DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Visualization of the Fluid Behavior on Microfluidic Large-Scale Integration Biochips

Jing Huang

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Visualization of the Fluid Behavior on Microfluidic Large-Scale Integration Biochips

Visualisierung des Fluidverhaltens auf Microfluidic Large-Scale Integration Biochips

Author:	Jing Huang
Supervisor:	Prof.Dr.-ing Ulf Schlichtmann
Submission Date:	15.12.2019

I confirm that this bachelor’s thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.12.2019   Jing Huang

Acknowledgments

I would like to express my deep gratitude to Prof. Dr.-Ing. Ulf Schlichtmann, head of chair electronic design automation, and to Dr.-Ing. Tsun-Ming Tseng, research and teaching assistant at the chair of electronic design automation, for offering me such an opportunity to work on this interesting topic. I would like to especially thank my supervisor, Ms. Mengchu Li, for her patient guidance, continuous encouragement and useful critiques in carrying out this work.

I would also like to extend my thanks to my friends who give me technical support and assistance in keeping my progress on schedule.

Abstract

Microfluidic biochips are miniaturized laboratories allowing thousands of bio-operations to be performed in parallel. However, they possess structures with high complexity, which triggers the development of design automation for biochips. Current design automation approach is capable of automatically producing the physical design of microfluidic biochips, scheduling the operations and binding operations to devices. Furthermore, a design automation tool called VOM provides the first approach to construct valid fluidic transportation paths dynamically. This thesis is an extension of VOM that visualizes the simulation results from VOM, so that experimenters could see the simulated process of bio-assay execution on microfluidic biochips.

Contents

1 Introduction

Microfluidic large-scale integration is a technology, which manipulates the flow of tiny amounts of fluids in the network of microchannels. This technology enables the construction of multilayer continuous-flow microfluidic biochips, which are essentially minimized labs. Due to the compact size of biochips, experiments are able to be conducted cheaper and faster than with conventional laboratory instruments[1]. In addition, in order to increase throughput, biochips also allow parallelization and multiplexing[2]. Hundreds or thousands of reactions can be performed simultaneously on a biochip. Besides, biochips can be automatically operated by software, and meanwhile, maintain the performance with precise and simple control over parameters. Because of these advantages, this technology gains popularity in the past few decades. There are many applications of it in different fields, such as chemistry, biology, and medical treatment[3, 4, 5, 6].

Multilayer continuous-flow microfluidic biochip is constructed with two layers of transparent elastomers using soft lithography technology[7]. Each layer is integrated with microchannels, and there is a flexible membrane between the two layers. This structure enables the formation of micro-valves. Figure 1.1 shows a polydimethylsiloxane (PDMS) push-down valve, where the layer with the control channel sits above the layer with the flow channel. Flow channel has a rounded profile, when the control channel is pressurized, it will push the membrane downwards to fit precisely in the flow channel. In this case, the fluid transportation inside the flow channel will be blocked[8]. A control channel can be pressurized with external pressure source via a control inlet. If a control channel is pressurized, valves on this channel are closed; otherwise, valves on this channel are open. Valves divide flow channels into small chambers, which can further build more complex microfluidic devices such as mixers.[9].

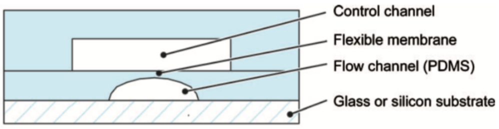


Figure 1.1: A two-layer polydimethylsiloxane (PDMS) push-down microfluidic valve[8].

However, as microfluidic devices become more complicated, manually designing microfluidic biochips tends to be more time-consuming and error-prone. Hence, researchers have worked on constructing an automated software synthesis flow, which can automatically produce an optimized physical chip design and an explicit protocol for executing bio-applications. So far, design automation approaches for physical design and high-level synthesis have been put into practice[10, 11, 12, 13]. However, some connections between the design and protocols are still missing. In particular, current works assume that fluid transportation paths are always valid. This assumption does not always hold and may cause inaccuracy in resource usage and harm the experimental performance.

Mengchu Li et al. recently proposed a design automation method called VOM to validate the flow path and to optimize the control channel pressurization sequence. However, the outputs of VOM are in plain-text and are not intuitive enough for the users. Users need to learn to interpret the results to ensure that the process happening inside of the biochips is designed as requested. Therefore, a visualization tool is in need to display a bio-chip design and animate transportation operations. Specifically, after inputting a schedule of fluid transportation operations, a user should be able to see the fluid movements in the flow channels and the pressure status in control channels with respect to time changes.

In general, the primary purpose of this thesis is to build a simulation tool to visualize the final outputs of VOM with images and animations. In the following, I would like to introduce the background knowledge related to my topic, then demonstrate an overview of my solution approach and the implementation process. Subsequently, I would review my experiment results.

