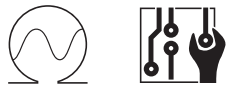
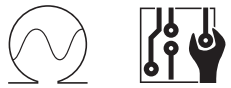


Technische Universität München
Department of Electrical Engineering and Information Technology
Chair of Electronic Design Automation

Simulation-Based High-Level Synthesis for Mircofluidic Large-Scale Integration (mLSI)

Master Thesis

Mengchu Li



Technische Universität München
Department of Electrical Engineering and Information Technology
Chair of Electronic Design Automation

Simulation-Based High-Level Synthesis for Mircofluidic Large-Scale Integration (mLSI)

Master Thesis

Mengchu Li

Supervising Professor :	Prof. Dr.-Ing. Ulf Schlichtmann
Topic issued :	01.10.2018
Date of submission :	15.03.2019

Mengchu Li
Matrikelnr.: 10854691
Ludwig-Maximilians Universität München
Informatics

Abstract

Multilayered continuous-flow microfluidic biochips are a rapidly developing platform for delicate bio-applications. Due to the high complexity of the biochip structure and the application protocols, there is an increasing demand for design automation approaches. Current research has enabled automated generation of biochip physical designs, and operation scheduling, and binding protocols, which has demonstrated the potential for better resource utilization and execution time reduction. However, the state-of-the-art high-level synthesis methods are on operation- and device-level, which assume fluid transportation paths to be always available but overlook the physical layout of the control and flow channels. This mismatch leads to a gap in the complete synthesis flow, and can result in performance drop, waste of resources due to redundancy or even infeasible designs. This work proposes to bridge this gap with a simulation-based approach, which takes a biochip design and a high-level protocol as inputs, and synthesizes channel-level pressurization protocols to support dynamic construction of valid fluid transportation paths. Experimental results show that the proposed method can efficiently validate and optimize the flow paths for feasible designs and protocols, detect redundant resource usage, and locate the conflicts for infeasible designs and protocols. It opens up a new direction to improve the performance and the feasibility of customized biochip synthesis.

Acknowledgements

I want to thank my family for their love and support.

Special thanks to my mother. I love you.

Many thanks to Tsun-Ming Tseng who is the best partner in research and in life I could ever imagine.

I would also like to thank Prof. Ulf Schlichtmann and Prof. Tsung-Yi Ho for their trust and teaching.

Contents

List of Figures

List of Tables

1. Introduction

Multilayered continuous-flow microfluidic biochips (Unger et al. 2000) are a promising lab-on-a-chip platform for high-throughput biological applications. It applies soft lithography technique to bond multiple patterned layers of elastomer, each of which consists of dedicated channels that allow gas or fluids to pass through. The multilayered structure enables the construction of active valves. A valve is a composite functional unit consisting of channel segments from different layers and a flexible membrane at the layer interface. A two-layer polydimethylsiloxane (PDMS) push-up valve (Melin & Quake 2007) is shown in Figure 1.1, where a flow layer is on top of a control layer. When gas or oil infuses the bottom control channel, the channel will be pressurized and thus push the membrane upwards. Since the flow channel segment has a rounded profile that perfectly fits the expanded membrane, the channel will be sealed and the fluid movement in this channel will be blocked (Lee et al. 2005).

With channels and valves, bioengineers have been able to build delicate microfluidic systems on coin-sized chips. But as the complexity of the application protocols and the integration scale of the biochips increase, manually designing the chips becomes more and more time-consuming and error-prone, which results in a strong demand for automated software synthesis tools. Many design automation researchers put their efforts into analyzing and solving this design problem. The ambition is to construct a completely automated synthesis flow, which can transform a *high-level abstraction* of a given application into a feasible and optimized

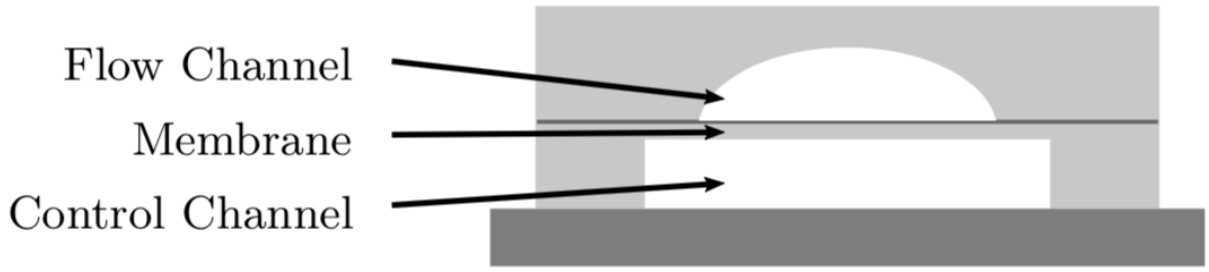


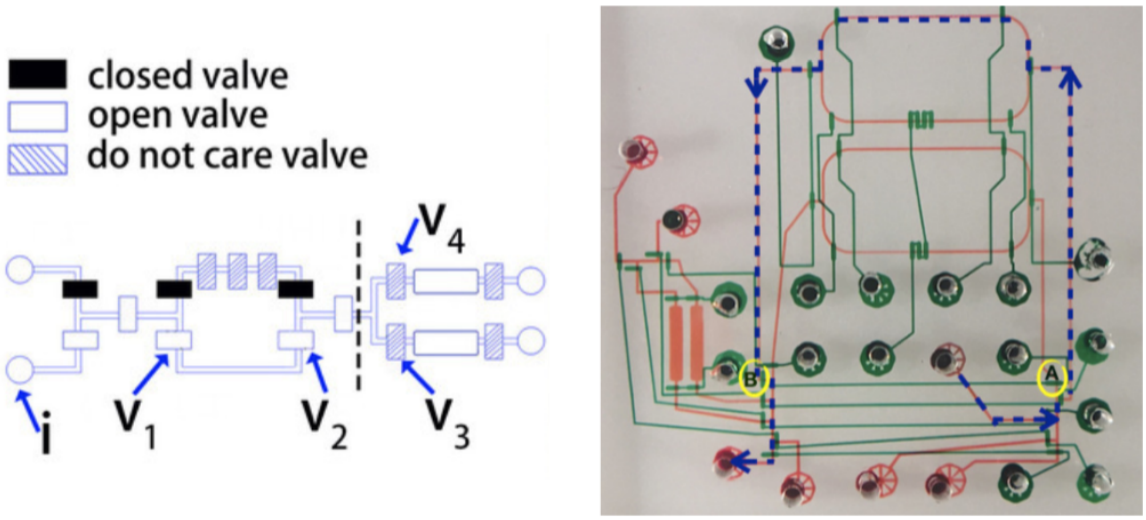
Figure 1.1.: A two-layer polydimethylsiloxane (PDMS) push-up valve. When the control channel segment is pressurized, the flow channel will be completely sealed by the expanded membrane.

chip design with an *explicit protocol* for executing the application.

With a decade of effort, researchers have achieved significant progress covering both high-level synthesis and physical design. Specifically, up to 2019, there have been: high-level modeling methods that optimize the resource utilization based on given application protocols (Li et al. 2016) (Li et al. 2017); place-and-route tools that synthesize manufacturing-ready physical designs targeting applications of different scales (Tseng et al. 2016) (Tseng, Li, Freitas, McAuley, Li, Ho, Araci & Schlichtmann 2018) (Tseng, Li, Freitas, Mongersun, Araci, Ho & Schlichtmann 2018), scheduling and fluid routing approaches that map operations to given physical topologies (Minhass et al. 2011) (Minhass et al. 2018), and other frameworks focusing on specific optimization criteria such as fluid storage (Tseng et al. 2015) (Liu et al. 2017) and control pin reduction (Hu et al. 2017) (Zhu et al. 2018), etc.

Though the proposed approaches gradually piece the complete automatic synthesis flow together, it still leaves some missing links between the designs and the protocols.

First of all, the state-of-the-art high-level synthesis methods are on operation- and device-level only, which assume fluid transportation paths to be always available



(a) (b)

Figure 1.2.: (a) A manually derived valve actuation protocol which intends to form a flow path from i to v_2. (b) A fabricated chip where valve A and valve B are connected to share the same pressure.

