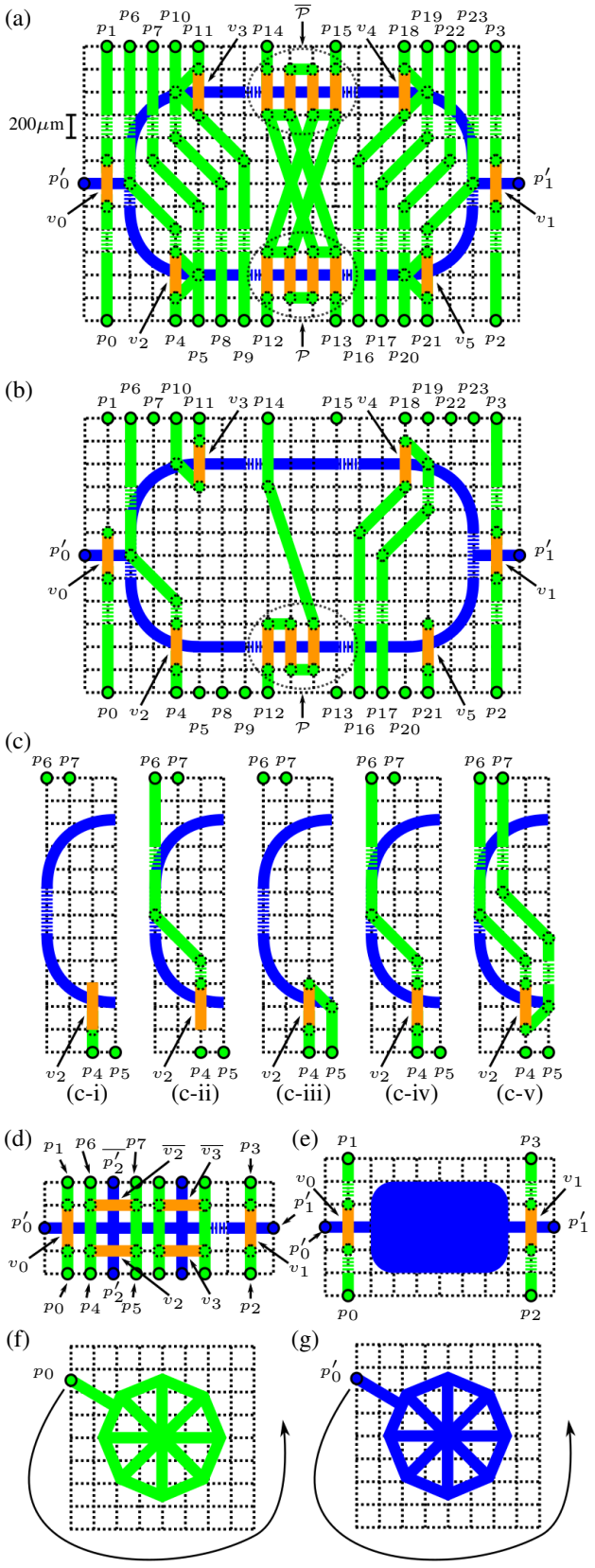


Figure 2: Module models: (a) Mixer. (b) A possible configuration for a mixer. (c) Provided options to connect the left-down valve (v_2) of the mixer module. (d) Switch. (e) Reaction chamber. (f) Control inlet. (g) Flow inlet/outlet.

TABLE I: Pin binding options to actuate valves.

mixer	
v_0	$\{p_0\},\{p_1\},\{p_0,p_1\}$
v_1	$\{p_2\},\{p_3\},\{p_2,p_3\}$
v_2	$\{p_4\},\{p_6\},\{p_4,p_5\},\{p_4,p_6\},\{p_6,p_7\}$
v_3	$\{p_9\},\{p_{11}\},\{p_8,p_9\},\{p_9,p_{11}\},\{p_{10},p_{11}\}$
v_4	$\{p_{16}\},\{p_{18}\},\{p_{16},p_{17}\},\{p_{16},p_{18}\},\{p_{18},p_{19}\}$
v_5	$\{p_{21}\},\{p_{23}\},\{p_{20},p_{21}\},\{p_{21},p_{23}\},\{p_{22},p_{23}\}$
$\mathcal{P}$	$\{p_{12}\},\{p_{13}\},\{p_{12},p_{13}\},\{p_{12},p_{14}\},\{p_{13},p_{15}\}$
$\overline{\mathcal{P}}$	$\{p_{14}\},\{p_{15}\},\{p_{14},p_{15}\}$

switch	
v_0	$\{p_0\},\{p_1\},\{p_0,p_1\}$
v_1	$\{p_2\},\{p_3\},\{p_2,p_3\}$
v_2	$\{p_4\},\{p_5\},\{p_6\},\{p_7\},\{p_4,p_5\},\ \{p_4,p_7\},\{p_5,p_6\},\{p_6,p_7\}$
$\overline{v_2}$	$\{p_4\},\{p_5\},\{p_6\},\{p_7\},\{p_4,p_5\},\ \{p_4,p_7\},\{p_5,p_6\},\{p_6,p_7\}$
...	... (analogous to v_2 and $\overline{v_2}$)

reaction chamber	
v_0	$\{p_0\},\{p_1\},\{p_0,p_1\}$
v_1	$\{p_2\},\{p_3\},\{p_2,p_3\}$

channels. The mixer module model is up-down asymmetric, which means rotating the module can result in four different orientations. The model shown in Figure 2(a) is horizontally placed without rotation. Our module model provides two options $\mathcal{P}$ and $\overline{\mathcal{P}}$ for the peristaltic pump, only one of which needs to be applied to perform the peristaltic protocol, and the other is removed from the eventual design. For each pump, we support two different physical structures composed of either three or four valves to enable flexible pin access. Our module model also provides multiple binding options for each valve. Figure 2(b) shows a possible configuration for a mixer, and Figure 2(c) shows five binding options provided for valve v_2. If v_2 is at the end of a pressure transportation path, it can be accessed either from the bottom side through pin p_4 or from the upper side through pin p_6. If v_2 is a transfer station along a pressure transportation path (i.e., v_2 shares pressure with other valves), it can be accessed either through pins on opposite module boundaries ((p_4,p_6)), or through pins on the same boundaries ((p_4,p_5), (p_6,p_7)). The detailed pin binding options for each valve and pump are listed in Table I. As shown in the table, our module model enables $3^2{\cdot}5^4{\cdot}(5{+}3){=}45000$ variants of the mixer structure, and thus provides strong support for flexible placement and routing strategies.

Switches are used to guide the fluid direction when flow channels cross. Figure 2(d) shows our module model for switches. The model consists of one main flow channel and several flow channel junctions. The number of junctions in a switch can be managed to support different communication protocols. For each channel junction, one of the flow pins on opposite module boundaries will be chosen to form the expected transportation path. Channel segments connecting the main channel with the unchosen pins will be removed from the eventual design. Note that flow pin selection influences the valve implementation since valves controlling the removed

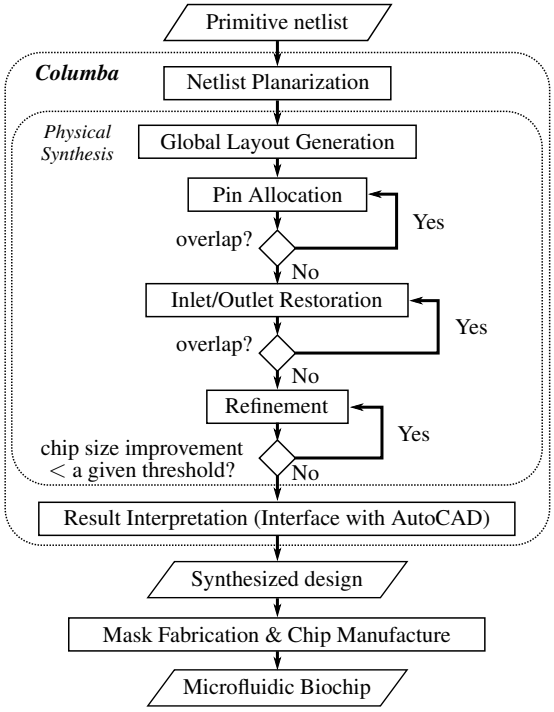


Figure 3: Chip production process supported by Columba.

flow channels will also be removed. Our switch module model also provides multiple binding options for each valve, the details are shown in Table I.

Reaction chambers are general platforms for many different operations [16] [17] [18]. A reaction chamber usually consists of a flow channel segment with two valves at both ends. Figure 2(e) shows our module model for reaction chambers. The height and width of the chamber are adjustable to support different application requirements. Detailed binding options for the two valves are listed in Table I.

Inlets/Outlets are punch holes on continuous-flow microfluidics for pressure or fluid communication with external instruments. Figure 2(f) and (g) show our module models for control and flow inlets/outlets, respectively. The models are designed according to the specifications in [7]. For the sake of planarity, each inlet/outlet possesses only one pin, which can be located anywhere on the module boundary.

III. Ovarall Flow of Columba

Columba is the first design automation tool for continuous-flow microfluidics that can synchronize with the manufacturing flow. We depict the complete chip production process supported by Columba in Figure 3.

The input of the production process is a plain-text **primitive netlist** of the required design, which specifies the number, type, dimension, and logic connection of the required mixers, reaction chambers, and flow inlets/outlets, as well as particular execution constraints such as mixers or reaction chambers executing in parallel. Columba starts its design process with **netlist planarization**, which transforms the primitive netlist into a planarity-guaranteed netlist by adding switch modules to the netlist. The refined netlist is then taken as the input for **physical synthesis**. Columba models the physical synthesis problem as a linear optimization problem, and solves it progressively in four sequential phases: **global layout generation** phase outputs a layout topology and specifies the pressure sharing relation among valves; **pin allocation** phase then details the design by performing explicit channel routing; **inlet/outlet restoration** phase further complements the design by adding inlets/outlets back to the design. After the first three physical synthesis phases, a valid design that fulfills all design-rules will be achieved. This design will be taken by the iterative **refinement** phase to pursue chip area and channel length reduction. When the improvement achieved in the refinement phase is smaller than a given threshold, the physical synthesis process terminates and the output will be sent for **result interpretation** and transformed into an AutoCAD [19] script. This script can be directly read by AutoCAD and then exported for **mask fabrication and chip manufacture**. We demonstrate two fabricated **microfluidic biochips** in Section VII.