2 Background

2.1 VOM

2.1.1 The objectives of VOM

Firstly, current research does not provide explicit sequence to pressurize control channels, the absence of which prevents the validation of a dynamically constructed flow path. As valves conduct the fluid movement in actual deployment, it is impossible to examine whether the fluid could reach out to the outlet following the target flow path without corresponding protocol. Even when the physical connection of flow channels forming a target flow path is valid, the flow would be blocked in the chip if a valve in this path is set to be off[14].

Second, control channel pressurization protocol and flow paths might need to be dynamically updated when implementing fluid-multiplexing operations. For instance, in Figure 2.1, there are two flow sub-paths: path 1-2-5-3 and path 1-2-4. Considering that path $(2, 3)$ is shorter than path $(2, 4)$, the fluid from M1 will reach M3 earlier than the other mixers. In case the whole assay fails due to inaccurate scheduling, the valve on the path $(5, 3)$ is suggested to be turned off to block the flow transportation on this segment.

Besides, VOM could also detect redundant design to decrease the waste of resources.

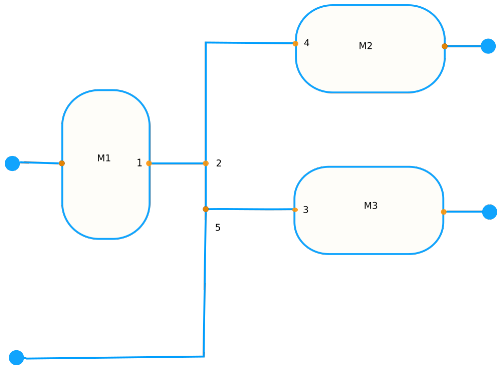


Figure 2.1: A bio-chip supports multiplexing from M1 to M2 and M3.

2.1.2 The method of VOM

VOM implements a simulation-based approach. Figure 2.2 presents an overview of the algorithmic flow of VOM. The inputs of VOM consist of a bio-chip design and a high-level protocol describing fluidic transportation operations in sequence. VOM first checks the compatibility of the flow layer. If flow layer design cannot support the target flow paths, it will throw a warning exception. Otherwise, VOM would move to the next step to build an optimal sequence to pressurize the control channel. If no such a sequence can construct the target flow path, a warning would be produced. Otherwise, it will test all potential channel-level protocols and choose the optimized one based on user-defined optimization criteria: resource-oriented and transportation-time oriented, then simulate the application process in an event-driven manner. When a fluid-multiplexing operation is detected, VOM will update the target flow path of current transportation operation in high-level protocol automatically and repeat this process until no update is required. Finally, VOM creates a channel-level protocol, a schedule for each fluid transportation operation, and a report on resource usage in the control layer as outputs.

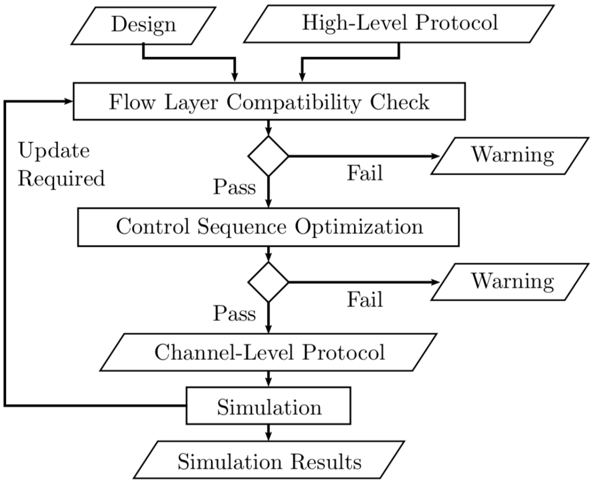


Figure 2.2: System diagram of VOM[14].

2.2 Design of the bio-chip

Understanding the design of the bio-chip is the most primary task to build a specialized simulation tool. The design refers to the floor-plan of the bio-chip. It introduces the structures of essential elements and principal components to be visualized, and some essential design rules applied in this test case from Columba 2.0[15].

2.2.1 Essential elements

Inlets, outlets, flow channels, control channels, valves, intersection points of two flow channels are the most fundamental elements of the bio-chip design. Inlets/Outlets are holes bond with external sources and channels. Reagents could enter the flow channel via flow inlet and exit from the flow outlet. Control inlet introduces gases to the control channel. In this thesis, the flow channel is colored with blue indicating fluids, and the default color of the control channel is green. In regards to valves, they are bound with control channels and flow channels simultaneously, so that they could compartmentalize the compounds in the flow channel when the control channel is filled with gas. Figure 2.3 shows that valves colored with orange reside on the control channel. Besides, for the convenience of description, this thesis would name the intersection point as a branch in the following context.[16].

Design rules

According to the design rules of Stanford Foundry[17], this thesis assumes the width of the flow channel and valves to be $100\mu m$, and the width of the control channel is set relatively smaller: $30\mu m$, but, flow/control inlet is much bigger, with the size of $1mm \times 1mm$.