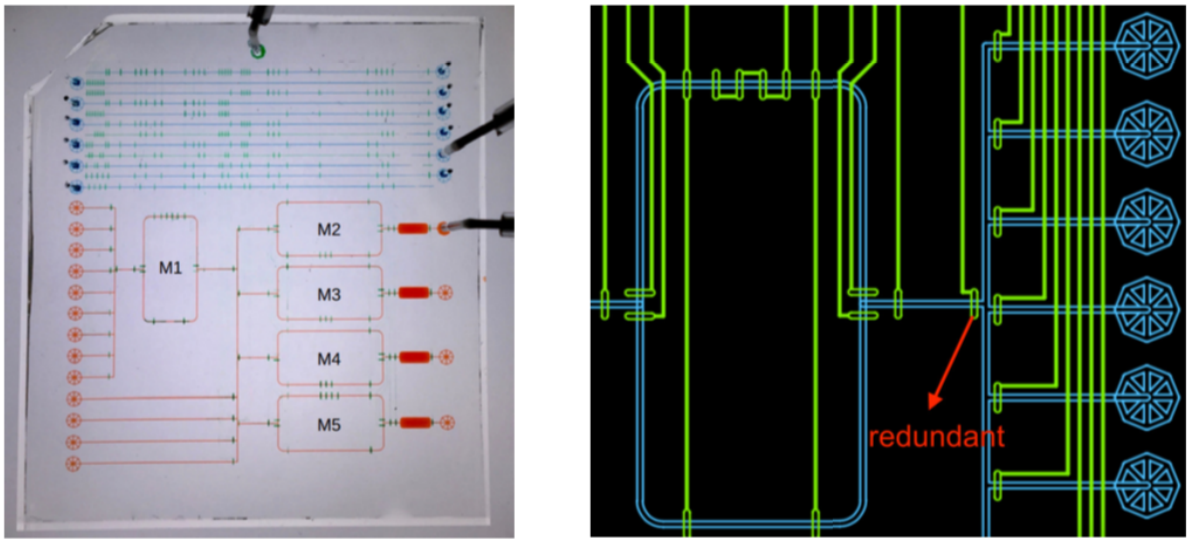
but overlook the interaction between the control and flow channels. Specifically, flow path construction is a dynamic process in multilayered biochips. To transport fluids from one location to another, it is not enough to ensure the static physical connection of flow channels, but also necessary to pressurize certain control channels to temporally block the fluid movement in unwanted directions. Though current approaches can synthesize the operation scheduling and device binding results, they cannot produce the explicit control channel pressurization sequences to validate the dynamic construction of flow paths. Flow path validation is indispensable because of two reasons:

- Reason 1: flow path construction must not only consider the local channel structure, but should consider the whole chip. As pointed out in (Minhass et al. 2018), to move fluids from location a to location b, three sub-paths need to be considered: a path from a to b, a path from an inlet to a, and a path from b to an outlet. Besides, as mentioned earlier, it is insufficient to ensure

the physical connection of the corresponding flow channels, but also necessary to guarantee that all valves along the paths are properly pressurized and depressurized. Manually deriving the pressurization sequences can be error-prone. An example from (Hu et al. 2017) is shown in Figure 1.2(a), which intends to transport fluids from an inlet i to the bottom half-ring between valves v_1 and v_2, but mistakenly sets valves right to the dash-line as *don't care* status, indicating that these valves are irrelevant and can be either pressurized or not. However, if v_3 and v_4 are both pressurized and thus 'closed', there will be no sub-path from v_2 to an outlet. Since the initial status of a channel is not a vacuum but filled with air, it is hardly possible[1] to form the expected pattern of fluid flow.

- Reason 2: different sub-paths may conflict with one another due to pressure-sharing valves. In multilayered biochips, valves are connected to external pressure sources via control pins, which are punch holes on the chip that occupy large area (Stanford Foundry n.d.). To reduce the control pin usage, it is a common approach to connect multiple valves sequentially with the same control channel to share the same control pin. In this case, pressurizing one control channel will lead to the pressurization of all valves along the channel, which may result in unexpected blockage of flow paths. For example, Figure 1.2(b) shows the photo of a biochip (Tseng, Li, Freitas, McAuley, Li, Ho, Araci & Schlichtmann 2018) consisting of two ring-shaped mixers. The blue dash line demonstrates a path for fluid transportation from an inlet to the upper mixer. To prevent the bottom mixer from contamination, an intuitive approach is to pressurize valve A. However, since valve A and valve B are sequentially connected by a green control channel, pressurizing valve A leads to the pressurization of valve B, which results in the unwanted blockage of

[1]Since PDMS is gas-permeable (Firpo et al. 2015), given large-enough pressure from i, the fluids may slowly push the air out of the channel, but the transportation time will be significantly prolonged and the fluids may be mixed with air bubbles.



(a) (b)

Figure 1.3.: (a) A biochip that supports fluid-multiplexing from M1 to M2, M3, M4 and M5. (b) Part of a biochip design where the functionality of the valve at the channel branch next to the mixer overlaps with the right separation valve of the mixer.

the sub-path from the target ring to an outlet.

Another mismatch between the design and the protocol results from fluid-multiplexing operations. Fluid-multiplexing refers to fluid transportation from the same source to multiple destinations, which is common in high-throughput bio-applications. Though fluid multiplexing is typically specified as a single operation in current high-level protocols, it may require a dynamic change of the flow path if the flow channels connected to the different destinations are of different lengths[2]. For example, Figure 1.3(a) shows a photo of a biochip (Wu et al. 2009) containing five ring-shaped mixers that support fluid-multiplexing from M1 to M2, M3, M4, and M5. When the transportation from M1 to its nearest mixer M3 finishes, the transportation to the other target mixers will be still in progress. In this case, the

[2]Because of the area confinement, current physical synthesis methods usually trade off the length-matching feature for routability. Since the application execution time is dominated by biological phenomena but not fluid transportation, this trade-off is considered acceptable(Crites et al. 2017).

control channel pressurization protocol needs to be updated to block the fluid motion in M3 without disturbing other sub-paths. Neglecting this feature can lead to inaccuracy of the scheduling results, difficulty in the execution process, and most importantly, incorrect control layer design that fails to support the target application.

Last but not least, due to the absence of channel-level protocols, current physical synthesis methods cannot precisely estimate the usage of control resources, which leads to redundancy in the design. For example, Figure 1.3(b) shows a design synthesized by a state-of-the-art tool (Tseng, Li, Freitas, Mongersun, Araci, Ho & Schlichtmann 2018), which places a valve at each branch of the flow channels to control fluid transportation in all directions. However, the functionality of the valve at the flow channel branch close to the mixer completely overlaps with the right separation valve of the mixer, and thus can be removed without hurting the performance of the chip. Removing redundant control components including valves, channels, and pins means cleaner control layer design and chip size reduction. Besides, since control channels are much thinner than flow channels (Stanford Foundry n.d.), they are more vulnerable to damages and are thus a major source of chip defects (Hu et al. 2014). Thus, avoiding redundancy also contributes to the robustness of the chip.

In this work, we (my colleagues and I) propose to bridge the gap between current high-level and physical synthesis methods with a simulation-based approach named VOM. It takes a biochip design and a high-level protocol as inputs, and then synthesizes and optimizes channel-level pressurization protocols to support dynamic construction of valid flow paths. The contribution of VOM includes:

- It is the first approach that synthesizes channel-level protocols, which provides a basis for validating the dynamic construction of flow paths.

- It proposes an event-driven mechanism to automatically update the channel-level protocol to support fluid-multiplexing.

- It optimizes the channel-level protocol based on adjustable criteria including execution time and resource usage.

- It detects the design redundancy and locates the conflicts for incompatible designs and protocols, which opens up a new direction to improve the performance and the feasibility of customized biochip synthesis.

2. VOM Building Blocks

2.1. An Overview of VOM

In this section, we will shortly introduce the basic building blocks of VOM to provide a quick overview of the our algorithmic flow. The system diagram is shown in Figure 2.1.

