Columba S: A Scalable Co-Layout Design Automation Tool for Microfluidic Large-Scale Integration

Tsun-Ming Tseng[†], Mengchu Li[†♯], Daniel Nestor Freitas[▽], Amy Mongersun[▽], Ismail Emre Araci[▽], Tsung-Yi Ho[⋆◇], and Ulf Schlichtmann[†]

{tsun-ming.tseng, ulf.schlichtmann}@tum.de, mengchu.li@campus.lmu.de, {dnfreitas, amongersun, iaraci}@scu.edu, tyho@cs.nthu.edu.tw

[†]Chair of Electronic Design Automation, Technical University of Munich, Arcisstraße 21, 80333 München, Germany

[♯]Ludwig-Maximilians-Universität München, Geschwister-Scholl-Platz 1, 80539 München, Germany

[▽]Department of Bioengineering, Santa Clara University, 500 El Camino Real, Santa Clara, CA 95053, United States

[⋆]Department of Computer Science, National Tsing Hua University, No. 101, Section 2, Kuang-Fu Road, 30013 Hsinchu, Taiwan

[◇]Institute for Advanced Study, Technical University of Munich, Lichtenbergstraße 2 a, 85748 Garching, Germany

ABSTRACT

Microfluidic large-scale integration (mLSI) is a promising platform for high-throughput biological applications. Design automation for mLSI has made much progress in recent years. Columba and its succeeding work Columba 2.0 proposed a mathematical modeling method that enables automatic design of manufacturing-ready chips within minutes. However, current approaches suffer from a huge computation load when the designs become larger. Thus, in this work, we propose Columba S with a focus on scalability. Columba S applies a new architectural framework and a straight channel routing discipline, and synthesizes multiplexers for efficient and reconfigurable valve control. Experiments show that Columba S is able to generate mLSI designs with more than 200 functional units within three minutes, which enables the design of a platform for large and complex applications.

1 Introduction

Microfluidic large-scale integration (mLSI) is an emerging platform for biochemical applications. Compared with other lab-on-a-chip technologies, mLSI shows advantages in performing precise control for complex fluid manipulations [1], and is widely used in advanced biological applications such as single cell capturing [2] and protein analysis [3].

An mLSI chip usually consists of a flow layer for fluid transportation and a control layer for pressure transportation. The precise on-chip control of mLSI is supported by micro-valves, which are constructed with channel segments from both layers. When pressure is transported to the control segment of a valve, it will lead to a shape change of the membrane between the two layers, and thus block the corresponding flow segment. In this manner, by implementing valves at target channels that require fluid control, complex functional units can be formed for various operations such as mixing [3], washing [4], and heating [5]. With the development of mLSI manufacturing technology, valve density can currently achieve 1 million valves per cm^2 [6], which makes large-scale parallel applications possible.

Currently, most mLSI chips are designed manually. However, as the application complexity increases, design complexity increasingly challenges human capabilities. Despite the advanced manufacturing technology, most manual mLSI designs only consist of tens of functional units or fewer [3] [7] [8], and large manual mLSI designs are usually fit into homogenous matrix-like structures that only support simple application protocols [9] [10].

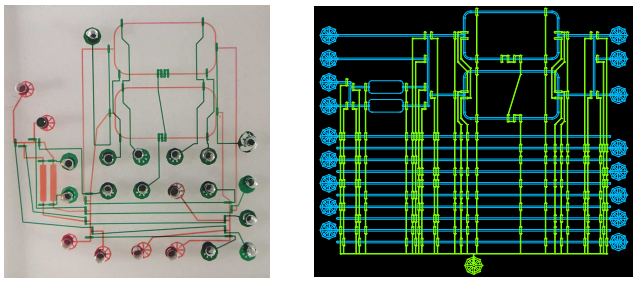
The mLSI design problem can be modeled as two interacting single-layered place-and-route problems [11] [12]. The flow layer design requires arranging the functional units in a compact manner while ensuring the channel connection between certain functional units and fluid inlets. For complex designs, flow channel crossings are sometimes inevitable. In this case, designers also need to minimize the number of channel crossings and implement extra switches for fluid guidance. The placement of functional units and switches will decide the location of valves, which leads to interactions with the control layer design. Since control channels are strictly prohibited from overlapping, the routability of the control layer is highly dependent on the flow layer, which means that the two single-layered problems cannot be solved separately. Besides the basic design rules, mLSI design must also take the usage of pressure inlets into consideration, the number of which is limited due to their large area consumption and reliability concerns [13].

Design automation approaches have been proposed in recent years to aid [14] [15] or replace [16] manual design. Columba [11] and its succeeding work Columba 2.0 [12] proposed the first design automation tool that can synthesize manufacturing-ready mLSI designs without human intervention. The Columba works applied a mathematical modeling method that described the place-and-route problems and the valve control behaviors comprehensively, which ensured their output design to meet all design rules even for delicate applications. However, their capability is limited by the computing power when dealing with heterogeneous designs with hundreds of functional units. Thus, the demand for a design automation tool that matches the mature mLSI manufacturing technology is still pressing.