2.2.2 Flow structure: principal components

- Mixer
  As one of the most essential components in the microfluidic devices. Mixer is mainly responsible for efficient mixing. It has a ring-shaped flow channel, whose two opposite sides (either left-right or up-down) are often connected with two more flow channels, a peristaltic pump formed by three or four interconnected valves, another six position fixed valves as well as corresponding control channels(see Figure 2.3). This structure helps to accelerate mixing process. In Figure 2.4, two different reagents load in the mixer with closed valves at the bottom and top. After peristaltic pumps had been actuated, two reagents moved circularly along the ring-shaped flow channel and started mixing, which is apparently faster

than passive diffusional mixing [2]. Additionally, mixer is asymmetric, which implies that a mixer could be transformed into four different mixers with rotation.

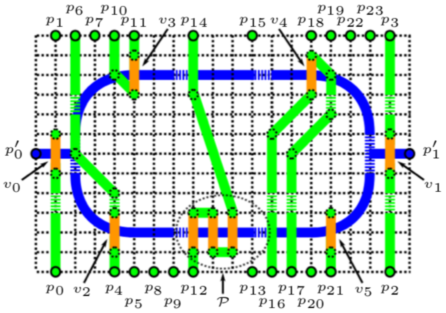


Figure 2.3: A mixer with two flow channels on the leftmost and rightmost, six valves $v_0, v_1, v_2, v_3, v_4, v_5$ at fixed position $p_0, p_1, p_4, p_{11}, p_{18}, p_{21}$ respectively, and a peristaltic pump p at p_{12}[15].

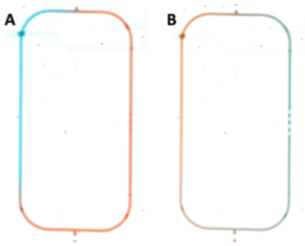


Figure 2.4: Red represents one reagent, and blue represents another reagent. A: before mixing B: during mixing(See Supplementary Video ”RotaryMixerMixing.gif” [18]))[15].

- Reaction chamber
  A reaction chamber contains one main flow channel and two valves on the end of it: an inlet valve and an outlet valve. The size of the chamber depends on the requirement of the application.
- Switch
  A switch utilizes valves to guide the direction of the fluid movement when flow channels intersect with each other. There is a main flow channel with two valves at both ends and multiple flow channels growing straightly from one side of the main flow channel. However, these side flow channels are never across through

the main flow channel. Apart from this, each side flow is bound with exactly one valve.

2.2.3 Control structure: pressure control

If one control channel links with only one valve, as the amount of valves grows with the increasing complexity of the application, more control inlets should be appended to support control channels. In order to reduce the usage of control inlets, a better approach is to connect a control channel with a group of valves from different components to share pressure[19] [20]. As figure 2.5 shows, each control channel links multiple valves. Therefore, pressurizing one control channel will close all valves along this control channel sequentially.

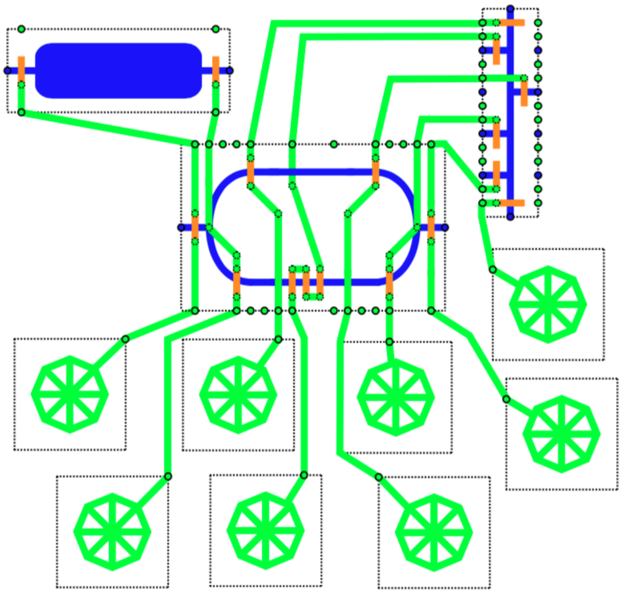


Figure 2.5: Part of a biochip design where the control inlets connect valves of different components[15].

3 Research methodology

3.1 Objective

The desired simulation tool aims to parse and simulate a series of scheduled flow transportation operations from VOM, each of which specifies a set of fluid transportation paths as well as corresponding due time, and a channel-level protocol defining the sequence of pressurized control inlets.

fluid transpiration path

As except for inlets and outlets, only the status of valves(open/closed) or multiplexing(branches) could influence the direction of the fluid movement in the flow channel, the liquid transportation path is interpreted with a unique ordered pair of every two landmarks(valve, inlet, outlet, branch) and the percentage of total length in-between. Besides, for the sake of simple and accurate description, these landmarks are preferred to be specified with identifiers rather than with coordinates.

Therefore, the first task of this simulation tool is to recognize the path within the directed segment between landmarks given their identifiers and length-percentage, so that users could locate any segment on the design.

corresponding due time

The flow movement following the transportation path is intended to be animated within the given time to visualize the rate of the fluid since different kinds of liquids would encounter various hydraulic resistance at different positions of the channel.

control inlet

When a sample of gas is introduced to control inlet, its molecules could disperse throughout the control channel within one second. After the connected control channels and all valves along them are distinguished given the identifier of a control inlet, this process could be simulated by coloring this path instantly.