2.1.1. Inputs

VOM takes a design and a high-level (operation- and device-level) protocol as the input.

- The *design* specifies the physical features of a given multilayered biochip. VOM interprets the the flow layer structure into a weighted graph $G = (V, E)$ consisting of vertices V and undirected edges E. Each vertex $v \in V$ represents a 'landmark' on the chip, which can be an inlet, an outlet, a valve, or a flow channel intersection. Each edge $e \in V$ represents a flow channel between the landmarks, with weight coefficients w_e, h_e, and l_e representing the dimension of this flow channel. Besides, the design also specifies the control channel connection among valves.

- The *high-level protocol* specifies a sequence of fluid transportation operations.

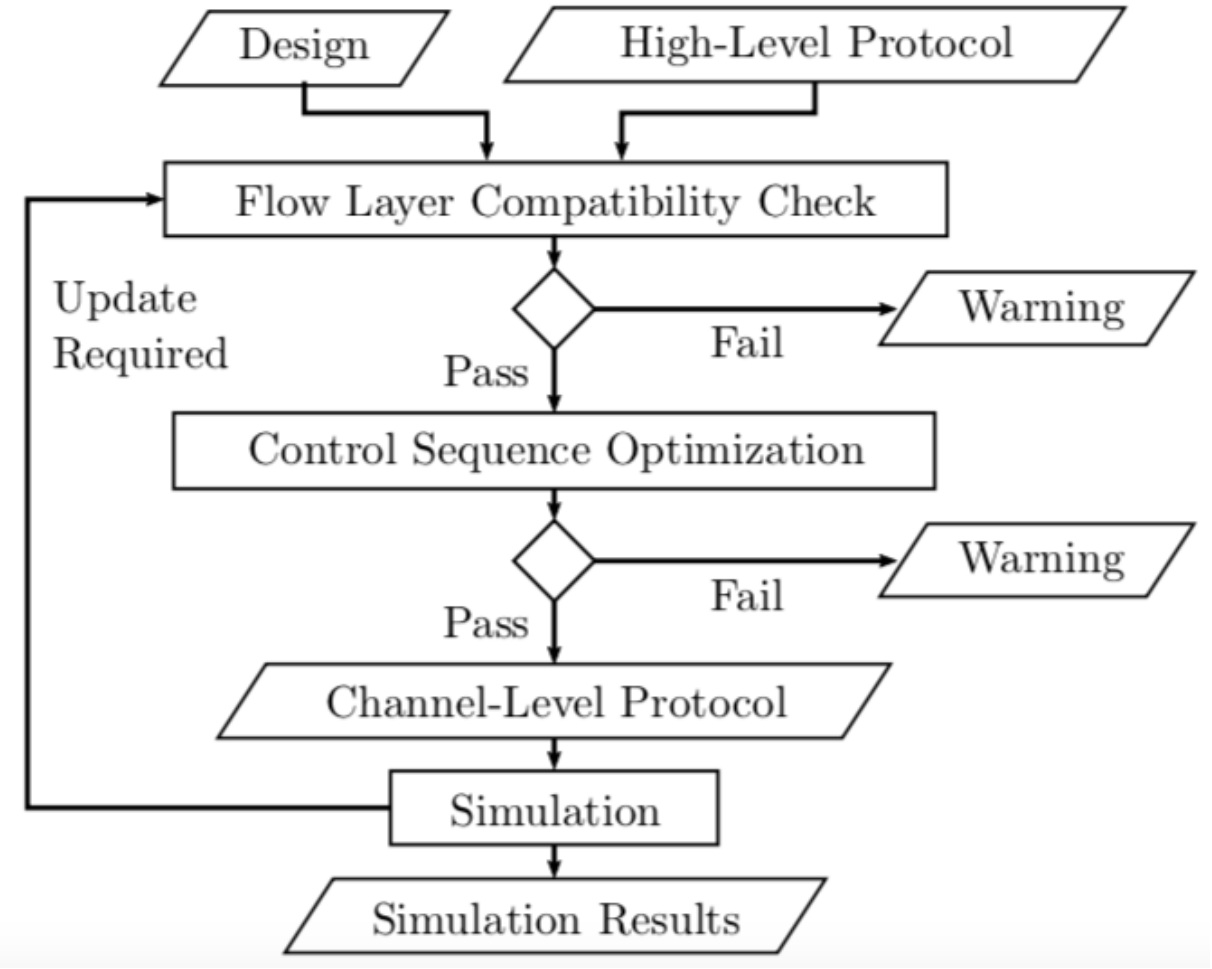


Figure 2.1.: System diagram of VOM

For each operation, it describes the target flow path as a set of inlet vertices that initiate the fluid movement, a set of destination vertices at which fluid movement ends, and a set of vertices that should be excluded from the flow path.

2.1.2. Flow Layer Compatibility Check

Neglecting the control channel connections, VOM first performs a quick check to find out whether the target flow paths can be supported by the flow layer design.

If the check fails, VOM outputs a warning message indicating that the flow-layer design is incompatible with the protocol. Otherwise, VOM proceeds to the next step.

2.1.3. Control Sequence Optimization

In this step, VOM restores the channel connection among valves based on the original control-layer design, and looks for all candidate control channel pressurization protocols that can construct the target flow path. If the construction fails, VOM outputs a warning message indicating that the control-layer design is incompatible with the protocol, i.e. some control channel connections are improper. Otherwise VOM carries out an optimization process based on user-defined criteria to finalize a channel-level protocol from all candidates.

2.1.4. Simulation

Based on the graph G and the channel-level protocol, VOM simulates the application execution process in an event-driven manner, which allows it to predict the demand for protocol update in fluid-multiplexing operations. If such demand is detected, VOM automatically updates the high-level protocol and turns back to synthesize the channel-level protocol that supports the update.

2.1.5. Output

If the design and the high-level protocol are compatible, VOM outputs three types of results: a *channel-level protocol* where the dynamic formation of all flow paths

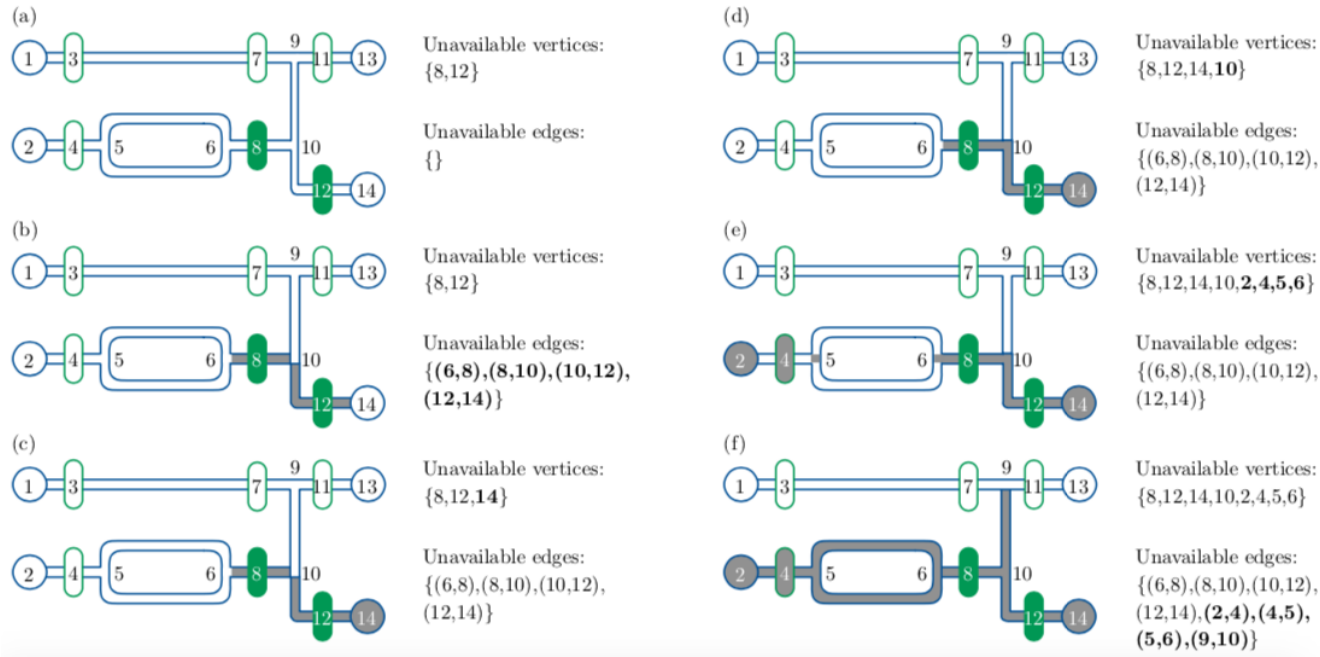


Figure 2.2.: An example of the recursive modification procedure for locating all valid flow paths.

are validated; a *schedule* for each fluid transportation operation; and a report on *resource usage* in the control layer.