In this work, we propose *Columba S* ("S" stands for "scalable"), which drastically speeds up the automated design process, and is especially suitable for very large-scale designs.

Figure 1 provides an intuitive comparison between Columba (2.0) and Columba S. Figure 1(a) shows a chip fabricated from a kinase activity [17] design generated by Columba 2.0 [12], where flow channels are filled with red dye and control channels are filled with green dye. Figure 1(b) shows the design generated by Columba S for the same application, where flow channels are in blue and control channels are in green. Compared to Columba 2.0, Columba S shows

DAC 2018, DOI: 10.1145/3195970.3196011
http://ieeexplore.ieee.org/document/8465905
https://dl.acm.org/citation.cfm?id=3196011



(a) (b)

Figure 1: Designs generated by (a) Columba 2.0 (b) Columba S. Program run time (seconds): (a) 56 (b) 0.9. Number of inlets: (a) 22 (b) 18. Flow channel length (mm) (a) 58.9 (b) 39.85 (in the functional region).

advantages in the following aspects:

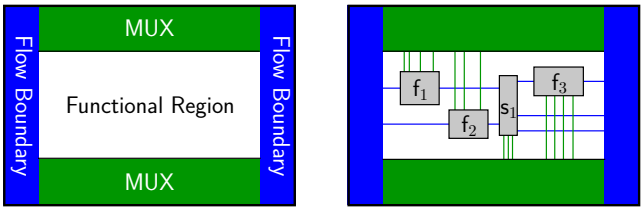
- **Run time reduction**. Columba S supports a new architectural framework and a straight channel routing discipline, which significantly reduces the problem space and thus alleviates the computation load. Experiments show that Columba S speeds up the design process by tens to thousands of times, and is capable of generating designs that contain more than 200 functional units within three minutes.
- **Inlets reduction and reconfigurability**. Columba S applies multiplexers for valve control. Each multiplexer guarantees the control of n independent valves with $2\lceil\log_2(n)\rceil+1$ inlets [9]. The efficient inlets usage allows Columba S to support individual control of independent valves regardless of their working periods. Thus, the same design is adaptable for different scheduling protocols. Columba 2.0 applies a pressure sharing technology to reduce the inlet usage, the performance of which is less significant and the design does not adapt to changes of the application scheduling protocols.
- **Flow channel length reduction**. Columba S centralizes the functional units and routes the fluid transportation paths without detour, which benefits both the application execution time [18] and the chip reliability [19]. The new layout also allows a better overview of the application execution and brings convenience to microscope observation.

2 Architectural Framework & Channel Routing Discipline

Columba S distinguishes itself from the previous Columba works with its ultrahigh efficiency, which provides the foundation for automated large-scale designs. This efficiency is contributed by its new architectural framework and channel routing discipline.

Figure 2(a) illustrates *the Columba S architectural framework*, which consists of a functional region surrounded by two opposite flow boundaries and (at most) two opposite multiplexers (MUXs). The functional region is the core of the design that is responsible for all fluid transportation and manipulations, while the flow boundaries and the multiplexers can be regarded as abstract chip boundaries that support flexible locations of fluid and pressure inlets, respectively.

Under this framework, Columba S focuses on the functional region design. The functional region is constructed with modules and channels. A "module" is a rectangular box that defines the physical layout inside and around a functional unit or a switch. Modules are connected to each other and to chip boundaries via channels. Columba S applies a



(a) (b)

Figure 2: (a) Our architectural framework. (b) An initial design fit to our framework.

straight channel routing discipline where *all control channels are routed vertically and all flow channels are routed horizontally*. Figure 2(b) shows an initial design fit to this framework, where f_1, f_2 and f_3 represent the modules of three functional units and s_1 represents the module of a switch. With the proposed routing discipline, all control channels are parallel to one another and directly extend to chip boundaries, while all flow channels are either in parallel or extend in the same direction. Thus, the computation effort for arranging channel detour and preventing channel crossing can be saved.

The implementation of our architectural framework and channel routing discipline is based on two prerequisites:

1. A module model library that supports horizontal flow channel access and vertical control channel access to all modules;
2. A multiplexing technology to pressurize/depressurize every control channel individually with a small number of inlets.

2.1 Columba S Module Model Library

The *module* concept is first proposed in [11] to synchronize the interaction between the control and the flow layers. As mentioned in Section 1, since the control and the flow layers interact with each other through valves, the two single-layered design problems cannot be solved separately. Thus, for important microfluidic components that contain valves, [11] modeled them as rectangular boxes (modules) that can be accessed via pins at their boundaries, and thereby transformed the layer interaction problem into a pin selection problem with less design difficulty. A module defines the inner-structure as well as the corresponding channel routing patterns of a microfluidic component. [11] proposed a module model library that provided several options for module modification and rotation. This library was then extended in [12] to include more options for channel routing, which allowed control channels to pass through modules to reduce the routing detour. Based on [12], we propose a modified module model library for Columba S to support its new architectural framework and channel routing discipline.