3.2 The overview of the approach

Another extension project of VOM made by YanLu Ma can already allocate each landmark a distinct integer and display the design of bio-chip with the indices of landmarks on it in JPEG(see Figure 3.3). However, a JPEG image can support neither interaction nor animation as demanded. This simulation tool would be then built with the two layers to achieve the objectives specified above. Figure 3.1 manifests a general procedure of this project.

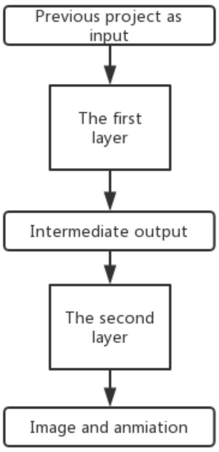


Figure 3.1: Flowchart of this project.

3.2.1 The first layer

In order to situate an arbitrary transportation path, it is necessary to make segments bound by landmarks to be accessed by a pair of identifiers separately. The main idea is to map each landmark to its allocated identifier and then explore all directed segments between two adjacent landmarks, which denote the initial point and the terminal point of this segment. It could be realized by analyzing the parsed design of bio-chip in C++.

Apart from paths of flow channels between landmarks, paths of control channels controlled by control inlets would also be extracted from this layer, which automatically links the paths and identifiers together.

Finally, it would produce an analyzed design of bio-chip, whose segments accord with landmarks.

3.2.2 The second layer

In the second layer, this approach attempts to present the analyzed design of bio-chip on an interactive web page with JavaScript, so that requests from users could be dynamically processed. After parsing the request and identifying the required percentage of a segment, the transportation path within the segment can be further located in this layer.

The design is drawn using SVG, which can be manipulated by JavaScript. SVG not only supports animation but also allows the user to scale the design to any size without losing quality. Therefore, running liquids and gases can be visualized with high resolution.

3.2.3 Intermediate output

Due to the incompatibility of C++ and JavaScript, the analyzed design of bio-chip generated in the first layer would be stored in a plain-text file as an output, which would be then passed to the second layer and then parsed via JavaScript.

It is preferred to directly define the path as a string following the rule of the element <path> of SVG in the first layer, since <path> could draw both straight edges and curved edges. Besides, <path> is inherently specified with prompts and a set of coordinates, which exempts from the self-defined signs to distinguish the direction and type of edges. It further reduces the complexity of parsing processing of the second layer.

Path

Draw graphics with <path>	
basic shapes	<path>
straight edge (a, b)	M x_a y_a L x_b y_b
arc (a, b)	M x_a y_a A rx ry x-axis-rotation large-arc-flag sweep-flag x_b y_b

Table 3.1: Draw straight and curved segment with <path>

M, *L*, and *A* are signs of commands to draw paths. *M* means "Move to", *L* refers to "Line to". When drawing an straight edge from point *a* to point *b*, it first moves to (x_a, y_a) and then draws a line to (x_b, y_b).

A indicates an arc, a section of an eclipse, or a circle with radius *rx* and *ry*. $x - axis - rotation$ determines the rotation of this arc. If the arc is greater than 180 degrees, then $large - arc - flag$ sets to 1, otherwise, 0. If an arc draws clockwise from point a to point b, then sweep-flag is 1, otherwise 0.

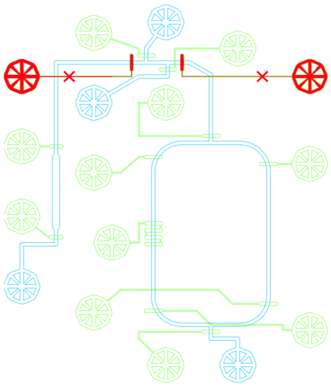


Figure 3.2: The original design of bio-chip specified with AutoCAD

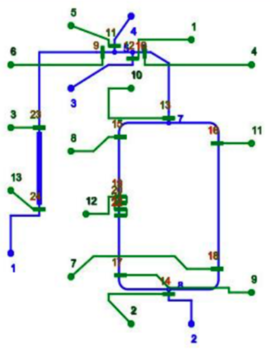


Figure 3.3: The parsed design of bio-chip with landmarks from the project of Yanlu Ma

3.3 Input

In the project of Yanlu Ma, the original design of bio-chip (see Figure 3.2) specified with an AutoCAD-compatible script has been parsed into a set of arrays in C++. The first layer takes the following outputs from this project as inputs: the parsed design of bio-chip, the identifiers of landmarks as well as the width and the length of the flow channels. Although the width of the flow channel is given following the design rules of Stanford Foundry(see section 2.1), due to its flexibility in the reaction chamber, directly applying the default value would lead to an error.

The parsed design of bio-chip

The parsed design of bio-chip interprets its structure into an undirected graph consisting of a set of vertices and a set of edges, each of which is defined through an unordered pair of vertices. Each vertex might represent a flow channel intersection or a branch, an inlet, and an outlet. In terms of an edge, it could indicate a flow channel, a control channel, or a valve. A valve connects with a flow edge at its middle point.