IV. Netlist Planarization

The input of Columba is a primitive netlist in plain-text format, which specifies the number, type, dimension and logic connection of the required mixers, reaction chambers and flow inlets/outlets. Switches and control inlets/outlets are not required to be specified in this primitive netlist, since their implementation highly depends on the chip physical structure, and thus can hardly be predicted by biochip end-users.

As we have mentioned in Section I, not all logical connections can be transformed as physical connections in a 2D plane without introducing any crossing. For continuous-flow microfluidic design, when transforming the logic connections specified in the primitive netlist into physical connections, i.e. placement and routing solutions, flow channel crossings can be inevitable. In order to guarantee feasible fluid transportation paths, switches need to be implemented at flow channel junctions. So far, previous design automation work either neglects this issue or introduces switches arbitrarily when performing physical synthesis [11] [20] [21]. However, the implementation of switches not only affects flow channels but also results in extra routing efforts in the control layer for the actuation of valves. Since previous approaches cannot foresee the existence of switches and thus cannot reserve appropriate resources for them, the feasibility of the designs cannot be guaranteed.

Columba embeds a netlist planarization approach before physical synthesis to solve this problem. It adds switches to the netlist and thus turns the primitive netlist into a planarity-guaranteed netlist, where every logic connection can be transformed into feasible flow channel connection without introducing any channel crossing. The new netlist clearly specifies the number of switches, the number of channel junctions in each switch, and logic connections among switches and other modules, which enables our physical synthesis algorithm to model the resource usage accurately.

Before introducing the proposed netlist planarization, we first take a look at the formal definition of planarity. Planarity is defined in Kuratowski's theorem [10]: *a graph is planar if and only if the graph does not contain a subgraph that is a subdivision of* K_5 *or of* $K_{3,3}$. Based on this theorem, we give the following lemma:

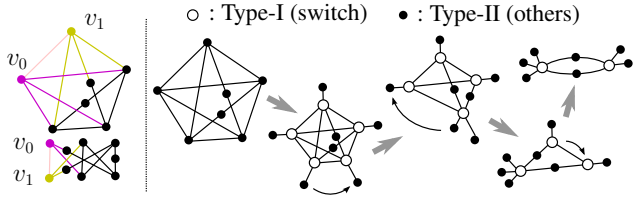


Figure 4: (a) Subdivision examples of K_5 and $K_{3,3}$. (b) An example of introducing and merging Type-I modules (switches) to planarize the netlist.

Lemma. *For a graph $G=(V,E)$ that does not contain a barycentric subdivision of K_5 or $K_{3,3}$, G is non-planar only if there exist two adjacent vertices $v_0,v_1\in V$ both with degrees larger than or equal to 3.*

To verify this lemma, we aim to show that a non-barycentric subdivision of K_5 or $K_{3,3}$ contains two adjacent vertices $v_0,v_1\in V$ both with degrees larger than or equal to 3. A barycentric subdivision is a special subdivision that subdivides each edge of the respective graph. As shown in Figure 4(a), a non-barycentric subdivision of K_5 contains at least two adjacent vertices both with degree 4, and a non-barycentric subdivision of $K_{3,3}$ contains at least two adjacent vertices both with degree 3.

Corollary. *In the graph $G=(V,E)$ representing the input netlist, if G contains no barycentric subdivision of K_5 or of $K_{3,3}$ as its subgraph, and for every vertex with a degree larger than or equal to 3, each of its adjacent vertices has a degree no more than 2, the graph is planar.*

Based on the derived corollary, we classify our modules into two types: Type-I includes switch modules with unconstrained degree, and Type-II includes mixers, reaction chambers, and inlet/outlet modules with a degree limited to no more than 2. As shown in Figure 4(b), white circles represent Type-I modules and black circles represent Type-II modules. If a Type-II module needs to receive inputs from or deliver outputs to multiple other Type-II modules, instead of connecting these modules directly with each other, we connect them with Type-I modules as transfer stations to meet the degree limitation. If two Type-I modules are connected directly with each other, we will merge them together to form a new Type-I module. Therefore, for every two adjacent modules, at least one of them has the degree restricted to no more than 2, which means the netlist graph must be planar.

When no new Type-I module is introduced and no more modules are merged, a check for the existence of barycentric subdivision will then be performed. If a barycentric subdivision of K_5 or $K_{3,3}$ is ascertained, we will arbitrarily pick 2 Type-I modules from this subdivision and merge them together until no more barycentric subdivision exists, so that the netlist planarity can be guaranteed.

V. Co-Layout Physical Synthesis

With the planarized netlist, Columba solves the physical synthesis problem progressively in four sequential phases. Each phase is modeled as a mathematical linear optimization problem, sharing some general modeling characteristics.

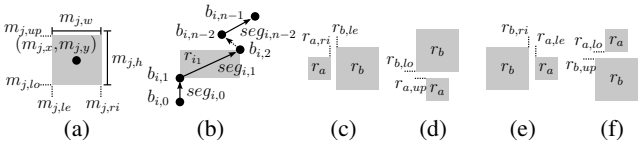


Figure 5: (a) Channel model. (b) Module model (dimension). (c)(d)(e)(f) Non-overlapping rectangles r_a and r_b.

A. General Modeling Characteristics

We model the physical structure of continuous flow microfluidics as points and lines in a 2D coordinate plane and describe the design constraints as linear formulas. In this paragraph, we introduce the common constraints shared by our four physical synthesis phases.

1) Module, Channel and Chip Area

The fundamental building blocks in our mathematical models are modules and channels, which are specified by characteristic points.

For a module m_j with index $j\in\mathbb{N}$, we specify the coordinates of its center point as $(m_{j,x},m_{j,y})$, and we specify the width and height of the module as $m_{j,w}$ and $m_{j,h}$. Thus, we can specify the module boundaries as $m_{j,le},m_{j,ri},m_{j,up},m_{j,lo}$, where $m_{j,le}=m_{j,x}-\frac{1}{2}m_{j,w}$, $m_{j,ri}=m_{j,x}+\frac{1}{2}m_{j,w}$, $m_{j,up}=m_{j,y}+\frac{1}{2}m_{j,h}$, and $m_{j,lo}=m_{j,y}-\frac{1}{2}m_{j,h}$, as shown in Figure 5(a). We denote the set of all modules in a design as M.

We model a channel as a polyline connected with two modules. As shown in Figure 5(b), for a channel c_i with index $i\in\mathbb{N}$, we denote its segments as $seg_{i,0},\cdots,seg_{i,n-2},n\in\mathbb{N},n\geq 2$; and its bending points as $b_{i,0},\cdots,b_{i,n-1}$, thereof $b_{i,k}$ and $b_{i,k+1}$ with $0\leq k\leq n-2$, $k\in\mathbb{N}$ are called the ends of the segment $seg_{i,k}$; and $b_{i,0}$ and $b_{i,n-1}$ are called the ends of the channel c_i. We model the length of $seg_{i,k}$ as the Manhattan distance between its segment ends, which is denoted by a continuous variable $l_{i,k}$ and confined by the following constraints:

$$(b_{i,k,x}-b_{i+1,k,x})+(b_{i,k,y}-b_{i+1,k,y})\leq l_{i,k}, \quad (1)$$
$$(b_{i,k,x}-b_{i+1,k,x})+(b_{i+1,k,y}-b_{i,k,y})\leq l_{i,k}, \quad (2)$$
$$(b_{i+1,k,x}-b_{i,k,x})+(b_{i,k,y}-b_{i+1,k,y})\leq l_{i,k}, \quad (3)$$
$$(b_{i+1,k,x}-b_{i,k,x})+(b_{i+1,k,y}-b_{i,k,y})\leq l_{i,k}. \quad (4)$$

We denote the set of all channels in a design as C. For each $c_i\in C$, we denote the set of all segments in c_i as S_{c_i} and the set of all bending points in c_i as B_{c_i}.

We introduce two continuous variables: x_τ, y_τ to represent the x-dimension and the y-dimension of a chip, and we denote $\max\{x_\tau,y_\tau\}$ as a continuous variable d_τ. The related constraints are listed as follows:

$$x_\tau\leq d_\tau,\ y_\tau\leq d_\tau; \quad (5)$$

$\forall c_i\in C$, $\forall(b_{i,k,x},b_{i,k,y})\in B_{c_i}$,

$$\rho_c\leq b_{i,k,x}\leq x_\tau-\rho_c, \quad (6)$$
$$\rho_c\leq b_{i,k,y}\leq y_\tau-\rho_c; \quad (7)$$

$\forall m_j \in M$,

$$\rho_m \leq m_{j,le}, \tag{8}$$
$$m_{j,ri} \leq x_\tau - \rho_m, \tag{9}$$
$$\rho_m \leq m_{j,do}, \tag{10}$$
$$m_{j,up} \leq y_\tau - \rho_m; \tag{11}$$

where ρ_c (resp. ρ_m) is a user-defined constant representing the minimum spacing distance from a channel (resp. a module boundary) to the chip boundary.

2) Prohibition on Overlapping Modules and Channels

Our planarity-guaranteed netlist ensures the feasibility for Columba to generate a design without any overlapping modules and channels.