2.2. Flow Path Validation

Before diving into the detailed synthesis flow of VOM, we first introduce our flow path validation algorithm, which is an important module frequently applied in most of the synthesis steps.

We perform flow path validation based on a Graph $G = (V, E)$ that models the physical structure of a chip, and a set of vertices $V_{block} \subseteq V$ that specifies the initial blockage in the chip.

As the first step, we assign an *availability* for each vertex $v \in V$ and each edge

$e \in E$ to indicate whether the vertex/edge can be used to form a valid flow path. All edges are initialized as *available* by default, and for each vertex $v \in V$, if $v \in V_{block}$, it is initialized as *unavailable*, otherwise it is initialized as *available*.

We then propose an algorithm to recursively modify the availability of all vertices and edges to locate the valid flow paths. To guide the modification, we formulate the following rules:

1. If an edge consists of an unavailable vertex, the edge is unavailable.

2. If a vertex belongs to an unavailable edge and the vertex does not represent a flow channel intersection, the vertex is unavailable.

3. If a vertex belongs to only one available edge and the vertex is not an in-/outlet, the vertex is unavailable.

4. If a vertex is not reachable from the inlet or it cannot reach the outlet, the vertex is unavailable.

Figure 2.2 demonstrates our modification process with an example, where the design consists of 14 vertices, among which vertices 8 and 12 are specified as initially unavailable, as shown in Figure 2.2(a).

The first modification rule is based on the fact that the blocked end of a flow channel can neither initiate nor receive a fluid flow. As a result, we specify the edges connected to vertices 8 and 12 as unavailable, as shown in Figure 2.2(b).

The second modification rule indicates that a chip component without valid flow channel connection is invalid. Thus, as an end point of the unavailable edge $(12, 14)$, vertex 14 is specified as unavailable, as shown in Figure 2.2(c).

The third modification rule follows the fact that an intermediate point in the flow path must be connected to at least two unblocked flow channels. Since vertex 10 only has one valid flow channel connection and it is neither an inlet nor an outlet, it is specified as unavailable, as shown in Figure 2.2(d).

The fourth modification rule indicates that a valid flow path must include an inlet and an outlet to ensure proper pressure drop. As a result, vertices 2, 4, 5, and 6 are specified as unavailable, as shown in Figure 2.2(e).

By recursively applying the modification rules until no more vertex or edge can be specified as unavailable, we can locate a valid flow path, as shown in Figure 2.2(f).

3. VOM Synthesis Flow

Based on the system diagram and the flow path validation approach, this section gives a detailed description of each VOM synthesis step.

3.1. Flow Layer Compatibility Check

The first step of VOM is to quickly check whether a target flow path can be supported by the given flow layer design.

A target flow path is specified in the input protocol as a set of inlet vertices V_{inlet}, one or multiple destination vertices V_{target}, and a set of vertices that should be excluded from the flow path (to avoid contamination) V_{exclude}.

VOM carries out the check with flow path validation.

3.1.1. Specify V_{block}

To find out whether fluids can arrive at the destination vertices V_{target} without contaminating V_{exclude}, we initially add all vertices in V_{exclude} to V_{block} to indicate that these vertices are not available for forming the flow path. Besides, we also add inlet vertices that do not belong to the current flow path to V_{block} to prevent

them from contamination.

However, not all vertices in V_{block} have the risk of contamination. Specifically, a flow path consists of two sub-paths: a sub-path from an inlet to the destination, and a sub-path from the destination to an outlet. Since the fluid transportation will be stopped when fluids arrive at their destination, the latter sub-path will not be contaminated. Adding contamination-free vertices into V_{block} leads to two issues:

1. It may hurt the feasibility of a valid flow path. Take the chip shown in Figure 2.2(a) as an example. For a given protocol $V_{\text{inlet}} = \{1\}$, $V_{\text{target}} = \{7\}$ and $V_{\text{exclude}} = \{8, 11, 12\}$, which aims to transport fluids from inlet 1 to the flow channel between 3 and 7 whiling keeping all other channels contamination free, it is safe to specify V_{exclude} as empty since all vertices except for 1, 3 and 7 are in the contamination-free sub-path. However, if we add the whole set of V_{exclude} to V_{block}, there will be no valid flow path left in the design, since vertex 7 cannot reach any outlet.

2. It may prolong the time needed for the transportation. Still using the above example, if we leave vertex 11 available and add 8 and 12 to V_{block}, though a valid flow path can be constructed as shown in Figure 2.2(f), the hydraulic resistance[1] of the flow path will be enlarged compared to keeping V_{block} empty. Thus, fluids will take more time to reach their destination.

VOM avoids these issues by transforming G into a directed graph, which is done with a two step procedure:

- First, a distance value is assigned to each node and its reachable inlet. The distance value is defined as the number of edges connecting this node to the

[1] A more detailed discussion about hydraulic resistance will be introduced in Section 3.2

corresponding inlet. If there are multiple options for the distance value, i.e. if the node is connected to an inlet by different edges, the largest option will be chosen as the final distance value.

- Second, the direction of each edge is decided by the distance value of its two nodes, where the node with a smaller distance value is considered as the origin of the edge, and the node with a larger distance value is considered as the end of the edge.

The detailed algorithm for calculating the distance value is shown in Algorithm 1.

Based on the directed graph, we can derive whether a vertex is in a contamination-free sub-path. If a vertex in V_{block} has no risk of contamination, we remove it from V_{block}.

3.1.2. Check Compatibility

With the design graph G and the set V_{block}, VOM performs flow path validation to locate all available vertices and edges. If for all $v \in V_{\text{inlet}} \cup V_{\text{target}}$, v remains available after the validation, we can conclude that the flow-layer structure is capable of supporting the target flow path, and VOM proceeds to the next step.

3.2. Control Sequence Optimization

If the input passes the flow-layer compatibility check, VOM first collects all control channel pressurization protocols that can support the target flow paths, and then decides one final protocol based on user-defined optimization criteria.

Data: Graph $G = (V, E)$, $I \subset V$

find out all $v \in V$ that represent branches, record the number of their available connections as $c(v)$;
mark all $e \in E$ as *unprocessed*;
for $i \in I$ **do**
    build a distance record $d(v, i) \to \mathbb{Z}^+$ to denote the distance between an arbitrary vertex v and the inlet i, initialize $d(i, i)$ to 0;
    while *no new record is added* **do**
        while *no new record is added* **do**
            for $(v_1, v_2) \in E$, (v_1, v_2) *marked as unprocessed* **do**
                if $d(v_1, i)$ *is known* **then**
                    if v_1 *is not a branch or* $c(v_1) == 1$ **then**
                        mark the edge as *processed*;
                        if v_2 *is not a branch or* $c(v_2) == 1$ **then**
                            add a distance record $d(v_2, i) = d(v_1, i) + 1$
                        else
                            $c(v_2) - = 1$
                        end
                    end
                end
                if $d(v_2, i)$ *is known* **then**
                    analogous to the above block, and thus ommited;
                end
            end
        end
        for $v \in V$ *with* $c(v) > 1$ **do**
            int dist= 0;
            for $(v', v) \in E$ *marked as unprocessed* **do**
                if *dist*<= $d(v', i)$ **then**
                    dist= $d(v', i) + 1$
                end
            end
            if *dist*> 0 **then**
                add a distance record $d(v, i)$ =dist
            end
        end
    end
end

Algorithm 1: Distance assignment

3.2.1. Collect Control Channel Pressurization Options

VOM builds a search tree to efficiently traverse all control channel pressurization options. The algorithm is inspired by the idea of *branch and cut* for solving integer linear programming problems (Padberg & Rinaldi 1991).