The Columba S module model library contains three types of modules: mixers, reaction chambers, and switches. Compared to [12], we remove the inlet modules from the Columba S library, since the new architectural framework excludes the inlets from its functional region. Some channel routing options that were provided by Columba 2.0 but had been rarely chosen in the automated design process are also removed to save the computational effort. The support of sieve valves and celltraps is added to the module model of mixers to support advanced operations. And we enlarge the spacing of pumping valves to resolve the manufacturing concern. To guarantee the new channel routing discipline, we prohibit the rotation of the modules in the new library, and re-design the module model for switches.

Figure 3(a) shows the modified module model for mixers, where control and flow channels are indicated by green and

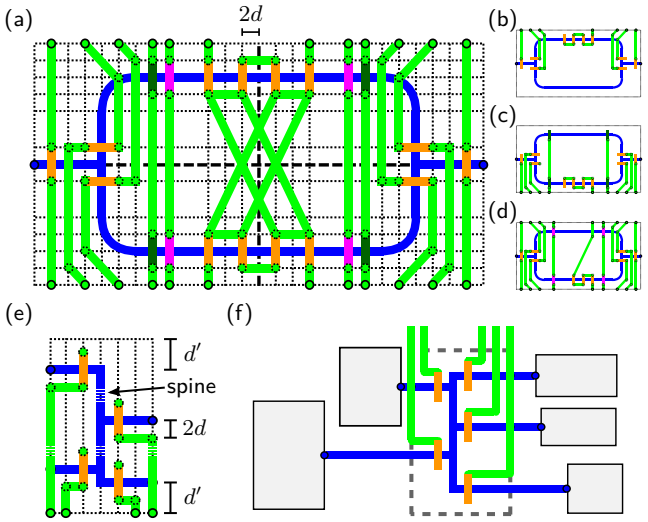


Figure 3: (a) The module model for rotary mixers. d represents the minimum channel spacing distance ($100\mu m$). (b)(c)(d) Different mixer configurations. (e) The module model for switches providing valve access from the bottom. d' is set to $750\mu m$ to prevent the overlapping of fluid inlets in the flow boundaries. (f) An exemplary design.

blue lines, respectively, and the orange/purple/dark green rectangles above control channels indicate different kinds of valves. As shown in the figure, flow channels inside a mixer can be accessed from the horizontal directions via pins on the left and the right module boundaries, and valves inside a mixer can be accessed via control channels from the top, from the bottom, or from the both vertical directions, as shown in Figure 3(b), (c) and (d), respectively. Additionally, the mixer in (c) contains four sieve valves to support washing operations [20], and the mixer in (d) contains four separation valves to support cell-capturing operations [18].

Figure 3(e) shows the new module model for switches. Switches are managed flow channel crossings consisting of one flow channel spine and several flow channel junctions, where fluids can be guided to their expected directions by valves. The switch model proposed in [12] had a fixed inner-structure, where the distance between two adjacent flow channel junctions was set to a constant value. We abandon this setting in Columba S and allow the flow channel spine to extend in the vertical direction (indicated by the dash lines in Figure 3(e)) to adapt the flow channel access from different locations. To support the vertical control channel routing discipline, we move the pins for valve access to the top and the bottom module boundaries and route the control channels accordingly. Figure 3(e) and (f) show the cases that the valves are accessed from the bottom and from the top, respectively. Figure 3(f) also illustrates an exemplary design containing five modules that require pairwise flow channel connection, which is arranged by a switch. As shown in the figure, all flow channels directly extend from the module in the horizontal direction to the switch, thereby forming the required channel junctions without detour.

2.2 Multiplexer Design

Under the Columba S architectural framework, all independent valves have direct access to chip boundaries, from where they can be individually pressurized/depressurized with multiplexers. Columba S applies the multiplexing technology proposed in [9] and [21], and the multiplexer design follows the physical synthesis results of the functional region.

We illustrate our multiplexing approach in Figure 4, where 15 control channels (indicated by green lines) are extended straightforwardly from the functional region to the multi-

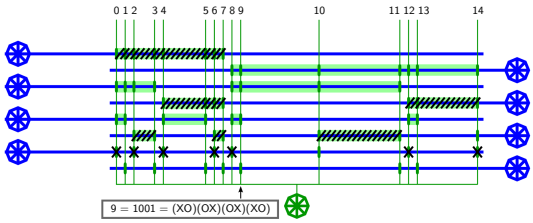


Figure 4: Multiplexer design and multiplexing function.

plexer. Our multiplexer uses pressurized flow channels (indicated by blue lines) to carry out the multiplexing process. To distinguish the flow channels in the multiplexers from the flow channels in the functional region, we refer to them as MUX-flow channels in the rest of this section. As shown in Figure 4, the MUX-flow channels overlap with the control channels and form valves (indicated by green rectangles) at some of the overlapping area. By pressurizing the MUX-flow channels to a certain pressure level, valves along the pressurized MUX-flow channels will be inflated (indicated by black slashes) and thus block the pressure transportation in the corresponding control channels. The explicit multiplexing strategy is described as follows:

1. Each control channel is indexed with a $\lceil \log_2(n) \rceil$-bit binary number, where n represents the total number of control channels. In particular, each of the 15 control channels in Figure 4(a) is indexed with a 4-bit binary number, and the 9th control channel is indexed with 1001.
2. The pressure level of the valves along each MUX-flow channel is denoted with a character, where O represents that the valves along the channel are not inflated, and X represents that the valves along the channel are inflated.
3. Each control-channel bit is assigned by a pair of MUX-flow channels: a MUX-flow channel pair configured to OX sets the corresponding bit to 0, and a MUX-flow channel pair configured to XO sets the corresponding bit to 1. In particular, the 4-bit index in Figure 4 is set by 4 MUX-flow channel pairs, which are configured to XO, OX, OX and XO, respectively, and thus forming a 4-bit binary number 1001. Under this configuration, all control channels other than the channel indexed as 1001 are blocked by inflated valves, ensuring the 9th control channel to be the only pressure transportation path.

In this manner, our multiplexer guarantees the control of n independent valves with $2\lceil \log_2(n) \rceil + 1$ inlets. Compared with the pressure sharing techniques proposed in [12], the inlets usage of Columba S is more predictable and does not rely on the application scheduling protocols. A sacrifice made by Columba S to achieve the very efficient inlets usage is that it only supports simultaneous control of at most two independent valves, while Columba 2.0 supports simultaneous control of all independent valves. Since current technology supports rapid valve actuation (10ms for each valve [22]), and the pressure level of a valve can be sustained for over 10 minutes [1] despite the gas-permeable [23] nature of PDMS, this sacrifice is acceptable for application execution.

3 The Columba S Design Flow

Columba S takes a plain-text netlist as its input, and outputs a manufacturing-ready design. Figure 5 shows the overall flow of mLSI production supported by Columba S.

3.1 Preparation: Netlist Planarization

The input of Columba S is a plain-text file specifying the number, type, and logic connection of the required functional units, which we call a *netlist description*. To implement the required logic connection without introducing channel conflicts, the primitive netlist needs to be planarized [11] [12]. Columba S applies the *netlist planarization* approach

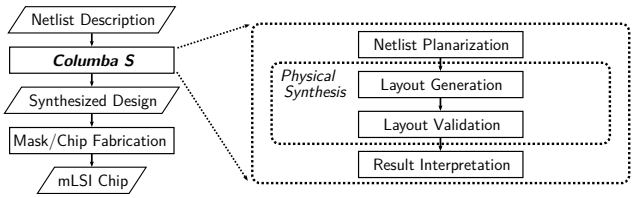


Figure 5: The overall flow of Columba S.

proposed in [12], i.e. adding switches to the netlist and refining the logic connection accordingly. Details of this approach can be found in [12] and are thus omitted in this paper.

3.2 Physical Synthesis

With the planarized netlist, Columba S constructs integer-linear-programming models to carry out the physical synthesis. The mLSI chip structure is modeled as points and lines in a 2D coordinate plane, and the design constraints are modeled as linear formulas. The synthesis process consists of two phases: the *layout generation* phase focuses on the design of the functional region, which roughly decides the location of all modules and channels; and the *layout validation* phase completes the design with explicit module placement, channel routing, and chip boundary restoration, which includes mostly engineering efforts.

In the following, we introduce some expressions and constraints applied in both phases:

Module, Channel and Chip Area The fundamental building blocks in our models are modules and channels, the locations of which are both modeled as rectangular boxes. If we denote the set of all modules and channels as $\mathcal{R}$, then for each $r_{i\in\mathbb{N}}\in\mathcal{R}$, we introduce four non-negative continuous variables v_{r_i,x_l}, v_{r_i,x_r}, v_{r_i,y_t}, and v_{r_i,y_b} to represent the x-coordinates of the left and right module boundaries, and the y-coordinates of the top and bottom module boundaries, respectively. If we denote the width and length of r_i as w_{r_i} and l_{r_i}, the boundaries can be constrained as:

$$v_{r_i,x_r}=v_{r_i,x_l}+w_{r_i}\ \wedge\ v_{r_i,y_t}=v_{r_i,y_b}+l_{r_i}. \tag{1}$$

If r_i is a *mixer* or a *reaction chamber*, w_{r_i} and l_{r_i} will be specified as a constant defined in the netlist description.

If r_i is a *switch*, w_{r_i} can be calculated as:

$$w_{r_i}=4d+c_{r_i}\cdot 2d,$$

where d represents the minimum spacing distance between channels, and c_{r_i} represents the number of flow channel junctions that r_i contains, which is specified in the planarized netlist. Since a switch is allowed to extend in the y-direction, as mentioned in Section 2.1, l_{r_i} will remain undefined and thus v_{r_i,y_t} and v_{r_i,y_b} will not be constrained initially.