Moreover, flow channels could construct different components of the design of bio-chip, which results in a variety of combinations of edges.

- Mixer:
  As Figure 3.4 shows, a ring-shaped flow channel could be seen as ten edges with

ten vertices. In this test case, edges {2, 3}, {4, 5}, {7, 8}, and {9, 10} are quarter circles, whose radius is 283.3μm, while the rest edges are straight. As the edges on the left side of {1, 6} are symmetric with on the right side, only the coordinates of point a and point b are provided in the input to infer the whole structure of the mixer.

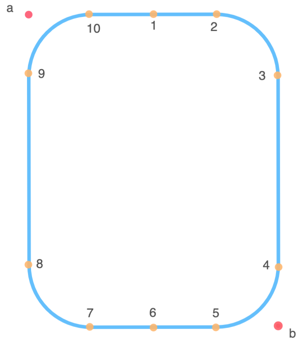


Figure 3.4: Ring-shaped flow channel of the mixer.

- Switch:
  The main channel and side channels of a switch are provided independently in inputs. For instance, edge {1, 6} is the main channel, and other edges {2, 7}, {3, 8}, {4, 9}, {5, 10} are side channels(Figure 3.5).

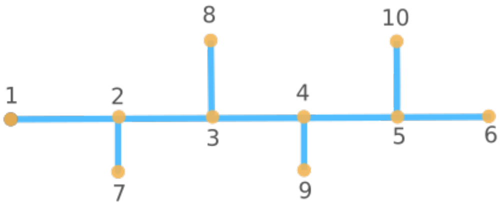


Figure 3.5: Flow channels of a switch.

- Chamber:
  A chamber contains a thicker edge {2, 3} as the main reaction chamber because of adjustable width and another two edges {1, 2} and {3, 4} for the delivery of the liquids(Figure 3.6).

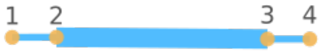


Figure 3.6: Flow channels of the chamber.

- The rest channels:
  The rest channels are edges used to bind different components. Some of them might be horizontal, some might be vertical, and others might be oblique.

3.4 Terminology explanation

In this following context, I will use "edge", "segment", and "path" to describe different relationships of vertices in different stages.

An edge refers to an unordered pair of vertices denoted with curly braces. In Figure3.7(a), edge $\{3, 8\}$ implies the connection between vertex 3 and vertex 8. And a segment represents a sequence of directed edges without direction. For instance, a segment between vertex 3 and vertex 10 in Figure3.7(b) is constructed via $((3, 8), (8, 9), (9, 10))$ or $((10, 9), (9, 8), (8, 3))$ rather than via edge $\{3, 10\}$, which does not even exist in Figure3.7. Path is the description of a directed segment or a directed edge. It is defined by <path> of SVG. In Figure3.7(c), path $(10, 11)$ specifies a directed straight edge from vertex 10 to vertex 11. It could be described by "M x_{10} y_{10} L x_{11} y_{11}".

(a) $\{1, 5\}$, $\{2, 6\}$, $\{4, 7\}$, $\{3, 8\}$, $\{8, 9\}$, $\{9, 10\}$

(b) $\{1, 5\}$, $\{2, 6\}$, $\{4, 7\}$, $(3, 10)$ or $(10, 3)$

(c) $\{1, 5\}$, $\{2, 6\}$, $\{4, 7\}$, $(10, 11)$, $(11, 3)$

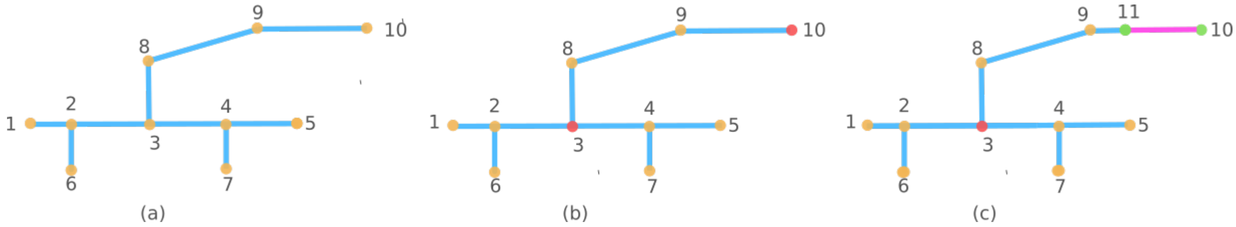


Figure 3.7: Yellow indicates regular vertices, red means landmarks, greed implies origin or destination. (a): The portion of the parsed design from previous project. (b): Segment $(3, 10)$ or $(10, 3)$ is extracted from the first layer. (c): The flow movement along path $(10, 11)$ in the second layer.

4 Implementation

4.1 Map landmarks to identifiers

A valve can either be a horizontal rectangular or a vertical rectangular(Figure 4.1(a), (d)). However, in the flow layer structure, a valve works as a vertex to separate the flow segment(Figure 4.1(c), (f))), while in the control layer structure, it is an edge to bind control channels(Figure 4.1(b), (e)).