In order to model the area consumption of channels, we consider each channel segment as the diagonal of a virtual rectangle. An example is shown in Figure 5(b), where the gray rectangle r_{i_1} represents the area consumption of its diagonal $seg_{i,1}$. For a channel segment $seg_{i,k}$, we denote the coordinates of its two ends as $(b_{i,k,x}, b_{i,k,y})$, $(b_{i,k+1,x}, b_{i,k+1,y})$, and specify the boundaries of its virtual rectangle r_{i_k} by $r_{i_k,le}, r_{i_k,ri}, r_{i_k,up}, r_{i_k,lo}$, where $r_{i_k,le} = \min\{b_{i,k,x}, b_{i,k+1,x}\}$, $r_{i_k,ri} = \max\{b_{i,k,x}, b_{i,k+1,x}\}$, $r_{i,k,up} = \max\{b_{i,k,y}, b_{i,k+1,y}\}$, and $r_{i,k,lo} = \min\{b_{i,k,y}, b_{i,k+1,y}\}$, linearized by the following constraints:

$$r_{a,ri} \geq b_{i,k,x}, \quad \wedge \quad r_{a,ri} \geq b_{i,k+1,x}, \tag{12}$$
$$r_{a,up} \geq b_{i,k,y}, \quad \wedge \quad r_{a,up} \geq b_{i,k+1,y}, \tag{13}$$
$$r_{a,le} \leq b_{i,k,x}, \quad \wedge \quad r_{a,le} \leq b_{i,k+1,x}, \tag{14}$$
$$r_{a,lo} \leq b_{i,k,y}, \quad \wedge \quad r_{a,lo} \leq b_{i,k+1,y}. \tag{15}$$

For the convenience of description, in this paragraph, we also use the notation r_j to represent the area consumption of a module m_j, where $r_{j,le} = m_{j,le}$, $r_{j,ri} = m_{j,ri}$, $r_{j,up} = m_{j,up}$, and $r_{j,lo} = m_{j,lo}$.

As shown in Figure 5(c)(d)(e)(f), there are four possible relative locations for two nearby rectangles r_a and r_b without area overlap. We model these relative locations as the following constraints:

$$r_{a,ri} \leq r_{b,le} - \rho_{a,b} + q_1\mathcal{M}, \tag{16}$$
$$r_{a,up} \leq r_{b,lo} - \rho_{a,b} + q_2\mathcal{M}, \tag{17}$$
$$r_{a,le} \geq r_{b,ri} + \rho_{a,b} - q_3\mathcal{M}, \tag{18}$$
$$r_{a,lo} \geq r_{b,up} + \rho_{a,b} - q_4\mathcal{M}, \tag{19}$$
$$q_1 + q_2 + q_3 + q_4 = 3, \tag{20}$$

where q_1, q_2, q_3, and q_4 are auxiliary binary variables, $\mathcal{M}$ is an extremely large auxiliary constant, and $\rho_{a,b}$ is the user-defined minimum spacing distance between r_a and r_b. With (20), exactly one of q_1, q_2, q_3, and q_4 must be 0, which means the relative position of every two nearby rectangles must fulfill one of the above mentioned four situations.

3) Objective

Our four physical synthesis phases share some general optimization objectives, i.e. chip area reduction and channel length reduction.

Chip area is an important metric to evaluate a continuous-flow microfluidic design. Smaller chips require less manufacturing cost and provide convenience for microscope observation. Besides the x-dimension x_τ and the y-dimension y_τ of the chip, we also include the continuous variable $d_\tau = \max\{x_\tau, y_\tau\}$ into the optimization objective to balance the two dimensions.

Channel length is another metric to evaluate a continuous-flow microfluidic design, since long channels raise reliability and yield concerns [22]. For each channel $c_i \in C$, we compute its length as the total length of its segments $\sum_{seg_{i,k} \in S_{c_i}} l_{i,k}$.

B. Global Layout Generation

The global layout generation phase aims to decide the layout topology, setting the foundation for the posterior optimization phases.

1) Model Reduction for Modules and Channels

In the global layout generation phase, we perform model reduction for modules and channels to alleviate computing effort.

Our module models for mixers, reaction chambers and switches proposed in Section II are asymmetric, i.e. rotating the modules can change the module orientation and thus affect the physical synthesis solutions. Columba omits the influence of module rotation in the global layout generation phase, and models each module $m_j \in M$ as a simple square bounding box, the side length of which is set as $\max\{m_{j,h}, m_{j,w}\}$. With center point coordinate $(m_{j,x}, m_{j,y})$, the module boundaries can be computed by $m_{j,le} = m_{j,x} - \frac{1}{2}\max\{m_{j,h}, m_{j,w}\}$, $m_{j,ri} = m_{j,x} + \frac{1}{2}\max\{m_{j,h}, m_{j,w}\}$, $m_{j,up} = m_{j,y} + \frac{1}{2}\max\{m_{j,h}, m_{j,w}\}$, and $m_{j,lo} = m_{j,y} - \frac{1}{2}\max\{m_{j,h}, m_{j,w}\}$. Pins are omitted in this phase, and channels are routed to center points of modules instead.

In the global layout generation phase, a channel $c_i \in C$ is modeled as a straight line instead of the proposed polyline. The simplified channel model for c_i only has one segment $s_{i,0}$ and two bending points $b_{i,0}$ and $b_{i,1}$, where $b_{i,0}$ and $b_{i,1}$ represent both the channels ends and the segment ends. Regarding the relatively large number of control channels, Columba collects control channels whose ends share the same coordinates as control *channel bundle*s, and performs routing for control channel bundles instead of explicit control channel routing. For example, if m_a and m_b are two modules with center points $(m_{a,x}, m_{a,y})$ and $(m_{b,x}, m_{b,y})$, $C_{a,b}$ are the set of all control channels that connect m_a with m_b, i.e. $\forall c_k \in C_{a,b}, B_{c_k} = \{(m_{a,x}, m_{a,y}), (m_{b,x}, m_{b,y})\}$, then all channels in $C_{a,b}$ are collected to form a control channel bundle $\mathring{c}_{a,b}$, which is routed as a single control channel in the global layout generation phase. We introduce an integer variable $n_{\mathring{c}_{a,b}}$ to denote the number of control channels contained in $\mathring{c}_{a,b}$, i.e., $n_{\mathring{c}_{a,b}} = |C_{a,b}|$, for channel restoration in the posterior phase.

In addition, we do not consider the area consumption of control and flow inlets/outlets, and omit all channels connected to control and flow inlets/outlets in this phase.

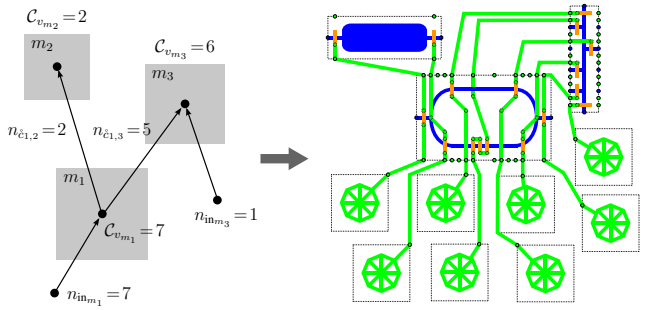


Figure 6: (a) An example of pressure-sharing result. (b) A possible layout based on (a) after splitting bundles, allocating pins, and restoring control inlets.

2) Pressure Sharing

On mLSI, a valve has two states: *open* indicates that the valve is not actuated; and *closed* indicates that the valve is actuated and thus its underlying flow channel is blocked. If several valves are sequentially connected to the same control inlet, we can assume that these valves share the same status, since pressure from the control inlet will be transported along the entire control channel and actuates all the passing valves. We call this feature *pressure sharing*. In this work, we consider pressure sharing as a feature among modules. For arbitrary two modules $m_a, m_b \in M$, if there exists a valve v_a in m_a and a valve v_b in m_b, where v_a and v_b are connected to the same control inlet, we say m_a *shares pressure* with m_b. The valves that belong to the same module, e.g., peristaltic valves sequentially connected in a mixer, will not be discussed as pressure sharing objects.

Based on different triggering conditions, we classify pressuring sharing into two categories: *passive pressure sharing* and *active pressure sharing*.

Passive pressuring sharing has been discussed in [8], referred as "valve compatibility". [8] introduces a new valve status: X, which represents *don't care*, indicating that the module that the valve belongs to is idling, and thus the status of the valve has no impact on the application. Therefore, if two modules do not have any overlapping working period, i.e., one of the two modules must idle when the other one is working, all valves in the idling module are in status X and can be arbitrarily actuated. Columba takes advantage of this feature and allows modules without overlapping working period to share pressure with each other. For two modules m_a and m_b that are allowed to share pressure, Columba assigns two control channel bundles $\mathring{c}_{a,b}, \mathring{c}_{b,a}$ to them, and introduces the following constraints:

$$n_{\mathring{c}_{a,b}} - q_5 \mathcal{M} \leq 0, \tag{21}$$

$$n_{\mathring{c}_{b,a}} - q_6 \mathcal{M} \leq 0, \tag{22}$$

$$q_5 + q_6 \leq 1, \tag{23}$$

where q_5 and q_6 are auxiliary binary variables, and $\mathcal{M}$ is an extremely large auxiliary constant. Above constraints make sure that at least one of $n_{\mathring{c}_{a,b}}, n_{\mathring{c}_{b,a}}$ must be 0, i.e., $\mathring{c}_{a,b}, \mathring{c}_{b,a}$ cannot both exist. According to the direction of pressure transportation, we call $\mathring{c}_{a,b}$ an *outgoing* channel bundle of m_a, and an *incoming* channel bundle of m_b. For each module m_j, valves that belong to m_j receive pressure either directly from control inlets, or from other modules via incoming control channel bundles of m_j. We introduce an integer variable $n_{\text{in}_{m_j}}$ to represent the number of control inlets that are connected to m_j, a set C_{in,m_j} that contains all the incoming channel bundles of m_j, a set C_{out,m_j} that contains all the outgoing channel bundles of m_j, and a constant $C_{v_{m_j}}$ to represent the number of valves that demand individual control in m_j. For valves that are sequentially connected in the same module, e.g., a pump in a mixer, we consider that only one valve demands individual control. The pressure sharing condition thus can be modeled as the following constraints:

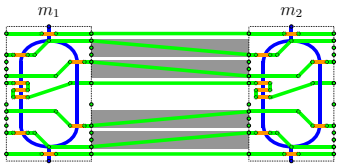


Figure 7: Straight channel connection between parallel modules without causing overlapping of vitual rectangles.

$$n_{\text{in}_{m_j}} + \sum_{\mathring{c}_{k,j} \in C_{\text{in},m_j}} n_{\mathring{c}_{k,j}} = \mathcal{C}_{v_{m_j}}, \tag{24}$$

$$\sum_{\mathring{c}_{j,k} \in C_{\text{out},m_j}} n_{\mathring{c}_{j,k}} \leq \mathcal{C}_{v_{m_j}}. \tag{25}$$

Constraint (24) makes sure that every valve or pump must be connected either directly to a control inlet, or to another module via an incoming control channel bundle. Constraint (25) makes sure that the total number of control channels contained in the outgoing control channel bundles of m_j does not surpass the number of valves in m_j that demands individual control.