If r_i is a *control channel*, w_{r_i} will be specified as $2d$, and l_{r_i} will remain undefined since control channels extend in the y-direction. On contrast, if r_i is a *flow channel*, l_{r_i} will be specified as $2d$, and w_{r_i} will remain undefined since flow channels extend in the x-direction.

We then denote the x- and y-dimension of a chip as continuous variables $v_{x_{\max}}$ and $v_{y_{\max}}$, and confine all $r_i\in\mathcal{R}$ to the chip by introducing the following constraints:

$$0\le v_{r_i,x_l}\ \wedge\ v_{r_i,x_r}\le v_{x_{\max}}\ \wedge\ 0\le v_{r_i,y_b}\ \wedge\ v_{r_i,y_t}\le v_{y_{\max}} \tag{2}$$

Non-Overlapping Constraints Since our rectangles take the minimum spacing distance d between channels into account, we allow two $r_i,r_j\in\mathcal{R}$ to be placed next to each other as long as their inner areas do not overlap (except for flow and control channels, which are allowed to overlap since they belong to different layers). Thus, we have four options for the relative location of r_i with respect to r_j: left, right, bottom, and top. The following constraints are proposed to describe these options:

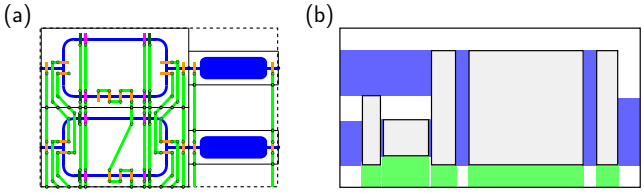


Figure 6: (a) A rectangle consisting of parallel sequential modules. (b) Synthesis results of the layout generation model for the design shown in Figure 1(b).

$$v_{r_i,x_r}\le v_{r_j,x_l}+q_1M\ \wedge\ v_{r_j,x_r}\le v_{r_i,x_l}+q_2M, \tag{3}$$
$$v_{r_i,y_t}\le v_{r_j,y_b}+q_3M\ \wedge\ v_{r_j,y_t}\le v_{r_i,y_b}+q_4M, \tag{4}$$

where q_1,q_2,q_3 and q_4 are auxiliary binary variables and M is a very large constant. If an auxiliary variable is set to 1, the corresponding inequation will become a tautology, but if an auxiliary variable is set to 0, the corresponding inequation will decide the relative location of r_i and r_j. We then introduce the following constraint to choose exactly one of these options:

$$q_1+q_2+q_3+q_4=3, \tag{5}$$

3.2.1 Layout Generation

This phase of the flow focuses on the functional region.

Parallel Functional Units Parallelization is an important feature in mLSI design [1]. By connecting multiple functional units with common control channels, identical valve actuation sequences can be processed in these units in parallel, which lays a foundation for high-throughput applications. In our layout generation model, we merge the rectangles of parallel functional units into a single large rectangle to omit their inter-connection and thus save the computational effort. Figure 6(a) shows a rectangle in which two pairs of sequentially connected mixers and reaction chambers share their control channels to work in parallel. The size of the new rectangle is set to the minimum size that can accommodate the original rectangles of the parallel units.

Channel Merge For model reduction, we merge channels into a single rectangle under one of the following conditions:

1. Suppose r_m is a rectangle that contains valves (i.e. that is not a channel), we merge all control channels connected to r_m as a new rectangle r_c, and set $w_{r_c}=w_{r_m}$;
2. Suppose $r_{m'}$ is a rectangle that contains more than one functional unit, we merge all flow channels connected to the same boundary of $r_{m'}$ as a new rectangle r_f, and set $l_{r_f}=l_{r_{m'}}$;
3. Suppose r_s is a rectangle that contains a switch and requires access to fluid inlets, we merge all flow channels that connect r_s to a flow boundary into a new rectangle r_{f_s}, and set $l_{f_s}=n\cdot d'$, where n is the number of flow channels, and d' is a constant that prevents potential overlapping of fluid inlets in the flow boundary.

Figure 6(b) shows the synthesis results of the layout generation model for the design shown in Figure 1(b): blue rectangles are merged from flow channels and green rectangles are merged from control channels.

Channel to Chip Boundaries We introduce the following constraints to connect all rectangles r_i/r_j that require access to fluid/pressure inlets to flow/MUX boundaries:

$$v_{r_i,x_l}\le 0+q_5M\ \wedge\ v_{r_i,x_r}\le v_{x_{\max}}+q_6M, \tag{6}$$
$$v_{r_i,x_l}\ge 0-q_5M\ \wedge\ v_{r_i,x_r}\ge v_{x_{\max}}-q_6M, \tag{7}$$
$$q_5+q_6=1, \tag{8}$$

Table 1: Design features comparison between Columba 2.0 and Columba S.