Since most flow segments join valves at their middle points, I regard this specific vertex as a landmark. Then I map all landmarks, including the middle vertices of valves, branches, inlets, and outlets to their corresponding identifiers so that a vertex could be identified as a landmark if it exists in this map.

Besides, even though the length of the valve is predefined, its two endpoints could not be located with this middle point due to the uncertainty of the exact shape. Therefore, the valves will be stored in the form of an edge and a vertex simultaneously.

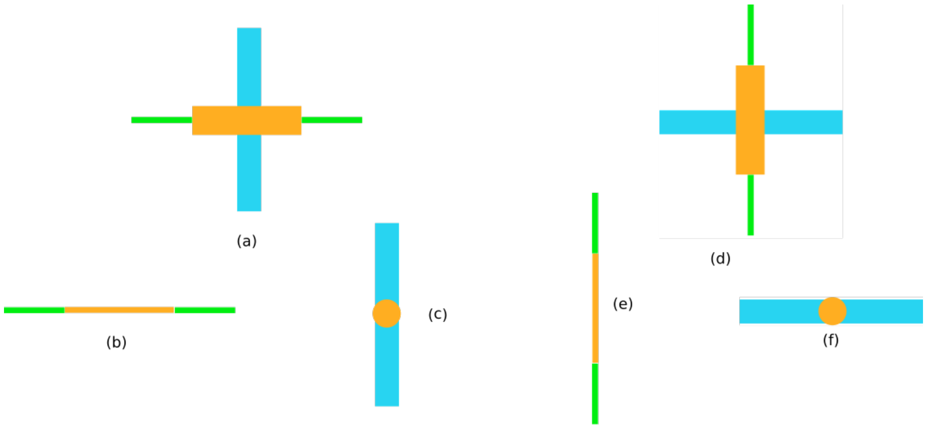


Figure 4.1: Different representations of the valve in different channels. Valve: orange, Flow channel: blue, Control channel: green.

4.2 Extract flow paths between landmarks

4.2.1 Mixer

Due to the mixer's structure, landmarks here refer to branches and valves. Each mixer possesses exactly two branches. As for valves, though in the current design of biochip, four valves of mixers are position-fixed, the simulation tool assumes that valves could be placed anywhere to maintain the robustness and flexibility.

The branches divide the whole ring-shaped flow channels into two symmetric sections, either left-right or up-down, which apply various criteria to distinguish straight edges and curved edges when extracting the paths. Besides, curved paths are dependent on the position of the circle. For instance, for a left-right mixer, if the arc is on the left of a circle, it is drawn counterclockwise from the initial point to the terminal point with $sweep - flag$ set to 1; otherwise, clockwise with $sweep - flag$ set to 0. In the following, I will explain the method to extract flow paths between landmarks on a left-right mixer and apply the analog method on the up-down mixer.

Gather all candidate landmarks

The first step to locate an oriented segment is to gather all candidate landmarks. In Figure 3.4, the landmarks in the rectangular, whose diagonal is edge $\{6,\ a\}$, are precisely the candidate landmarks lying on the flow channels of the left-half mixer. To extract the desired path, I sort all candidate landmarks along the flow channel towards one direction to locate all pairs of adjacent landmarks. Besides, different sections of the flow channel implement different sorting strategies. If the extracting process starts from point 1, the landmarks in the upper part with directed edge $(1,\ 9)$ are sorted by decreasing x-coordinates, while the landmarks in the lower part with directed edge $(8,\ 6)$ are in an order with increasing x-coordinates. The candidate landmarks in the middle section with directed edge $(9,\ 8)$ will be ordered according to the increasing y-coordinates.

Determine paths between candidates

Algorithm 1 explains the method to acquire paths between landmarks in the upper-left part of the mixer (Figure 3.4). In the beginning, the first candidate is set to be the initial point. If a second candidate exists, there appear to be three variants of paths bounded by these two candidates in this section: completely straight, partially straight and partially round, or completely round. After identifying the representation of each variant, the second candidate would replace the first candidate as the initial point of the next path. Generally, the previous candidate is the starting point, and the current candidate is the terminal point. This process would repeat continuously until no more candidate exists. In the end, the last candidate would be the starting point of the next section.

Furthermore, to make the arc to be drawn accurately, a function is deployed to calculate the actual coordinate of the valve connecting with the flow channel based on the coordinate of the valve's center point and the radius of the circle.

The following demonstrates how to define the representation of a path when the terminal point situates in the different sections of the left-half mixer. As the starting point might not lie in the same section with the terminal point, the complexity of the path grows with the section(Figure Figure 3.4).

- upper-left part

  If a candidate lies on edge $\{1,\ 10\}$, a path could be specified with a directed straight edge from the previous candidate to the current candidate. If not, it then depends on the position of the former candidate. When the last candidate is on the left of point 10, this path is in the shape of an arc starting from the last candidate to the current candidate. Otherwise, a target path would be depicted by a directed edge from the last candidate to point 10 and an arc from point 10 to the current candidate.