An example of the modeling results for passive pressure sharing is shown in Figure 6(a). m_1 is a mixer containing 7 valves that require individual control according to our module model library. All these valves are directly connected to 7 control inlets. m_1 shares pressure with a chamber m_2 and a switch m_3, which contain 2 and 6 valves, respectively. All valves in m_2 are connected via $\mathring{c}_{m_{1,2}}$ to m_1. One valve in m_3 is connected to a control inlet, and all the rest are connected via $\mathring{c}_{1,3}$ to m_1. Figure 6(b) shows a possible physical layout of the chip that can be generated in later phases based on the pressure sharing modeling results. Flow channels outside the modules are omitted in Figure 6(b) since they are not related to pressure sharing.

Active pressure sharing is required by some application protocols to ensure synchronous pressure transportation. [23] has discussed some common execution limitations in biological applications, one of which is *parallel execution*. Large biological applications are usually composed of replicate operations to improve throughput [3] [24] [25] [26]. These replicate operations are executed in parallel and require synchronous valve control to streamline the process. We call the modules for executing the replicate operations *parallel modules*. Since parallel modules share the same module model, their valves

at the same relative locations can be directly connected to share pressure, as shown in Figure 7. For a group of n parallel modules $m_p,\cdots,m_{p+n-1}$, Columba applies constraints (24) and (25) to them, builds up a group of channel bundles $\mathring{c}_{k,k+1},\ k\in\mathbb{N}, p\leq k<p+n-1$, and introduces the following constraints:

$$n_{\text{in}_{m_p}}=\mathcal{C}_{v_{m_p}}, \tag{26}$$

$$n_{\text{out}_{m_{p+n-1}}}=0, \tag{27}$$

$$n_{\mathring{c}_{k,k+1}}=\mathcal{C}_{v_{m_p}}, \tag{28}$$

making sure that the pressure shared by parallel modules are transported sequentially from a group of control inlets to every parallel module.

3) Opposite Neighbor Modules

As mentioned in Section II, a module model can be regarded as a bounding box with pins on its boundary. Modules that have a logic connection according to the input netlist need to be physically connected by flow channels via flow pins, and modules that share pressure with each other need to be connected by control channels via control pins. In the global layout generation phase, we propose a heuristic method regarding the pin location of modules to provide convenience for explicit channel routing in the posterior synthesis phases.

Our input netlist of the physical synthesis phase specifies the flow layer logic connections among modules. Mixer modules and reaction chamber modules that are logically connected in the flow layer are defined as *flow neighbors*. Flow neighbors must be connected via flow pins. In our proposed module models for mixers and reaction chambers, each module has two flow pins, which means that each module can have no more than two flow neighbors. For a mixer or a reaction chamber module m_j, we denote its two flow neighbors as $m_{j_{n_1}}$ and $m_{j_{n_2}}$. As shown in our module model library in Section II, the two flow pins of m_j are located at the opposite module boundaries. In order to prevent long detours of flow channels, we introduce the following constraints to place $m_{j_{n_1}}$ and $m_{j_{n_2}}$ at the opposite sides of m_j, and thus enable them to be connected straightforward to the opposite flow pins of m_j:

$$m_{j_{n_1},ri}\leq m_{j,le}-\rho_n+q_7\mathcal{M}, \tag{29}$$

$$m_{j,ri}\leq m_{j_{n_2},le}-\rho_n+q_7\mathcal{M}, \tag{30}$$

$$m_{j_{n_1},le}\geq m_{j,ri}+\rho_n-q_8\mathcal{M}, \tag{31}$$

$$m_{j,le}\geq m_{j_{n_2},ri}+\rho_n-q_8\mathcal{M}, \tag{32}$$

$$m_{j_{n_1},up}\leq m_{j,lo}-\rho_n+q_9\mathcal{M}, \tag{33}$$

$$m_{j,up}\leq m_{j_{n_2},lo}-\rho_n+q_9\mathcal{M}, \tag{34}$$

$$m_{j_{n_1},lo}\geq m_{j,up}+\rho_n-q_{10}\mathcal{M}, \tag{35}$$

$$m_{j,lo}\geq m_{j_{n_2},up}+\rho_n-q_{10}\mathcal{M}, \tag{36}$$

$$q_7+q_8+q_9+q_{10}=3, \tag{37}$$

where ρ_n is a user-defined constant representing the spacing distance between flow neighbors, q_7, q_8, q_9, and q_{10} are auxiliary binary variables, one of which must be set to 0 according to constraint (37), representing an alignment possibility. For example, if q_7 is set to 0, it means that $m_{j_{n_1}}$ is located at the left side of m_j and $m_{j_{n_2}}$ is located at the right side of m_j. Note that in the global layout generation phase, modules are modeled as square bounding boxes without orientation. Therefore, flow neighbors of m_j can be placed either at the horizontally opposite sides of m_j or at the vertically opposite sides of m_j. The orientation of m_j can be fixed according to different modeling results of flow neighbor locations. If the two flow neighbors of m_j are placed horizontally opposite of each other, m_j will be placed horizontally as well, as shown in Figure 2(a), so that its two flow neighbors can easily access their corresponding flow pins. Analogously, if the flow neighbors of m_j are placed at the vertically opposite sides of m_j, m_j will be placed vertically, too, so that its flow pins can be accessed on the upper and lower module boundaries.

Mixer and reaction chamber modules that share pressure need to be connected by control channels via control pins. For a group of parallel modules that share pressure actively, each module is connected with no more than two other modules in the control layer. We call parallel modules that are directly connected in the control layer *control neighbors*. In our module models for mixers and reaction chambers, the module boundaries that control pins belong to are perpendicular to the module boundaries that flow pins belong to. For example, if a mixer or a reaction chamber module m_j is placed horizontally with flow pins on its left and right module boundaries, the control pins of m_j must be on the upper and lower module boundaries. Therefore, we introduce the following constraints to each group of parallel modules $m_p,\cdots,m_{p+n-1}$, to let the control neighbors of each module in this group be placed at the direction perpendicular to its flow neighbors.

$\forall p\leq k<p+n-1, k\in\mathbb{N}$:

$$m_{k+1,x}\leq m_{k,x}+q_7\mathcal{M}, \tag{38}$$

$$m_{k,x}\leq m_{k+1,x}+q_7\mathcal{M}, \tag{39}$$

$$m_{k+1,x}\leq m_{k,x}+q_8\mathcal{M}, \tag{40}$$

$$m_{k,x}\leq m_{k+1,x}+q_8\mathcal{M}, \tag{41}$$

$$m_{k+1,y}\leq m_{k,y}+q_9\mathcal{M}, \tag{42}$$

$$m_{k,y}\leq m_{k+1,y}+q_9\mathcal{M}, \tag{43}$$

$$m_{k+1,y}\leq m_{k,y}+q_{10}\mathcal{M}, \tag{44}$$

$$m_{k,y}\leq m_{k+1,y}+q_{10}\mathcal{M}. \tag{45}$$

If a flow neighbor of m_k is placed at the left or the right side of m_k, i.e., q_7 or q_8 is set to 0 according to constraints (29)-(32), the x-coordinate of m_k and m_{k+1} are forced to be the same according to constraints (29)-(32). Analogously, if a flow neighbor of m_k is placed at the upper or the lower side of m_k, i.e., q_9 or q_{10} is set to 0 according to constraints (33)-(36), the y-coordinate of m_k and m_{k+1} are forced to be the same according to constraints (42)-(45). Note that we force parallel modules to be placed in alignment. As shown in Figure 7, our module models support straightforward control channel connections between parallel modules that are in alignment, and thus much routing effort can be saved.

4) Global Layout Generation Model

In the global layout generation phase, besides the general optimization objectives as mentioned in Section V-A3,

Columba also aims to minimize the number of control inlets when determining the control layer layout. Thus, the global layout generation model can be formulated as follows:

Minimize:

$$\alpha d_\tau+\alpha_1 x_\tau+\alpha_2 y_\tau+\beta\sum_{c_i\in C}\sum_{seg_{i,k}\in S_{c_i}} l_{i,k}+\gamma\sum_{m_j\in M} n_{\text{in}_{m_j}}$$

Subject to: (1)–(45)

where α, α_1, α_2, β, and γ are adjustable weight coefficients.

C. Pin Allocation

After the global layout generation phase, Columba obtains a layout topology where channels are routed to module center points instead of pins. The pin allocation phase restores the original models for modules and channels, and routes channels back to pins.