Except for a virtual root (level-0), each node in the search tree represents a control channel pressurization option, where a level-1 node is a set of cardinality 1 representing an option of pressurizing one control channel, a level-2 node is a set of cardinality 2 representing an option of pressurizing two control channels, etc. Thus, for a design that consists of n control channels, there will be n level-1 nodes in the tree.

We then perform breadth-first search to traverse the tree. For each node, we perform flow path validation with V_{block} defined as all valve vertices contained in the control channels of the node, i.e., we assume all control channels in this option to be pressurized and look for the resulting flow paths. Based on the validation results, we assign the node to one of three different classes:

1. The resulting paths do not contain all vertices in V_{inlet} and V_{target}. In this case, we classify the node as *fail*, which represents an invalid option.

2. The resulting paths contain all vertices in V_{inlet} and V_{target}, and do not contain any vertex in V_{exclude}. In this case, we classify the node as *success*, which represents a valid option.

3. The resulting paths contain all vertices in V_{inlet} and V_{target}, but they also contain some of the vertices in V_{exclude}. In this case, we classify the node as *candidate*.

After traversing all nodes on the same level, for each node that is classified as fail or success, we cut all nodes that represent a superset of this node from the tree. Because if a node is classified as *fail*, it means that this option leads to unwanted blockage of the flow paths. Thus, options containing the same set of control channels will also lead to the same unwanted blockage. On the other hand, if a node is classified as *success*, it means that this option is enough to support the protocol. Thus, it is also unnecessary to include additional control channels.

After traversing all nodes in the tree, we collect the nodes classified as success as potential control channel pressurization options.

Figure 3.1 shows an illustration of our synthesis method for a design consisting of four control channels. We first build a search tree, and then performs flow path validation for each of the level-1 nodes. Based on the validation results, $\{c_1\}$ is classified as success and $\{c_2\}$ is classified as fail. Thus, we cut all the supersets of $\{c_1\}$ or $\{c_2\}$ from the tree, which leaves $\{c_3, c_4\}$ as the only level-2 node. After another flow path validation process, $\{c_3, c_4\}$ is classified as success. Thus, we collect it together with $\{c_1\}$ as the candidates.

3.2.2. Optimization Criteria

VOM provides two optimization criteria for selecting a control channel pressurization protocol from all candidates. A user of VOM can choose one of the criteria according to his/her demands.

Resource-Oriented Optimization

The first criterion is to execute the application with a minimized number of control channels. Control channels that are never pressurized during the whole application can be considered as redundant and thus removed from the chip. Since the removal of a control channel also results in the removal of all valves it contains and the control pin it is connected to, by minimizing the number of pressurized control channels, we minimize the control-layer redundancy.

We propose an integer-linear-programming method to solve this optimization problem. Specifically, we model the problem as the follows:

Input:

— A set C of natural numbers representing the indices of all control channels in a design.

— A sequence $O = o_1, \cdots, o_n$ representing all fluid transportation operations in the application.

— A function f that maps each operation $o \in O$ to a set of control channel pressurization options $\{C_1, \cdots, C_m\}$ where $C_1, \cdots, C_m \subseteq C$.

Output:

— A set $M \subseteq C$ representing the control channels that have been pressurized in the application. In other words, M satisfies:

$$\forall o \in O \; \exists M' \subseteq M : \; M' \in f(o)$$

Optimization objective:

— Minimize the cardinality of M.

To build the model, we introduce a binary variable b_{c_i} for each c_i in C to indicate whether the channel is pressurized in the application. Also, we introduce a binary variable b_{o_j,C_k} to represent whether a control channel pressurization option $C_k \in f(o_j)$ is selected to execute operation o_j.

We introduce the following constraint for each $o_j \in O$ to model that exactly one control channel pressurization option must be chosen for each operation:

$$\sum_{C_k \in f(o_j)} b_{o_j,C_k} = 1. \tag{3.1}$$

We introduce the following constraints to model that if an option $C_k \in f(o_j)$ is chosen to execute an operation o_j, all control channels $c_\tau \in C_k$ must be identified as pressurized, i.e. $b_{o_j,C_k} = 1$ implies $b_{c_\tau} = 1$:

$$\forall o_j \in O \; \forall C_k \in f(o) \; \forall c_\tau \in C_k : \quad b_{o_j,C_k} \leq b_{c_\tau}. \tag{3.2}$$

Thus, the model can be formulated as:

$$\begin{aligned} &\textbf{Minimize:} && \sum_{c_i \in C} b_{c_i}. \\ &\textbf{Subject to:} && (1)(2). \end{aligned}$$

Based on the optimization results, for each operation o_j, we select the control

channel pressurization option C_k with $b_{o_j,C_k} = 1$.

Transportation-Time-Oriented Optimization

The second criterion is to execute the application with minimized fluid transportation time. Fluid flows in micro- and nano-environments are considered, as a first approximation, to be viscous, incompressible, and generally laminar (Stone 2007), which enables the prediction of fluid motion based on relatively simple calculations. Specifically, the hydraulic behavior of pressure-driven flow is governed by the Hagen-Poiseuille equation, which corresponds to Ohm's law in electric circuit analysis, where the pressure drop is analogous to the voltage drop, the volumetric flow rate to the current, and the hydraulic resistance to the electric resistance (Oh et al. 2012). Thus, if we assume the input pressure to be a constant value, to achieve the shortest transportation time with a maximized flow rate, we need to select the control channel pressurization protocol that forms a path with smallest hydraulic resistance.

Our optimization follows the following model of fluid dynamics in micro- and nano-environments (Stone 2007):

Given a pressure drop Δp and the hydraulic resistance R_H of a fluid-filled channel, the flow rate Q ($\frac{\text{volume}}{\text{time}}$) is calculated as

$$Q = \frac{\Delta p}{R_H}. \tag{3.3}$$

For a rectangular channel of height h and width w in multi-layered continuous-flow

microfluidic biochips, its hydraulic resistance is approximated as:

$$R_H = \frac{12\mu L}{h^3 w}, \tag{3.4}$$

where μ is the viscosity of the to-be-filled fluids and L is the length of the channels.

Analogous to an electric circuit, for n channels with individual hydraulic resistance $R_1, \cdots, R_n$ that are placed in series, the effective resistance R_s is calculated as:

$$R_s = R_1 + \cdots + R_n, \tag{3.5}$$

and if those channels are placed in parallel, the effective resistance R_p is calculated as:

$$R_p = \frac{1}{\frac{1}{R_1} + \cdots + \frac{1}{R_n}}. \tag{3.6}$$

For each control channel pressurization option, we perform flow path validation to achieve its resulting flow path and calculate its hydraulic resistance. Specifically, we build hyper-edges for edges that are connected in parallel or in series, where the resistance of a hyper-edge is calculated after the resistance of all its edge-segments has been calculated. To note is that a hyper-edge can be built upon other hyper-edges to represent a multi-level fluid-multiplexing structure.

After the calculation, we select the control channel pressurization protocol that constructs a flow path with the smallest hydraulic resistance.

3.3. Simulation: Event-driven Protocol Update

With the optimized channel-level protocol, VOM performs simulation to automatically update and synthesize the protocol for fluid-multiplexing operations.

The simulation starts from the first operation in the input protocol, for which we set V_{block} as the set of all valve vertices that are pressurized according to the optimized channel-level protocol, and perform flow path validation.