App.		Dimension: $v_{x_{max}} * v_{y_{max}}$ (mm^2)			L_f(mm)			# c_{in}			Run time(sec)		
		Columba	Columba S		Col.	Columba S		Col.	Columba S		Col.	Columba S	
ref	#u	2.0	1-MUX	2-MUX	2.0	1-MUX	2-MUX	2.0	1-MUX	2-MUX	2.0	1-MUX	2-MUX
[8]	6	19.40*23.15	19.80*27.45	19.80*34.20	135.1	77.05 (-43%)	78.45 (-42%)	17	13 (-24%)	20 (+18%)	309.1	0.8	0.6
[3]	9	14.20*41.50	28.00*30.75	28.00*39.00	152.2	114.2 (-25%)	113.1 (-26%)	26	13 (-50%)	22 (-15%)	299.2	0.7	0.9
[7]	8	28.55*23.95	22.20*29.65	22.20*37.90	219.5	146.85 (-33%)	147.25 (-33%)	23	13 (-43%)	22 (-4%)	705.1	0.7	0.9
[12]	21	27.10*57.70	29.60*57.25	29.60*64.00	315.1	172.25 (-45%)	172.25 (-45%)	31	13 (-58%)	20 (-35%)	749.8	1.5	1.5
\	129	\	132.60*174.95	79.80*184.70	\	3916.6	2096	\	17	28	\	71.9	72.7
\	257	\	145.40*322.15	92.60*333.40	\	8338.65	4827.4	\	17	30	\	156.2	157.7

L_f: length of flow channels. #c_{in}: number of control inlets. #u: number of functional units.

$$v_{r_j,y_b} \le 0 + q_7 M \ \wedge\ v_{r_j,y_t} \le v_{y_{\max}} + q_8 M, \quad (9)$$

$$v_{r_j,y_b} \ge 0 - q_7 M \ \wedge\ v_{r_j,y_t} \ge v_{y_{\max}} - q_8 M, \quad (10)$$

$$q_7 + q_8 = 1, \quad (11)$$

where q_5,q_6,q_7 and q_8 are auxiliary binary variables and M is a very large constant. If an auxiliary variable is set to 1, the corresponding two inequations will become tautology, but if an auxiliary variable is set to 0, the corresponding inequations will decide the boundary that the rectangle is connected to. Note that if the number of multiplexers is specified as 1 in the netlist, we will set $q_7 \equiv 0$ and $q_8 \equiv 1$ and thus r_j must be connected to the bottom MUX boundary.

Channel to Modules For flow/control channels connected to modules, we introduce similar constraints as (6)–(11) to ensure that the rectangles of the corresponding channels and modules share one of their boundaries in the y/x direction.

Switch Boundaries As mentioned in Section 2.1, the flow channel spine of our switch module can extend in the vertical direction. We set the y-coordinates of the top and bottom boundaries of a switch rectangle r_s to be dependent on the channel rectangles $r_{c_1},\cdots,r_{c_n}$ connected to this switch:

$$\forall 1 \le i \le n, \quad v_{r_s,t} \ge v_{r_{c_i},t} \ \wedge\ v_{r_s,b} \le v_{r_{c_i},b}. \quad (12)$$

Minimization Objective The optimization objective of the layout generation model is to minimize the dimension and the total channel length of the functional region. The dimension can be denoted by $v_{x_{\max}}$ and $v_{y_{\max}}$ as introduced earlier, and the channel length can be calculated as

$$l_{\text{total}} = \sum n_{r_f}(v_{r_f,x_r} - v_{r_f,x_l}) + \sum n_{r_c}(v_{r_c,y_t} - v_{r_c,y_b}), \quad (13)$$

where r_f and r_c represent the rectangles for flow and control channels, and n_{r_f} and n_{r_c} are constants denoting the number of channels contained in the corresponding rectangles. Thus, the minimization objective can be set as:

$$\alpha v_{x_{\max}} + \beta v_{y_{\max}} + \gamma v_{xy_{\max}} + \kappa l_{\text{total}},$$

where α, β, γ, and κ are adjustable weight coefficients, and $v_{xy_{\max}}$ is a continuous variable set as $\max\{v_{x_{\max}}, v_{y_{\max}}\}$ to balance the x-y dimension.

3.2.2 Layout Validation

This phase of the flow takes the synthesis results of the layout generation model as its inputs.

In this phase, we restore the original models for modules and channels, and synthesize the physical layout of the flow boundaries as well as the multiplexers. Fluid inlets are synthesized along the flow boundaries that have channel access, and multiplexers are synthesized along the MUX boundaries in the manner introduced in Section 2.2.

During the restoration, the location of most modules and channels can be directly calculated from the location of the rectangles specified by the earlier optimization phase. A similar restoration approach has been proposed in [12] and is thus omitted here. One exception is that we allow flow channel junctions of a switch to choose their location along the flow channel spine despite the boundaries of the original rectangle reserved for them, since with the explicit channel routing information, the optimal location of these channel junctions may change.