1) Model Restoration for Modules and Channels

In the global layout generation phase, modules are modeled as square bounding boxes without pins. We restore the original module models proposed in Section II in the pin allocation phase. For a module m_j, we denote the set of its control pins as $P_{j,c}$ and the set of its flow pins as $P_{j,f}$. We index all pins as shown in Figure 2, and denote the k-th control (resp. flow) pin of a module m_j as $p_{j,k}$ (resp. $p'_{j,k}$). The coordinates of $p_{j,k}$ (resp. $p'_{j,k}$) can be calculated as $(m_{j,x}+\kappa_{j,k,x}, m_{j,y}+\kappa_{j,k,y})$ (resp. $(m_{j,x}+\kappa'_{j,k,x}, m_{j,y}+\kappa'_{j,k,y})$), where $\kappa_{j,k,x}$ (resp.$\kappa'_{j,k,x}$) and $\kappa_{j,k,y}$ (resp.$\kappa'_{j,k,y}$) are constants indicating the horizontal and vertical offset distances from the center point of m_j to $p_{j,k}$ (resp. $p'_{j,k}$).

In the global layout generation phase, control channels between two modules m_a and m_b are collected as a control bundle $\mathring{c}_{a,b}$, specified with an integer $n_{\mathring{c}_{a,b}}$ indicating the number of actual control channels contained in the channel bundle. We split a channel bundle $\mathring{c}_{a,b}$ back to $n_{\mathring{c}_{a,b}}$ explicit control channels in the pin allocation phase. We call the control channels split from $\mathring{c}_{a,b}$ *outgoing control channels* of m_a, and *incoming control channels* of m_b. For a module m_j, we denote the set of all its incoming (resp. outgoing) control channels as C_{c_{in},m_j} (resp. C_{c_{out},m_j}). Analogously, we denote the set of incoming (resp. outgoing) flow channels of m_j as C_{f_{in},m_j} (resp. C_{f_{out},m_j}).

2) Routing Channels to Pins

We introduce an auxiliary binary variable $q_{c_i,p_{j,k}}$ (resp. $q_{c_i,p'_{j,k}}$) to indicate whether a control (resp. flow) channel c_i is connected to a module m_j via pin $p_{j,k}$ (resp. $p'_{j,k}$). We apply the following constraints to each module m_i to make sure that each pin of m_i is accessed by no more than one channel:

$$\forall p_{j,k}\in P_{j,c},\ \sum_{c_i\in C_{c_{\text{in}},m_j}\cup C_{c_{\text{out}},m_j}} q_{c_i,p_{j,k}}\le 1, \tag{46}$$

$$\forall p'_{j,k}\in P_{j,f},\ \sum_{c_i\in C_{f_{\text{in}},m_j}\cup C_{f_{\text{out}},m_j}} q_{c_i,p'_{j,k}}\le 1. \tag{47}$$

Then we introduce the following constraints to each channel c_i that is connected to m_j to make sure that c_i is connected to exactly one pin of m_j:

$$\forall c_i\in C_{c_{\text{in}},m_j}\cup C_{c_{\text{out}},m_j},\ \sum_{p_{j,k}\in P_{j,c}} q_{c_i,p_{j,k}}=1, \tag{48}$$

$$\forall c_i\in C_{f_{\text{in}},m_j}\cup C_{f_{\text{out}},m_j},\ \sum_{p'_{j,k}\in P_{j,f}} q_{c_i,p'_{j,k}}=1. \tag{49}$$

Thus, for each incoming control channel $c_i\in C_{c_{\text{in}},m_j}$ with bending points $b_0,\cdots,b_{n-1}$, we introduce the following constraints to model its pin connection:

$$b_{i,n-1,x}=m_{j,x}+\sum_{p_{j,k}\in P_{j,c}} q_{c_i,p_{j,k}}\kappa_{j,k,x}, \tag{50}$$

$$b_{i,n-1,y}=m_{j,y}+\sum_{p_{j,k}\in P_{j,c}} q_{c_i,p_{j,k}}\kappa_{j,k,y}, \tag{51}$$

making sure that if $q_{c_i,p_{j,k}}$ is set to 1, the control channel end $b_{i,n-1}$ and the pin $p_{j,k}$ will have the same coordinate. Similar constraints are also introduced to each outgoing control channel $c_i\in C_{c_{\text{out}},m_j}$, only with different notation of the channel end:

$$b_{i,0,x}=m_{j,x}+\sum_{p_{j,k}\in P_{j,c}} q_{c_i,p_{j,k}}\kappa_{j,k,x}, \tag{52}$$

$$b_{i,0,y}=m_{j,y}+\sum_{p_{j,k}\in P_{j,c}} q_{c_i,p_{j,k}}\kappa_{j,k,y}. \tag{53}$$

Analogous constraints are also introduced to flow channels and flow pins, only with different pin notations. We omit the detailed constraints in this work due to the similarity.

3) Pin Binding Options

In this paragraph, we call a valve or a pump that requires individual control a *controlled unit*. Our module model provides multiple pin binding options for each controlled unit as mentioned in Table I. These options are derived from the following modeling features.

In our module models, each controlled unit can be accessed via several different pins. For example, as shown in Figure 2(a), v_0 can be accessed either via p_0 or p_1, and pump $\mathcal{P}$ can be accessed via p_{12}, p_{13}, p_{14}, or p_{15}. The pin access options of a controlled unit u_h in a module m_j depends on the inner-structure of m_j, which is specified in our module library. We denote the set of all pin access options of u_h in m_j as P_{m_j,u_h} and introduce the following constraints to make sure that each controlled unit is connected to exactly one incoming control channel.

$$\sum_{c_i\in C_{c_{\text{in}},m_j}}\ \sum_{p_{j,k}\in P_{m_j,u_h}} q_{c_i,p_{j,k}}=1. \tag{54}$$

In our model, a controlled unit is either the end of a pressure transportation path, or a transfer station of a pressure transportation path. Therefore, a controlled unit u_h in a module m_j can be connected to no more than one outgoing control channel of m_j. We model this feature as the following constraint:

$$\sum_{c_i\in C_{c_{\text{out}},m_j}}\ \sum_{p_{j,k}\in P_{m_j,u_h}} q_{c_i,p_{j,k}}\le 1. \tag{55}$$

For a controlled unit u_h in a mixer module m_j, there may exist pin access options $p_{j,k_1}, p_{j,k_2}\in P_{m_j,u_h}$ that are very close

to each other. For example, as shown in Figure 2(a), v_2 can be accessed via both p_4 and p_5. Since p_4 and p_5 are close to each other (the distance between p_4 and p_5 is 200μm), connecting v_2 to p_4 or p_5 will not make much difference in the routing results. In this case, we introduce a priority between p_4 and p_5 to shrink the solution space and thus save computation effort. For $p_{j,k_1}, p_{j,k_2} \in P_{m_j,u_h}$, if p_{j,k_1} has a higher priority than p_{j,k_2}, we introduce the following constraint:

$$\sum_{c_i \in C_{c_{\text{in}},m_j} \cup C_{c_{\text{out}},m_j}} q_{j,k_2} \leq \sum_{c_i \in C_{c_{\text{in}},m_j} \cup C_{c_{\text{out}},m_j}} q_{j,k_1}, \quad (56)$$

which means that a control channel connected to m_j can only access pin p_{j,k_2}, if p_{j,k_1} has already been accessed by other channels.

In our mixer and switch modules, some pin binding options serve similar functionality. For example, both $\{p_{j,12}, p_{j,14}\}$ and $\{p_{j,12}, p_{j,15}\}$ support the functionality to access the pump in m_j via two pins on opposite module boundaries. To shrink the solution space and thus save computation effort, we mark one of the two binding options (in this case $\{p_{j,12}, p_{j,15}\}$) as *insignificant*, and remove it from the model. For the pin binding options $\{p_{j,k_1}, p_{j,k_2}\}$ that are marked as insignificant, we introduce the following constraint:

$$\sum_{c_i \in C_{c_{\text{in}},m_j} \cup C_{c_{\text{out}},m_j}} q_{j,k_1} + \sum_{c_i \in C_{c_{\text{in}},m_j} \cup C_{c_{\text{out}},m_j}} q_{j,k_2} \leq 1. \quad (57)$$

In our switch modules, as mentioned in Section II, a flow channel junction can be directly accessed by exactly one of the flow pins on opposite module boundaries. We denote the flow pin pairs connected to the same flow channel junction of a switch m_j as $p'_{j,k}$ and $p'_{j,\bar{k}}$ and introduce the following constraint:

$$\sum_{c_i \in C_{f_{\text{in}},m_j} \cup C_{f_{\text{out}},m_j}} q_{j,k} + \sum_{c_i \in C_{f_{\text{in}},m_j} \cup C_{f_{\text{out}},m_j}} q_{j,\bar{k}} = 1. \quad (58)$$

4) Lagrangian Relaxation

In the pin allocation phase, control channel bundles are split into explicit control channels and thus require more chip area. In order to negotiate the area consumption of modules and channels, we allow the location of modules to be moved in a limited range compared with the global layout generated in the first phase. Moreover, we modify the constraint (20), which prohibits overlapping among rectangles r_a and r_b representing modules or channels, as follows:

$$q_1 + q_2 + q_3 + q_4 = 3 + \lambda_{a,b}, \quad (59)$$

thereof $\lambda_{a,b}$ is the $Lagrange\ multiplier$, the minimization of which is included in the optimization targets. If $\lambda_{a,b}$ is equal to 0, (59) is the same as (20), and one constraint among (16)-(19) has to be non-trivial, indicating a non-overlapping situation. If the overlapping is inevitable with the current channel models and module locations, $\lambda_{a,b}$ has to be set to 1. In this case, the pin allocation phase will be re-run and new bending points will be added to the channels with conflicts, thus enabling a detour around the conflict area. As shown in Figure 8(a), the initial states of channel c_1 and c_2 result in an inevitable crossing, considering their limited floating ranges. Therefore, we insert new bending points b_{1,k_1} to c_1 and b_{2,k_2} to c_2, the coordinates of which are constrained by the locations of all existing binding points along c_1 and c_2 as shown in Figure 8(a). With the new bending points, now the crossing can be avoided as shown in Figure 8(b).