With the validated graph, we build a hierarchical model of the flow path. For example, Figure 3.2 shows a flow path $1 \rightarrow 6$ consisting of three sequentially connected sub-paths $1 \rightarrow 2$, $2 \rightarrow 5$, and $5 \rightarrow 6$, among which the sub-path $2 \rightarrow 5$ again consists of two parallel sub-paths. With the Hagen-Poiseuille equation introduced in Section 3.2.2, we can calculate the hydraulic resistance of the sub-paths based on the hydraulic resistance of the underlying edges, and thus derive the flow rate distribution among the sub-paths. Specifically, in the example shown in Figure 3.2, the upper sub-path $2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ has a smaller resistance than the bottom parallel sub-path $2 \rightarrow 5$. As a result, the upper sub-path will inherit 60% of volumetric flow rate from its predecessor path, while the bottom sub-path will only inherit 40% of the volumetric flow rate from the same predecessor path. And since the sub-path $2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ contains a pair of parallel edges $(3, 4)$ with balanced hydraulic resistance, its volumetric flow rate is further divided into two parts. Thus, suppose that the input flow rate is $100\mu m^3/s$, the volumetric flow rate at each of the parallel edges $(3, 4)$ will be calculated as $100\mu m^3/s * 60\% * 50\% = 30\mu m^3/s$.

3.3.1. Edge Processing based on List-Scheduling

After computing the flow rate distribution, we can simulate the fluid status in each available edge at a given time applying a modified list-scheduling algorithm (Graham 1966). In general, the idea of list scheduling is to define a priority function for a set of to-be-accomplished tasks. With limited resource for execution, tasks with higher priority are carried out earlier than tasks with lower priority.

In the simulation process, the calculation of the fluid status in each edge of the design graph can be considered as a task, and the edge dependency shown in the hierarchical model can be considered as the priority function. Thus, the edge directly connected to the inlet is processed first, and other edges are processed after their predecessors.

3.3.2. Events in Fluid Transportation

To track the operation execution along time, *events* are introduced to model the important check points in the fluid transportation process. In VOM, an event happens at the moment when fluids reach the end vertex of an edge. An event is considered to be *insignificant*, if the vertex that has just been reached is not a destination vertex. On the other hand, an event is considered to be *significant*, if the vertex that has just been reached is a destination vertex.

Specifically, when the simulation starts, the time needed for the fluids to reach the closest vertex to an inlet is first calculated as t_1, which indicates that an event will be triggered at time t_1.

- If the event is insignificant, it indicates that the fluid status of all destination

vertices at time t_1 is unchanged, i.e. the fluid transportation operation is not completed at t_1. Thus, VOM can proceed to calculate the time needed for the fluids to reach the next-closest vertex, i.e. the time that needed to trigger the next event.

- If the event is significant, it indicates that a destination vertex is reached, i.e. the fluid transportation operation is possibly completed at t_1. In this case, VOM needs to check the fluid status at other destination vertices. If there is no other destination vertex, i.e. all destination vertices has been reached, VOM can proceed to simulate the next operation. Otherwise, the control channel pressurization protocol of the current operation needs to be updated at t_1 to support the unfinished parts of the fluid transportation.

3.3.3. Update of Control Sequences

To support the update of the protocols, after processing each edge, VOM takes a pause and generates an event signal to tell which vertex v_r has just been reached by the fluids. Based on this signal, it chooses one of the following three options to continue the synthesis:

1. If the event is insignificant, the simulation continues without changing any configuration.

2. If the event is significant, and v_r is the last destination vertex that is reached by fluids, the simulation of the current operation is terminated and VOM proceeds to the next operation.

3. If the event is significant, but there is at least one other destination vertex that has not been reached, the simulation of the current operation is termi-

nated and VOM updates the current high-level protocol by moving v_r from V_{inlet} to V_{target}. After that, it proceeds to generate the new control channel pressurization protocol for the updated operation.

To note is that when a significant event occurs, after the new control channel pressurization protocol has been obtained, another simulation based on the previous fluid status will be performed and the event-driven synthesis process will be repeated.

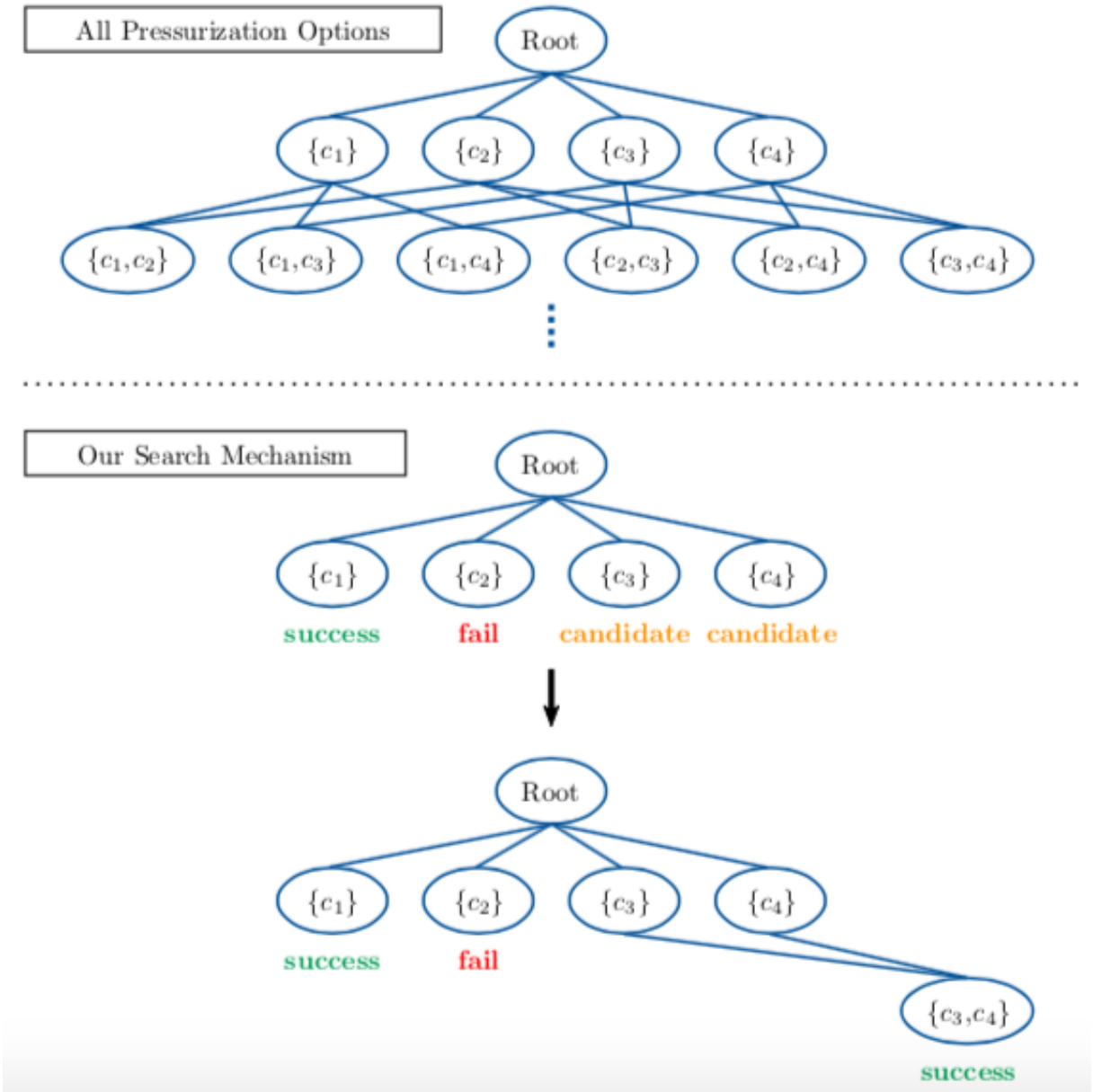


Figure 3.1.: Search and collect all valid control channel pressurization options

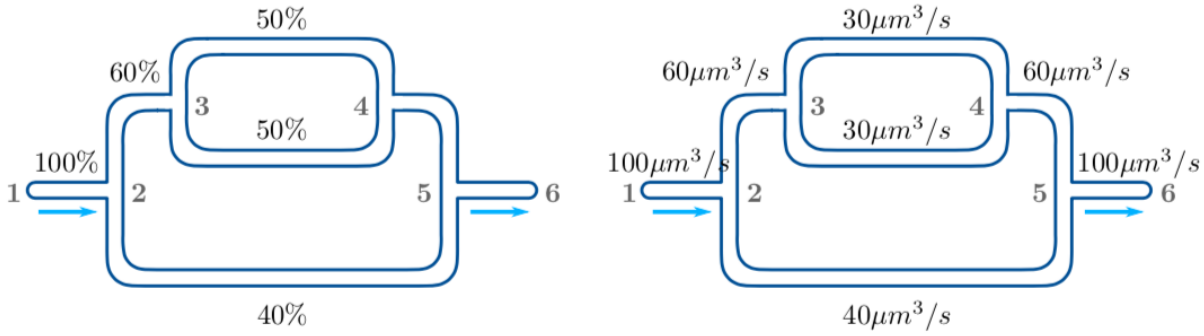


Figure 3.2.: An example of the flow rate distribution model.