3.3 Result Interpretation

Columba S outputs the physical synthesis results as an AutoCAD script file, which can be directly exported for mask fabrication.

4 Experimental Results

We implement the physical synthesis models in C++, and solve them using Gurobi [24], a mixed integer linear programming (MILP) solver. The program runs on a computer with 2.40 GHz CPU. We fabricate 2 ColumbaS-synthesized designs to demonstrate their feasibility.

Table 1 shows the feature values of the designs output by Columba 2.0 and Columba S (in the following we call them 2.0-designs and S-designs for short). The first 4 test cases have been reported in [12], and we add 2 large test cases (ChIP64 and ChIP128, both are synthetic applications based on [3]) to demonstrate the scalability of Columba S. For each test case, Columba S generates two designs: one contains a single multiplexer, and the other contains two multiplexers. The following trends can be derived from the comparison:

1. The program run time of Columba S is significantly reduced by tens to thousands of times. Columba 2.0 cannot solve the last two test cases within reasonable run time. But with the same computing power, Columba S synthesizes each design within few minutes.
2. The number of control inlets of S-designs is in general smaller than the 2.0-designs. In particular, 1-MUX designs use fewer control inlets than 2-MUX designs, since each multiplexer controls n channels with $1 + \lceil \log_2(n) \rceil$ control inlets. The logarithmic growth of inlets usage is essential for large-scale designs.
3. The flow channel lengths of S-designs are shorter than the 2.0-designs, since Columba S gathers all fluid manipulations in the functional region, and routes flow channels without detour. Note that the flow channels in the multiplexers in S-designs are not counted. Though these channels are fabricated in the flow layer, they are responsible for pressure transportation but not for fluid manipulation.
4. The area consumption of the S-designs is in general larger than the 2.0-designs. In particular, 2-MUX designs are larger than 1-MUX designs. Note that the extra area consumption results from the multiplexers but not from the functional region, and thus will not have a negative impact on application execution and microscope observation. As the scale (#u) of the designs increases, the area consumption of the functional region increases faster than the multiplexers, and thus the differences between the area consumption of S-designs and 2.0-designs become smaller.

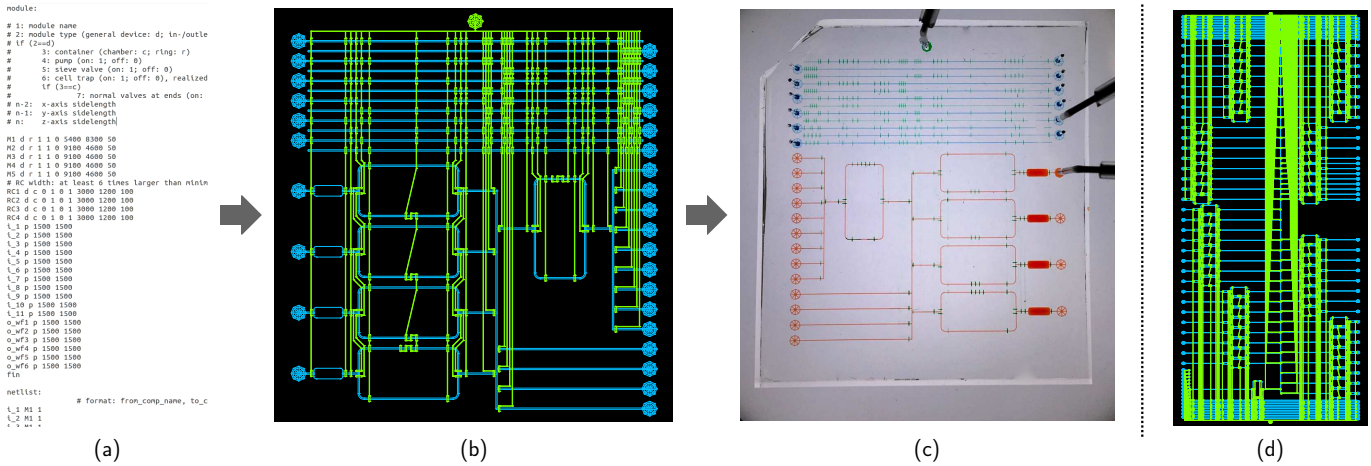


Figure 7: (a) Plain-text netlist description for a ChIP 4-IP application [3]. (b) The design synthesized from (a) by Columba S. (c) The chip fabricated from (b). (d) The Columba S 2-MUX design for a ChIP 64-IP application (partitioned into 8 parallel-execution groups).