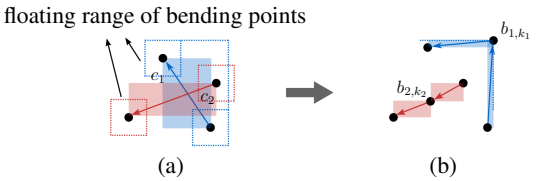


Figure 8: (a) An example of channel crossing. (b) An example of inserting new bending points.

5) Pin Allocation Model

We add the minimization of the Lagrange multiplier $\lambda_{a,b}$ into the optimization objective of the pin allocation model, and formulate the model as the following:

Minimize:

$$\alpha d_\tau + \alpha_1 x_\tau + \alpha_2 y_\tau + \beta \sum_{c_i \in C} \sum_{seg_{i,k} \in S_{c_i}} l_{i,k} + \xi \sum_{(r_a,r_b) \in L} \lambda_{a,b}$$

Subject to: (1)–(19), (46)–(59)

where α, α_1, α_2, β, and ξ are adjustable weight coefficients, and L represents the set of all nearby rectangles for modules and channels. The pin allocation phase terminates when the sum of Lagrange multipliers $\sum_{(r_a,r_b) \in L} \lambda_{a,b}$ is minimized to 0.

D. Inlet/Outlet Restoration

In the global layout generation phase and pin allocation phase, Columba omits the area consumption for flow and control inlets/outlets. Since inlets/outlets have a simple physical structure with only one pin and no valves, it is not so complicated to restore them in the inlet/outlet restoration phase.

In the inlet/outlet restoration phase, Columba restores the inlets/outlets by adding them to the non-overlapping constraints (16)-(19) and (59). Each inlet/outlet is modeled as a rectangle like other modules, and is prevented from overlapping with other modules and channels. The optimization objective, related constraints, as well as the termination control are kept the same as in the pin allocation phase.

E. Refinement

In previous phases, Columba focuses on finding a feasible solution. The modeling result of the inlet/outlet restoration phase is a valid design that fulfills all design rules, but can be further improved in the refinement phase by adjusting the weight coefficients of the optimization objetives. As in previous phases, modules are allowed to be moved in a limited range in this phase. The generated design will be compared with the design generated in the last phase or in the last iteration, and the refinement phase terminates when the improvement is less than a given threshold.

TABLE II: Generated design features.

application	ver.	$D(\text{mm}^2)$	$L(\text{mm})$	$\#(m,r,s,f_{\text{in/out}},c_{\text{in}})$	T
kinase	Columba1.0	15.05 × 15.05	163.54	2, 2, 3, 7, 24	5m22s
act. [24]	Columba2.0	17.20 × 15.00	174.65	2, 2, 3, 7, 15	1m14s
nucleic acid	Columba1.0	18.35 × 18.15	252.83	3, 3, 3, 11, 40	9m5s
proc. [3]	Columba2.0	16.30 × 24.55	257.50	3, 3, 3, 11, 17	5m20s
ChIP	Columba1.0	27.95 × 26.65	298.25	5, 4, 2, 17, 44	9m56s
(4IP) [25]	Columba2.0	14.20 × 43.50	261.55	5, 4, 2, 17, 26	4m25s
mRNA	Columba1.0	22.77 × 24.30	564.01	4, 4, 3, 18, 54	34m42s
iso. [26]	Columba2.0	29.10 × 21.25	388.55	4, 4, 3, 18, 23	23m49s
ChIP	Columba1.0	38.15 × 38.11	556.42	11, 10, 2, 23, 100	46m10s
(10IP)	Columba2.0	47.90 × 36.45	494.60	11, 10, 2, 23, 31	23m35s
cell-free	Columba1.0	n/a	n/a	8, 0, 2, 10, 75	> 10hr
bio. net. [29]	Columba2.0	20.00 × 37.00	477.05	8, 0, 2, 10, 24	10m32s

D: chip dimension. L: total length of channels. $\#m$, $\#r$, $\#s$, $\#f_{i/o}$, $\#c_i$: number of mixers, reaction chambers, switches, flow inlets/outlets, and control inlets. T: program runtime.

VI. Result Interpretion – AutoCAD Interface

After the physical synthesis, Columba interprets the modeling results and outputs a series of drawing commands into an AutoCAD script file, which can be directly read by AutoCAD and then exported for mask fabrication.

VII. Experimental Results

Columba is implemented in C++, and applies the Mixed Integer Linear Programming (MILP) solver Gurobi [27] to solve its mathematical model. We run Columba on a computer with 2.40 GHz CPU to perform automatic design for five applications.

The design rules we adopted for the test cases are from the Stanford Foundry [7]: the width of flow channels and valves is $100\mu m$, the width of control channels is $30\mu m$, the minimum spacing distance between channels is $100\mu m$, the size of a flow/control inlet is $1mm \times 1mm$, and the center-to-center distance between inlets/outlets is $2mm$.

Table II shows the feature values of the designs output by Columba. The first five test cases are from the the previous version of Columba [28] (Columba 1.0), and the last test case is new [29]. The first four test cases are based on the manual designs shown in [3], [24], [25], and [26], respectively, and the fifth test case is a synthetic chromatin immunoprecipitation (ChIP) application with 10 replicate immunoprecipitation (IP) operations, based on [25]. We record the dimension $D(\text{mm}^2)$, total channel length $L(\text{mm})$, and the number of different modules $\#(m,r,s,f_{\text{in/out}},c_{\text{in}})$ of the designs generated by the previous version of Columba [28] (Columba 1.0) and by the current version of Columba (Columba 2.0). The program runtime for generating these designs is denoted as T.

As shown in Table II, for each application, the designs output by Columba 1.0 and 2.0 have similar dimensions. In general, Columba 1.0 generates designs with more balanced x-y-dimensions, because Columba 1.0 does not support active pressure sharing among parallel modules. Parallel modules are required to be placed in alignment in Columba 2.0, and thus may require larger x- or y-dimension.

In most cases, Columba 2.0 achieves channel length reduction and significant control inlets reduction compared with Columba 1.0, mainly owing to the improved module models. Our new module models allow control channels to pass through modules via pins on opposite module boundaries, and thus avoid long detour of control channels and enable more pressure sharing options. Besides, active pressure sharing also contributes to the control inlets reduction.

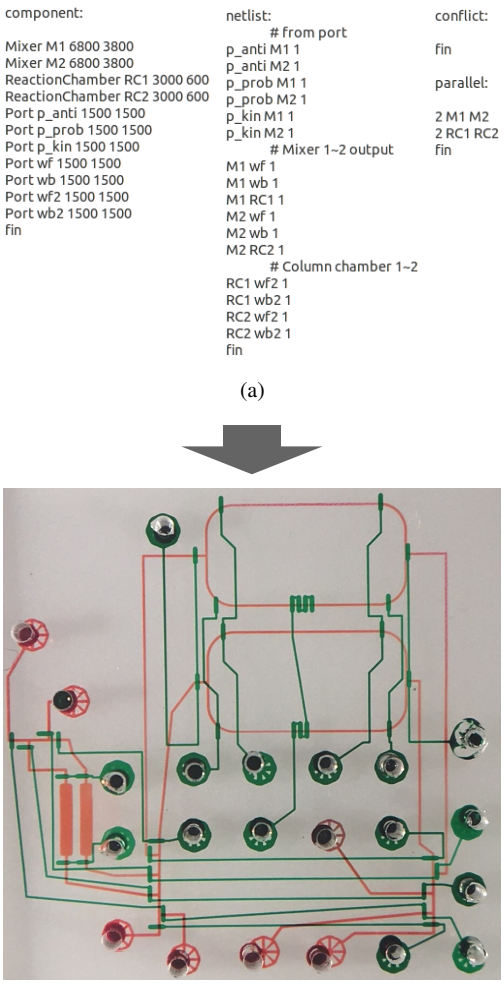


Figure 9: Input and output of Columba 2.0 for kinase activity application [24]. (a) Input primitive netlist. (b) Fabricated chip.

The program runtime of both Columba 1.0 and Columba 2.0 increases with the complexity of the input application. Compared with Columba 1.0, Columba 2.0 achieves significant runtime reduction, resulting mainly from three approaches: 1. The above-mentioned improvement of module models improves the routability of the design and reduces the time for finding feasible solutions. 2. In-alignment-placed parallel modules shrink the solution space. 3. The reduced number of control inlets alleviates the computational effort of the inlet/outlet restoration phase.

Columba is the first design automation tool that can seamlessly synchronize with the manufacturing flow. We show four synthesized designs, two of which have been fabricated, and discuss their properties from different perspectives.

Figure 9 shows the input primitive netlist and the output design for a kinase activity application [24]. The input primitive netlist (Figure 9(a)) looks like a high-level programming language consisting of three parts (columns): module declara-

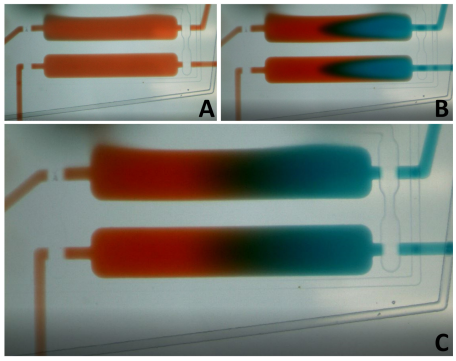


Figure 10: Demonstration of red and blue food dyes mixing in two reaction chambers. (a) $t=0s$. (b) $t=2s$. (c) $t=8s$.