- middle-left part

  The middle-left part comes right after the upper-left part. If the number of candidate landmarks in this part is more than zero, for the first candidate, the target path consists of a path from the last candidate to point 9 in the upper-left part as well as a directed line segment from point 11 to the current candidate. For the rest candidates, a path in-between is the line segment from the last candidate to the current candidate.

- lower-left part

  The first path from the last candidate to the current candidate is relatively complicated to define since the previous candidate could either locate in the upper-left part or the middle-left part. In this case, after an incomplete path from the last candidate to point 8 belonging to the sections above is induced, it will then append the path from point 8 to the current candidate of this part. The further possible paths are entirely on the inside of this part.

Algorithm 1 Extract paths in upper-left

```
Input: All valves and branches in upper-left: $valves$
       Joint point of straight segment and round segment: $joint$
       starting point of a segment: $start$
       starting point of a actual segment: $start_{actual}$
       Actual end point of a arc: $actual$
$start \leftarrow valves[0]$;
$start_{actual} \leftarrow start$;
if candidates is not empty then
    for $i = 1$ to (candidates.size - 1) do
        if $candidates[i]$ lies on straight segment then
            add $(start, candidates[i])$ into segments;
            $start_{actual} \leftarrow candidates[i]$;
        else
            $actual \leftarrow getActualPoint(candidates[i])$;
            if start lies on the straight segment then
                add $(start_{actual},\ joint,\ actual)$ into segments;
            else
                add $(start_{actual},\ actual)$ into segments;
            end
            $start_{actual} \leftarrow actual$;
        end
        $start \leftarrow candidates[i]]$;
    end
end
```

4.2.2 The general flow edges

The general flow edges include the edges constructing switches and chambers, and the edges connecting components. They do not process any arcs and thus could be grouped together. Since an edge might have no or multiple landmarks on it, it is necessary to build connections among these flow edges to extract paths between two adjacent landmarks. For instance, in Figure3.7, {3, 8}, {8, 9}, {9, 10} are initially independent edges. The edge {3, 8} has to be first connected with edge {8, 9}, and then with {9, 10}.

Besides, all flow channels are aimed to be split into multiple segments bounded by two distinct landmarks. As long as an endpoint of a segment could not be identified to be a landmark, this segment would continuously connect with the other flow edges until both endpoints of the newly merged directed segment are detected as landmarks.

I call this process as searching process. To finish it, the first and the last connected edge in the sequence must start or end with a landmark. It indicates that some edges provided in the input are bound with a landmark and a point. If this algorithm starts searching from these landmarks(root landmarks), it could be able to find all target paths between a pair of landmarks.

Locate all root landmarks

Except for valves, which are designed to lie in the middle of the edge, other landmarks could be origins or destinations of the searching process. Inlets and outlets are inherently entrances and exits of the fluid in the flow channels. As for branches, they could appear in the mixer and switch.

- Branches in a mixer: For the left-right mixer, mixer divides the general flow channels into flow channels connected with upper branches and flow channels connected with lower branches. Searching processes starting from the landmarks above the mixer will stop at the upper branches, and those processes starting from the lower branches will end at the landmarks below the mixer.
- Branches in a switch: In a switch, the main channels are stored separately from the side channels. Since the side channels are growing from branches, branches are in the middle of the leading segment and are simultaneously the endpoints of the side segments.

Extracting paths from root landmarks

Initially, all edges from the previous project will be stored twice in a map with one endpoint as key, another as value, and in reverse so that it could immediately find another endpoint when entering one endpoint. Therefore, by recursively looking for another endpoint from a searching-start landmark, the next connected edge could be located. In the meantime, I will extract all paths between landmarks on the newly merged segment.

Algorithm 2 introduces the recursive function to search all target paths with the initial point and the previous path both as the input. After the initial point is detected to be an endpoint of an edge, I sort all landmarks lying on this edge towards another endpoint. For the first landmark, if the previous path is an empty string, it implies that this recursion function is called originally with a root landmark as the initial point. In this case, the path between the root landmark and the first landmark could be directly collected. Otherwise, this path will be appended to the previous path before it is collected. In terms of the rest landmarks excluding the last landmark, a path could be defined with it and the landmark after it. As for the last landmark, if it happens to be the endpoint of the current segment, then this searching process stops; if not, this process will again concatenate the current segment and a subsequent edge to find more landmarks by calling itself recursively. If no landmarks are on the current segment, after the previous path is updated, it will also call itself.

4.3 Identify path controlled by control inlet

Each control inlet connects with a set of control edges and a set of valves working as straight edges to bind control edges in the design. Therefore, a control path controlled by a control inlet could be identified by constantly combining these edges until no succeeding edges exist. This searching function could also be implemented via recursion and applied to all known control inlets.

Besides, if an edge is confirmed to be a valve during the extraction, an edge will be transformed to a rectangular(see Algorithm 3).