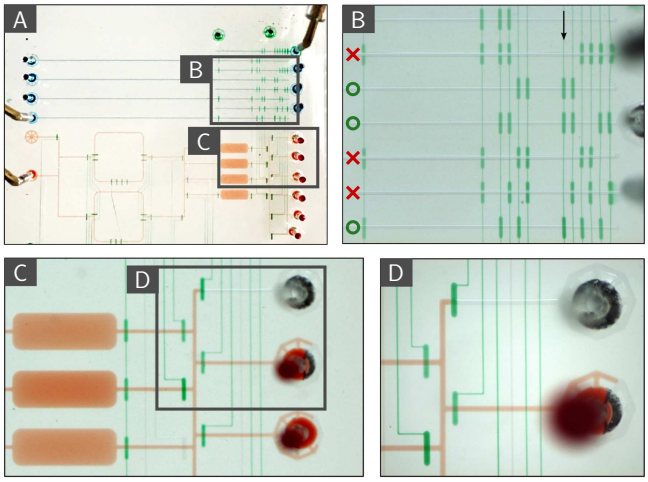


Figure 8: Multiplexing function on the fabricated design of an mRNA isolation application [7].

As a conclusion, Columba S keeps good performance as the design scales up. Compared with 1-MUX designs, 2-MUX designs provide simultaneous control of two valves but have to trade off with chip resources.

Figures 7(a)-(c) show the complete mLSI production flow supported by Columba S for the second test case [3], and Figure 7(d) shows the 2-MUX version Columba S design of the fifth test case. Figure 8 shows the fabricated design of the third test case. Figure 8(a) shows an overview of the design. Figure 8(b) shows the bit configuration to select the channel denoted by the arrow so that the corresponding valve shown in Figure 8(c) and (d) can be pressurized to block the fluid flow.

5 Conclusion

In this work we propose Columba S, an mLSI design automation tool focusing on scalability. We propose an architectural framework co-designed with module models and multiplexers. Under the new framework, we route all control channels in the vertical direction and all flow channels in the horizontal direction, which lays a foundation for efficient physical synthesis. We demonstrate the efficiency of Columba S with experiments, and we demonstrate the feasibility of Columba S designs with fabricated chips.

6 References

[1] J. Melin and S. Quake, "Microfluidic large-scale integration: the evolution of design rules for biological automation," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 36, pp. 213–231, 2007.

[2] D. D. Carlo, N. Aghdam, and L. P. Lee, "Single-cell enzyme concentrations, kinetics, and inhibition analysis using high-density hydrodynamic cell isolation arrays," *Anal. Chem.*, vol. 78, pp. 4925–4930, 2006.

[3] 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.

[4] C.-C. Lee, G. Sui, A. Elizarov, C. J. Shu, Y.-S. Shin, A. N. Dooley, J. Huang, A. Daridon, P. Wyatt, D. Stout, H. C. Kolb, O. N. Witte, N. Satyamurthy, J. R. Heath, M. E. Phelps, S. R. Quake, and H.-R. Tseng, "Multistep synthesis of a radiolabeled imaging probe using integrated microfluidics," *Science*, vol. 310, no. 5755, pp. 1793–1796, 2005.

[5] J. Liu, M. Enzelberger, and S. Quake, "A nanoliter rotary device for polymerase chain reaction," *Electrophoresis*, vol. 23, pp. 1531–1536, 2002.

[6] I. E. Araci and S. R. Quake, "Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves," *Lab on a Chip*, vol. 12, pp. 2803–2806, 2012.

[7] 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.

[8] 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.

[9] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic large-scale integration," *Science*, vol. 298, no. 5593, pp. 580–584, 2002.

[10] J. Liu, C. Hansen, and S. R. Quake, "Solving the "world-to-chip" interface problem with a microfluidic matrix," *Anal. Chem.*, vol. 75, pp. 4718–4723, 2003.

[11] 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.

[12] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, Early Access.

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

[14] N. Amin, W. Thies, and S. P. Amarasinghe, "Computer-aided design for microfluidic chips based on multilayer soft lithography," in *Proc. Int. Conf. Comput. Des.*, 2009, pp. 2–9.

[15] B. Crites, K. Kong, and P. Brisk, "Diagonal component expansion for flow-layer placement of flow-based microfluidic biochips," *ACM Trans. on Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 126:1–126:18, 2017.

[16] 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.

[17] C. Fang, Y. Wang, N. T. Vu, W.-Y. Lin, Y.-T. Hsieh, L. Rubbi, M. E. Phelps, M. MÃijschen, 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.

[18] 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, Automation, and Test Europe Conf.*, 2017, pp. 51:1–51:6.

[19] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1463–1475, 2014.

[20] 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, pp. 624–629.

[21] L. M. Fidalgo and S. J. Maerkl, "A software-programmable microfluidic device for automated biology," *Lab on a Chip*, vol. 11, pp. 1612–1619, 2011.

[22] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake, "Monolithic microfabricated valves and pumps by multilayer soft lithography," *Science*, vol. 288, no. 5463, pp. 113–116, 2000.

[23] M. W. Toepke and D. J. Beebe, "PDMS absorption of small molecules and consequences in microfluidic applications," *Lab on a Chip*, vol. 6, pp. 1484–1486, 2006.

[24] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*. http://www.gurobi.com.