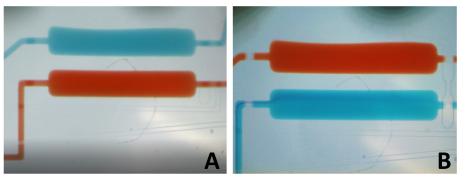


Figure 11: Demonstration of fluidic control using pneumatic valves. (a)(b) Red and blue dyes are filled alternately in the two reaction chambers.

tion, which specifies the type, name, and dimensions of each module; netlist description, which specifies the communication among modules; and special constraints/demands. With the primitive netlist and the design rules, Columba synthesizes the design completely automatically. Though this is the smallest design in our test cases and only involves two mixers and two chambers, it requires sophisticated valve control to support the application execution, and thus results in a complex control layer physical structure. Figure 9(b) shows a photo of the fabricated chip, mirrored from the design output by Columba. The flow layer of the chip is marked by red food dye, and the control layer of the chip is marked by green food dye.

Figures 10–12 demonstrate two essential functions: mixing and fluid routing, performed on our Columba-generated designs. Figure 10 demonstrates the mixing of red and blue food dyes in the reaction chambers on our microfluidic device. At $t=0s$ (Figure 10(a)) red dye filled both chambers, and the valves were closed. At $t=2s$ (Figure 10(b)) the right valves were opened. Blue dye was allowed to fill half of each chamber, and mixing occurred (Figure 10(c)). This function can be employed for biologically relevant studies requiring the reaction or incubation of two fluids. (See Supplementary Video "ChamberMixing.gif" [30])

Figure 11 demonstrates fluidic control using pneumatic valves. Both red and blue dyes were made to alternate between the top and bottom reaction chambers, showing the ability to control fluidic movement on-chip as required for different applications. (See Supplementary Video "FluidRouting.gif" [30])

Figure 12 demonstrates the function of our rotary mixer. At first, half of the mixer is filled with red dye and the other half is filled with blue dye (Figure 12(a)). Then the peristaltic pump is actuated to move fluids circularly around the mixer (Figure 12(b)). The mixing of two fluids is an integral step in many biochemical, biophysical and biomedical studies. (See

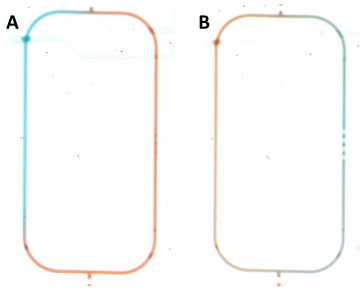


Figure 12: Demonstration of mixing function in the rotary mixer. (a) Before mixing. (b) During mixing.

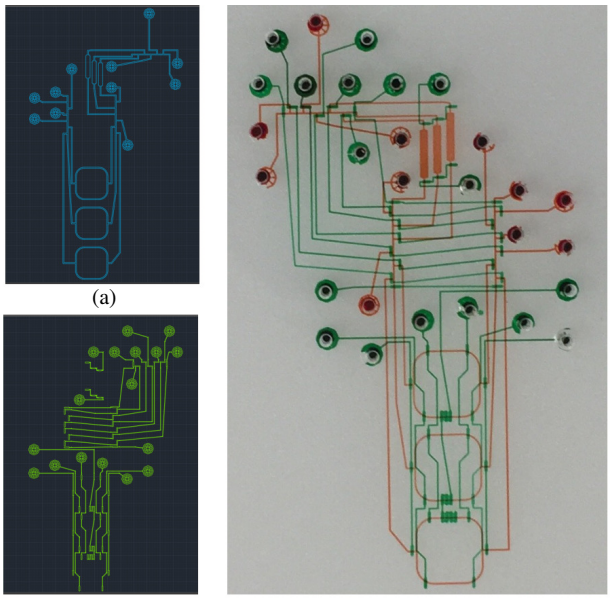
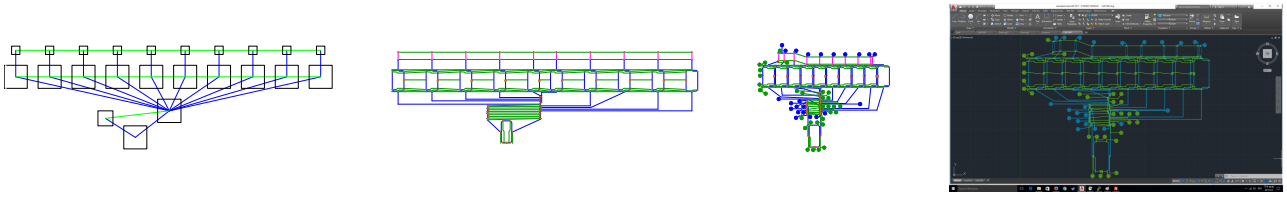


Figure 13: Design of a nucleic acid processor [3] synthesized by Columba 2.0. (a) Flow layer layout (b) Control layer layout (c) Fabricated chip.

Supplementary Video "RotaryMixerMixing.gif" [30])

Figure 13 shows the design of a nucleic acid processor [3] synthesized by Columba 2.0. The design contains three mixers and three reaction chambers. Different from the rectangular mixers applied in the kinase activity design, all mixer modules specified for the nucleic acid processor have the same x- and y-dimension. Columba provides a modification-friendly module model library that allows its user to define the dimensions of their required modules. Compared with the design synthesized by Columba 1.0, more than half control inlets are saved in the design synthesized by Columba 2.0.

We show the complete physical synthesis flow for our largest application ChIP (10IP) in Figure 14. The application requires one mixer for chromatin process (Ch) and ten parallel mixers as well as 10 parallel chambers for ten replicate immunoprecipitation processes (IP). Columba first determines the basic layout topology for this design in its global layout generation phase, as shown in Figure 14(a). In this phase, modules are simplified as square bounding boxes; channels are simplified as straight lines; control channels between the same modules are collected as control channel bundles; and inlets/outlets are omitted. The basic layout topology is then de-



(a) (b) (c) (d)

Figure 14: Resultant graphs of the design for ChIP (10IP). (a) Phase 1. (b) Phase 2. (c) Phase 3. (d) Phase 4 (final).

tailed in the pin allocation phase, as shown in Figure 14(b). In this phase, modules are restored as their original dimensions, and channels are modeled as polylines connected to pins on module boundaries. This design is further complemented in the inlet/outlet restoration phase, as shown in Figure 14(c). We can see from the figures that the chip area is gradually reduced as the design is detailed phase by phase. Figure 14(d) shows the final design after the refinement phase, which is zoomed in for a clear vision.

As the complexity of the biological application increases, the physical structure of the continuous-flow microfluidics becomes more complex, which leads to huge difficulty in manual design. Columba provides a design automation solution to solve this problem, and will be a milestone to the industrialization of continuous-flow microfluidics.

VIII. Conclusion and Future Possibilities

In this work, we proposed the co-layout synthesis tool Columba that can synthesize manufacturing-ready designs for continuous-flow microfluidic large-scale integration (mLSI) from plain-text netlist descriptions. Columba enables easy adaption to the continuous innovation of bioengineering by applying an extendable module model library. New microfluidic components and variants of existing microfluidic components can be easily modeled as modules with pin binding options, and thus fit into our proposed optimization model. In future work, we would synchronize Columba with front-end research [31] to enlarge its application scope. We aim to propose a complete design automation tool, to which the user can send either high-level abstract application protocols or low-level physical netlist descriptions as the inputs. We would also further analyze the difference between the manual designs and the computer-generated designs. We aim to run real applications on the designs output by Columba.

References

[1] R. A. W. III, P. C. Blainey, H. C. Fan, and S. R. Quake, "Digital PCR provides sensitive and absolute calibration for high throughput sequencing," *BMC Genomics*, 2009.

[2] X. Jiang, N. Shao, W. Jing, S. Tao, S. Liu, and G. Sui, "Microfluidic chip integrating high throughput continuous-flow PCR and DNA hybridization for bacteria analysis," *Talanta*, pp. 246–250, 2014.

[3] J. W. Hong, V. Studer, G. Hang, W. F. Anderson, and S. R. Quake, "A nanoliter-scale nucleic acid processor with parallel architecture," *Nature Biotechnology*, vol. 22, no. 4, pp. 435–439, 2004.

[4] F. K. Balagadde, L. You, C. L. Hansen, F. H. Arnold, and S. R. Quake, "Long-term monitoring of bacteria undergoing programmed population control in a microchemostat," *Science*, vol. 309, no. 5731, pp. 137–140, 2005.

[5] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "ILP-based alleviation of dense meander segments with prioritized shifting and progressive fixing in PCB routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 1000–1013, 2015.

[6] T.-M. Tseng, B. Li, C.-F. Yeh, H.-C. Jhan, Z.-M. Tsai, M. P.-H. Lin, and U. Schlichtmann, "An efficient two-phase ILP-based algorithm for precise CMOS RFIC layout generation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, accepted.

[7] Stanford Foundry, *Basic Design Rules*. http://web.stanford.edu/group/foundry.

[8] H. Yao, T.-Y. Ho, and Y. Cai, "Pacor: Practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips," in *Proc. Design Autom. Conf.*, 2015, pp. 142:1–142:6.

[9] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 1, pp. 55–68, 2017.

[10] W. T. Tutte, "How to draw a graph," *Proceedings of the London Mathematical Society*, vol. Third Series, no. 13, pp. 743–767, 1963.

[11] H. Yao, Q. Wang, Y. Ru, T.-Y. Ho, and Y. Cai, "Integrated flow-control co-design methodology for flow-based microfluidic biochips," *IEEE Des. Test. Comput.*, vol. 32, no. 6, pp. 60–68, 2015.