4. Experimental Results

We implement VOM in C++, and test its performance with three multilayered continuous-low biochip designs.

- The first design consists of one rectangular chamber and one ring-shaped mixer, as shown in Figure 4.1. Except for sequentially connected pumping valves, all valves in this design are directly connected to control pins and thus do not share pressure with the others.

- The second design consists of three ring-shaped mixers, as shown in Figure 4.2, where valves of similar functionality but in different mixers are sequentially connected to share the same control pins.

- The third design consists of two rectangular chambers and two ring-shaped mixers. It is an application-specific design proposed in (Tseng, Li, Freitas, McAuley, Li, Ho, Araci & Schlichtmann 2018) and shown in Figure 1.2(b). It also applies an aggressive pressure-sharing strategy to reduce the number of control pins on the chip, which results in a relatively larger number of sequentially connected valves.

Table 4.1 shows the input feature and the synthesis results. In the following we analyze the results with details:

For the first design, we tested five fluid transportation operations from different

Table 4.1.: Input features and synthesis results

Id	$\|L\|$	$\|V\|$	$\|C\|$	$\|O\|$	Fluid-Multiplexing	Update	$C_{removal}$	Compatibility.	Run Time
1	24	16	13	5	0	0	2	Yes	0.018s
2	48	34	12	6	0	0	0	Yes	0.645s
				2	2	4	3	Yes	0.409s
3	62	38	15	5	1	1	NA	No	5.022s
			16	5	1	1	2	Yes	13.722s

$|L|$: the number of landmarks in the design; $|V|$: the number of valves in the design $|C|$: the number of control channels in the design; $|O|$: the number of operations in the input protocol; Fluid-Multiplexing: the number of fluid multiplexing operations in the input protocol; Update: the number of automatic protocol updates that VOM performs; $|C_{\text{removal}}|$: the number of redundant control channels that can be removed from the design; Compatibility: whether the design is compatible with the high-level protocol; Run Time: program run time.

inlets to the chamber and to the mixer. VOM validates the flow paths for all five operations with channel-level pressurization protocol, based on which it detects two redundant control channels that can be safely removed from the design, as shown in Figure 4.1.

For the second design, we tested two input protocols:

- The first protocol consists of six fluid transportation operations, each starts from an inlet and ends at a half-ring-shaped channel inside a mixer. VOM validates the flow paths for all six operations with channel-level pressurization protocol, and the synthesis results show no redundancy.

- The second protocol consists of two fluid-multiplexing operations, which is more in accordance with the intention of the design (since the design consists of three parallel mixers). For each fluid-multiplexing operation, VOM automatically updates the control channel pressurization protocols twice based on the simulation results. Figure 4.2 illustrates the updates for the first fluid-multiplexing operation, which aims to transport fluids from an inlet to three half-ring-shaped mixer-segments simultaneously. At the beginning of the operation, two control channels are pressurized to form the flow paths without

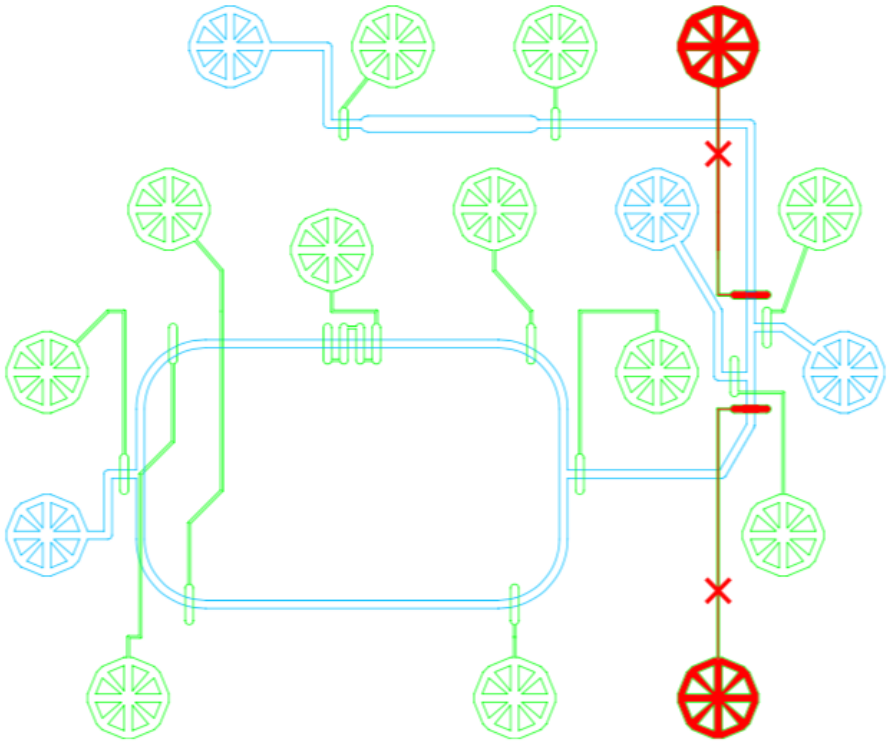


Figure 4.1.: A design consisting of a chamber and a mixer. Flow channels are in light blue and control channels are in green. The two control channels in red are redundant and can be safely removed from the design.

contaminating the other halves of the mixers, as shown in Figure 4.2(a). When the channel segment in the nearest mixer is filled, VOM detects that there are still two other sub-paths in process, as shown in Figure 4.2(b). Thus, it pressurizes another control channel to stop the fluid transportation in the finished sub-path without disturbing the others, as shown in Figure 4.2(c). Similarly, after fluids reach the middle mixer, VOM pressurizes another control channel to ensure the longest sub-path, as shown in Figure 4.2(d). Besides, if the design applies the fluid-multiplexing protocol, three control channels can be safely removed.

The third design is application-specific. Thus, we generated an input protocol consisting of five fluid transportation operations based on the bioapplication (Fang et al. 2010). The protocol includes a fluid-multiplexing operation, for which VOM

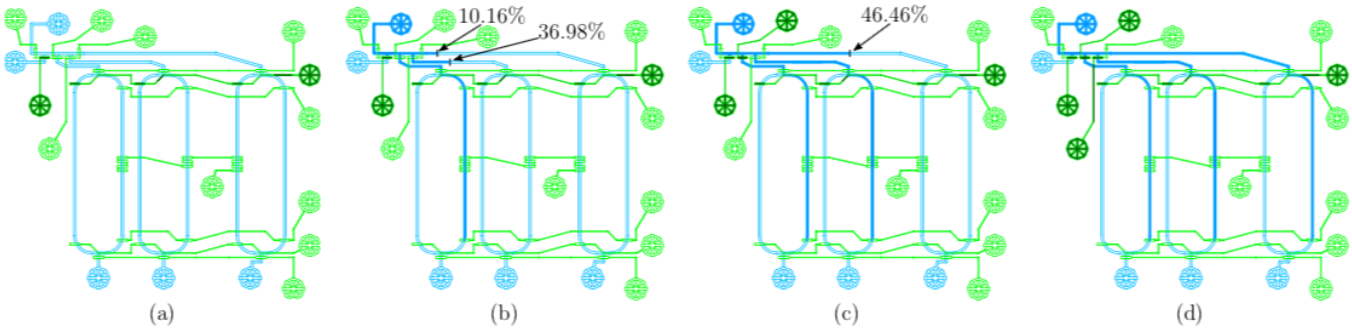


Figure 4.2.: Automatic update of channel-level pressurization protocol for a fluid-multiplexing operation. Fluids are in blue and pressurized control channels are in dark green. The numbers over the channels denote the percentage of the volume of the corresponding flow channel segment between two landmarks (valves in this case) that has been occupied by fluids.

updates the channel-level protocol once. When VOM tries to validate the flow path for the third operation from an inlet to one of the mixer, it finds out that there is no control channel pressurization option that can form the required path, due to two sequentially connected valves in two conflicting sub-paths. Thus, we added the fifth test case assuming one additional control channel to pressurize the conflicting valves separately. In this case, VOM validates the flow paths for all five operations with channel-level pressurization protocol and detects two redundant control channels.

For all three designs, VOM synthesizes the channel-level protocol within seconds. The program run time increases with the scale of the design and the protocol.

5. Conclusion

Design automation for multilayered continuous-flow microfluidic biochips has made significant progress in the last decade. However, there is a mismatch between the state-of-the-art high-level synthesis and physical design approaches, which hinders the development of a complete automatic synthesis flow. This work proposes VOM to build the missing link with flow path validation and control sequence optimization. VOM takes a biochip design and a high-level protocol as inputs, synthesizes, and optimizes channel-level pressurization protocols to validate the dynamic formation of required fluid transportation paths. For fluid-multiplexing operations, VOM automatically updates the high-level protocol, and synthesizes the corresponding channel-level protocol based on an event-driven simulation approach. We demonstrate the efficiency and the effectiveness of VOM with multiple biochip designs.

In general, VOM can efficiently generate channel-level pressurization sequences for feasible designs and protocols, and locate the conflicts for infeasible designs and protocols. It can also detect redundant resource usage of the input design, which opens up a new direction to improve the performance and the feasibility of customized biochips.

Bibliography

Crites, Brian, Kong, Karen & Brisk, Philip (2017): Diagonal component expansion for flow-layer placement of flow-based microfluidic biochips, ACM Trans. on Embedded Comput. Syst. **16**(5s): 126:1–126:18.

Fang, Cong, Wang, Yanju, Vu, Nam T., Lin, Wei-Yu, Hsieh, Yao-Te, Rubbi, Liudmilla, Phelps, Michael E., Müschen, Markus, Kim, Yong-Mi, Chatziioannou, Arion F., Tseng, Hsian-Rong & Graeber, Thomas G. (2010): Integrated microfluidic and imaging platform for a kinase activity radioassay to analyze minute patient cancer samples, Cancer Res. **70**(21): 8299–8308.

Firpo, G, Angeli, E, Repetto, L & Valbusa, U (2015): Permeability thickness dependence of polydimethylsiloxane (pdms) membranes, Journal of Membrane Science **481**: 1–8.

Graham, Ronald L (1966): Bounds for certain multiprocessing anomalies, Bell System Technical Journal **45**(9): 1563–1581.

Hu, Kai, Dinh, Trung Anh, Ho, Tsung-Yi & Chakrabarty, Krishnendu (2017): Control-layer routing and control-pin minimization for flow-based microfluidic biochips, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **36**(1): 55–68.

Hu, Kai, Yu, Feiqiao, Ho, Tsung-Yi & Chakrabarty, Krishnendu (2014): Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration, **33**(10): 1463–1475.

Lee, Chung-Cheng, Sui, Guodong, Elizarov, Arkadij, Shu, Chengyi Jenny, Shin, Young-Shik, Dooley, Alek N., Huang, Jiang, Daridon, Antoine, Wyatt, Paul, Stout, David, Kolb, Hartmuth C., Witte, Owen N., Satyamurthy, Nagichettiar, Heath, James R., Phelps, Michael E., Quake, Stephen R. & Tseng, Hsian-Rong (2005): Multistep synthesis of a radiolabeled imaging probe using integrated microfluidics, Science **310**(5755): 1793–1796.

Li, Mengchu, Tseng, Tsun-Ming, Li, Bing, Ho, Tsung-Yi & Schlichtmann, Ulf (2016): Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations, S. 624–629.

Li, Mengchu, Tseng, Tsun-Ming, Li, Bing, Ho, Tsung-Yi & Schlichtmann, Ulf (2017): Component-oriented high-level synthesis for continuous-flow microfluidics considering hybrid-scheduling, Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEE, IEEE, S. 1–6.

Liu, Chunfeng, Li, Bing, Yao, Hailong, Pop, Paul, Ho, Tsung-Yi & Schlichtmann, Ulf (2017): Transport or store?: Synthesizing flow-based microfluidic biochips using distributed channel storage, Proceedings of the 54th Annual Design Automation Conference 2017, ACM, S. 49.

Melin, J. & Quake, S. (2007): Microfluidic large-scale integration: the evolution of design rules for biological automation, Annu. Rev. Biophys. Biomol. Struct. **36**: 213–231.

Minhass, Wajid Hassan, McDaniel, Jeffrey, Raagaard, Michael, Brisk, Philip, Pop, Paul & Madsen, Jan (2018): Scheduling and fluid routing for flow-based microfluidic laboratories-on-a-chip, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **37**(3): 615–628.

Minhass, Wajid Hassan, Pop, Paul & Madsen, Jan (2011): System-level modeling and synthesis of flow-based microfluidic biochips, Proc. Int. Conf. Compil., Arch. and Syn. Embed. Sys., S. 225–234.

Oh, Kwang W, Lee, Kangsun, Ahn, Byungwook & Furlani, Edward P (2012): Design of pressure-driven microfluidic networks using electric circuit analogy, Lab on a Chip **12**(3): 515–545.

Padberg, Manfred & Rinaldi, Giovanni (1991): A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems, SIAM review **33**(1): 60–100.

Stanford Foundry (n.d.): Basic Design Rules, http://web.stanford.edu/group/foundry.

Stone, Howard A (2007): Introduction to fluid dynamics for microfluidic flows, CMOS Biotechnology, Springer, S. 5–30.

Tseng, Tsun-Ming, Li, Bing, Schlichtmann, Ulf & Ho, Tsung-Yi (2015): Storage and caching: Synthesis of flow-based microfluidic biochips, IEEE Design & Test **32**(6): 69–75.

Tseng, Tsun-Ming, Li, Mengchu, Freitas, Daniel Nestor, McAuley, Travis, Li, Bing, Ho, Tsung-Yi, Araci, Ismail Emre & Schlichtmann, Ulf (2018): Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **37**(8): 1588–1601.

Tseng, Tsun-Ming, Li, Mengchu, Freitas, Daniel Nestor, Mongersun, Amy, Araci, Ismail Emre, Ho, Tsung-Yi & Schlichtmann, Ulf (2018): Columba S: a scalable co-layout design automation tool for microfluidic large-scale integration, 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), IEEE, S. 1–6.

Tseng, Tsun-Ming, Li, Mengchu, Li, Bing, Ho, Tsung-Yi & Schlichtmann, Ulf (2016): Columba: Co-layout synthesis for continuous-flow microfluidic biochips, Proceedings of the 53rd Annual Design Automation Conference, ACM, S. 147.

Unger, M. A., Chou, H.-P., Thorsen, T., Scherer, A. & Quake, S. R. (2000): Monolithic microfabricated valves and pumps by multilayer soft lithography, Science **288**(5463): 113–116.

Wu, Angela R., Hiatt, Joseph B., Lu, Rong, Attema, Joanne L., Lobo, Neethan A., Weissman, Irving L., Clarke, Michael F. & Quake, Stephen R. (2009): Automated microfluidic chromatin immunoprecipitation from 2,000 cells, Lab on a Chip **9**: 1365–1370.

Zhu, Ying, Li, Bing, Ho, Tsung-Yi, Wang, Qin, Yao, Hailong, Wille, Robert & Schlichtmann, Ulf (2018): Multi-channel and fault-tolerant control multiplexing for flow-based microfluidic biochips, Proceedings of the International Conference on Computer-Aided Design, ACM, S. 123.