[12] T.-M. Tseng, B. Li, U. Schlichtmann, and T.-Y. Ho, "Storage and caching: Synthesis of flow-based microfluidic biochips," *IEEE Des. Test. Comput.*, vol. 32, no. 6, pp. 69–75, 2015.

[13] J. McDaniel, B. Crites, P. Brisk, and W. H. Grover, "Flow-layer physical design for microchips based on monolithic membrane valves," *IEEE Des. Test. Comput.*, vol. 32, no. 6, pp. 51–59, 2015.

[14] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proc. Design Autom. Conf.*, 2015, pp. 141:1–141:6.

[15] T.-M. Tseng, B. Li, M. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis with dynamic device mapping and fluid routing for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 12, pp. 1981–1994, 2016.

[16] R. Gomez-Sjoeberg, A. A. Leyrat, D. M. Pirone, C. S. Chen, and S. R. Quake, "Versatile, fully automated, microfluidic cell culture system," *Anal. Chem*, vol. 79, pp. 8557–8563, 2007.

[17] J. F. Zhong, Y. Chen, J. S. Marcus, A. Scherer, S. R. Quake, C. R. Taylor, and L. P. Weiner, "A microfluidic processor for gene expression profiling of single human embryonic stem cells," *Lab on a Chip*, vol. 8, no. 1, pp. 68–74, 2008.

[18] A. K. White, M. VanInsberghe, O. I. Petriv, M. Hamidi, D. Sikorski, M. A. Marra, J. Piret, S. Aparicio, and C. L. Hansen, "High-throughput microfluidic single-cell RT-qPCR," *Proc. Natl. Acad. Sci.*, vol. 108, no. 34, pp. 13 999–14 004, 2011.

[19] AUTODESK, *AutoCAD*. http://www.autodesk.com/products/autocad/overview.

[20] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga, "Architectural synthesis of flow-based microfluidic large-scale integration biochips," in *Proc. Int. Conf. Compil., Arch. and Syn. Embed. Sys.*, 2012, pp. 181–190.

[21] K.-H. Tseng, S.-C. You, J.-Y. Liou, and T.-Y. Ho, "A top-down synthesis methodology for flow-based microfluidic biochips considering valve-switching minimization," in *Proc. Int. Symp. Phy. Des.*, 2013, pp. 123–129.

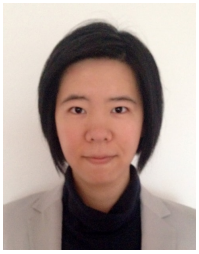
[22] C.-X. Lin, C.-H. Liu, I.-C. Chen, D. T. Lee, and T.-Y. Ho, "An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips," in *Proc. Design Autom. Conf.*, 2014, pp. 141:1–141:6.

[23] M. Li, T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations," in *Proc. Design, Automation, and Test Europe Conf.*, 2016.

[24] C. Fang, Y. Wang, N. T. Vu, W.-Y. Lin, Y.-T. Hsieh, L. Rubbi, M. E. Phelps, M. Mschen, Y.-M. Kim, A. F. Chatziioannou, H.-R. Tseng, and T. G. Graeber, "Integrated microfluidic and imaging platform for a kinase activity radioassay to analyze minute patient cancer samples," *Cancer Res.*, vol. 70, no. 21, pp. 8299–8308, 2010.

[25] A. R. Wu, J. B. Hiatt, R. Lu, J. L. Attema, N. A. Lobo, I. L. Weissman, M. F. Clarke, and S. R. Quake, "Automated microfluidic chromatin immunoprecipitation from 2,000 cells," *Lab on a Chip*, vol. 9, pp. 1365–1370, 2009.
[26] J. S. Marcus, W. F. Anderson, and S. R. Quake, "Microfluidic single-cell mRNA isolation and analysis," *Anal. Chem.*, vol. 78, pp. 3084–3089, 2006.
[27] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*. http://www.gurobi.com.
[28] T.-M. Tseng, M. Li, B. Li, T.-Y. Ho, and U. Schlichtmann, "Columba: Co-layout synthesis for continuous-flow microfluidic biochips," in *Proc. Design Autom. Conf.*, 2016, pp. 147:1–147:6.
[29] H. Niederholtmeyer, V. Stepanova, and S. J. Maerkl, "Implementation of cell-free biological networks at steady state," *Proc. Natl. Acad. Sci.*, vol. 110, no. 40, pp. 15 985–15 990, 2013.
[30] IEEE Xplore, *IEEE Xplore Digital Library*. http://ieeexplore.ieee.org.
[31] M. Li, T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Component-oriented high-level synthesis for continuous-flow microfluidics considering hybrid-scheduling," in *Proc. Design Autom. Conf.*, 2017, pp. 51:1–51:6.

Tsun-Ming Tseng (S'10–M'15) received the bachelor degree in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2010, the master degree in communications engineering from Technical University of Munich (TUM), Munich, Germany, in 2013, and the Dr.-Ing. degree from TUM, in 2017. He is currently a postdoc fellow in the Chair of Electronic Design Automation, TUM. His research interests focus on mathematical modeling methods for computer-aided design for emerging technologies, such as microfluidic biochips and optical networks-on-chips.



Mengchu Li (S'16) received the bachelor degree in Germanistik from Tongji University in China. She then came to Germany and is currently pursuing another Bachelor degree in Computer Science at Ludwig-Maximilians-Universität München (LMU) in Germany. She works as working student in the Chair of Electronic Design Automation in Technical University of Munich (TUM) in Germany. Her current research interests focus on design automation for continuous-flow microfluidics.

Daniel Nestor Freitas is currently pursuing his M.S. in Electrical Engineering at Santa Clara University, where he has previously received his B.S. in Bioengineering (2017). Daniel works under the guidance of Dr. Emre Araci, designing, fabricating and testing integrated microfluidic devices. Daniel has been recognized as an associate member of the Sigma Xi Research Society, has been supported by funding from numerous successful grant applications as an undergraduate, and was selected as the Santa Clara University School of Engineering 2017 recipient of the excellence in research award. Daniel's research interests include microfluidic large-scale integration (mLSI), and the development of microfluidic technologies that solve common problems with traditional laboratory techniques.

Travis McAuley received M.S.'17 and B.S'16 degrees from Santa Clara University in Bioengineering. He worked under the guidance of Dr. Emre Araci, designing, fabricating and testing integrated microfluidic devices. He is currently working for Motive Power, Inc at Santa Clara, California.

Bing Li received the bachelor's and master's degrees in communication and information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2000 and 2003, respectively, and the Dr.-Ing. degree in electrical engineering from Technical University of Munich (TUM), Germany, in 2010. He is currently a researcher with the Chair of Electronic Design Automation, TUM. His research interests include high-performance and lower-power design of computing systems and emerging systems.

Tsung-Yi Ho (SM'12) received his Ph.D. in Electrical Engineering from National Taiwan University in 2005. He is a Professor with the Department of Computer Science of National Tsing Hua University, Hsinchu, Taiwan. His research interests include design automation and test for microfluidic biochips and nanometer integrated circuits. He has presented 10 tutorials and contributed 11 special sessions in ACM/IEEE conferences, all in design automation for microfluidic biochips. He has been the recipient of the Invitational Fellowship of the Japan Society for the Promotion of Science (JSPS), the Humboldt Research Fellowship by the Alexander von Humboldt Foundation, and the Hans Fischer Fellow by the Institute of Advanced Study of the Technical University of Munich. He was a recipient of the Best Paper Awards at the VLSI Test Symposium (VTS) in 2013 and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems in 2015. He served as a Distinguished Visitor of the IEEE Computer Society for 2013-2015, the Chair of the IEEE Computer Society Tainan Chapter for 2013-2015, and the Chair of the ACM SIGDA Taiwan Chapter for 2014-2015. Currently he serves as an ACM Distinguished Speaker, a Distinguished Lecturer of the IEEE Circuits and Systems Society, and Associate Editor of the ACM JETC, ACM TODAES, ACM TECS, IEEE TCAD, and IEEE TVLSI, Guest Editor of IEEE D&T, and the Technical Program Committees of major conferences, including DAC, ICCAD, DATE, ASP-DAC, etc.

Ismail Emre Araci (M'14) I. Emre Araci has received his B.S. and M.S. degrees from Electrical Engineering Department, Ege University, Izmir, Turkey. He received his Ph.D. from College of Optical Sciences, University of Arizona in 2010. He then joined Prof. Stephen Quake's group in the Bioengineering Department at Stanford University as a postdoctoral associate and as the director of the Stanford Microfluidics Foundry (SMF). He has been an Assistant Professor at the Bioengineering Department, Santa Clara University, since 2015. He serves on the advisory board of several start-up companies. His primary research goals are directed toward the development and application of miniaturized microfluidic and optofluidic technologies for biology and medicine.

Ulf Schlichtmann (S'88–M'90) received the Dipl.Ing. and Dr.Ing. degrees in electrical engineering and information technology from Technical University of Munich (TUM), Munich, Germany, in 1990 and 1995, respectively. He was with Siemens AG, Munich, and Infineon Technologies AG, Munich, from 1994 to 2003, where he held various technical and management positions in design automation, design libraries, IP reuse, and product development. He has been a Professor and the Head of the Chair of Electronic Design Automation, TUM, since 2003, where he also served as Dean of the Department of Electrical and Computer Engineering, from 2008 to 2011, and as Associate Dean of Studies of International Studies since 2013. Since 2016, he is an elected member of TUM's Academic Senate and Board of Trustees. He also serves on a number of advisory boards. Ulf's current research interests include computer-aided design of electronic circuits and systems, with an emphasis on designing reliable and robust systems. In recent years, he has increasingly worked on emerging technologies, such as microfluidic biochips and optical interconnect.