Algorithm 3 Process to acquire path

```
Input: Starting point of a segment: start
       End point of a segment: end
       Valve: valve
if the edge is vertical then
    $valve \leftarrow (start.x,\ (start.y + end.y)/2)$;
end
if the edge is horizontal then
    $valve \leftarrow ((start.x + end.x)/2,\ start.y)$;
end
if valve is a landmark then
    $path \leftarrow path + rectangular(valve)$;
else
    $path \leftarrow path + getPath(start,\ end)$;
end
```

Algorithm 2 SEARCHFLOW(*start*, *previousPath*)

```
Input:
        Detected landmarks on a segment: landmarks
        Starting point of a segment: start
        End point of a segment: destination
if there is edge with endpoint start then
    for each detected edge do
        if edge is not connected yet then
            if edge(start, destination) has landmarks then
                if previousPath is empty then
                    add (start, landmarks[0]) into segments;
                else
                    path ← previousPath + getPath(start, landmarks[0]);
                    add path into segments;
                end
                tmp ← landmarks[0];
                for i = 1 to (landmarks.size - 1) do
                    add (tmp, landmarks[i]) to segments;
                    tmp ←landmarks[i];
                end
                if the last landmark of not equal to destination then
                    path ← getPath(landmarks[landmarks.size−], destination);
                    searchFlow(destination, path);
                end
            else
                path ← getPath(start, destination);
                if previousPath is empty then
                    searchFlow(destination, path)
                else
                    path ← previousPath + path;
                    searchFlow(destination, path);
                end
            end
        end
    end
end
```

4.4 Parse the input

After all paths between landmarks are extracted, it moves to the second layer. In this layer, the generated output from the first layer will be automatically uploaded to the website, where the user can see the image of biochip and interact with it.

Once the JavaScript program receives the uploaded file, it starts parsing strings in the file to a set of child elements of an SVG container to display the design. A flow path could be drawn by directly assigning its string to an attribute d of a new <path> element. Meanwhile, this program rebuilds one-to-one mappings from distinct paths to corresponding identifiers of landmarks in this phase. However, as the flow within two landmarks is expected to reach any reasonable extent, mapping cannot directly adopt the static flow paths in strings from the input. Instead, the path would be further parsed to a set of directed edges depicted with two vertices to allow the adjustment as the specified length changes(see Figure 4.2). For instance, path $(1,\ 2)$ could be converted to a sequence of directed edges including $(1,\ a)$, $(a,\ b)$, and $(b,\ 2)$ marked with its length additionally to inform the weight of the edge in the path. For an arc, the program also needs its angle and corresponding center point to locate its intern portion dynamically.

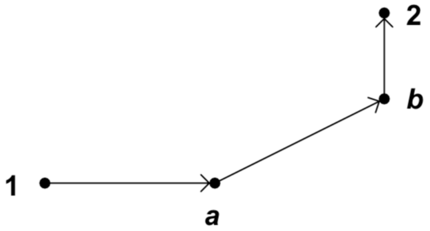


Figure 4.2: Endpoint 1 and endpoint 2 are landmarks, while point a and point b are normal vertices. This directed path from 1 to 2 contains a horizontal edge $(1,\ a)$, an oblique edge $(a,\ b)$ and a vertical edge $(b,\ 2)$. Besides, edge$(b,\ 2)$ occupied 15% of the whole path.

Preparation

- SVG: The following calculation is considered based on the SVG coordinates system. In SVG, the x-axis grows from left to right, while the y-axis grows from top to bottom.
- Degree to Radian: This explains the formula to convert degree to radiant and in

reverse.

$$degree = randian * \frac{180°}{\pi}$$

Parsing two endpoints and the central point is quite straightforward. The following explains the method to acquire additional attributes when parsing straight and curved segments.

4.4.1 Parsing the path of straight segment

- Length: For straight segments, by applying Pythagorean theorem, the length of the segment could be easily calculated. For instance, the length of path $(a,\ b)$ is $\sqrt{(a.x - b.x)^2 + (a.y - b.y)^2}$(Figure 4.2).

4.4.2 Parsing the path of curved segment

- Angle: In Figure 4.3, the centre of circle o and other two points a and b build a triangle in the circle. Additionally, the central angle α subtended by an arc L is exactly the angle of triangle - $\angle aob$, which could be computed via the law of cosines. Besides, c is the length of path $(a,\ b)$ and can be calculated by formula from section 4.4.1. Then α is:

$$cos\alpha = \frac{a^2 + b^2 - c^2}{2ab} => \alpha = \arccos \frac{r^2 + r^2 - c^2}{2r^2}$$

- Length: As the length of an arc in a circle is a portion of the circumference, the arc length can be computed with formula:

$$Arc\ length = \frac{central\ angle}{360°} * circumference$$

Thus,

$$L = \frac{central\ angle}{360°} * 2\pi * radius => \frac{\alpha}{180°} * \pi * r$$

.

4.5 Acquire flow path at given percentage

As the fluid in different portions of a parsed path might move at different speeds, to visualize flow motion in different phases, this simulation tool could dynamically access target fluid transportation paths with percentages. For clarity, I interpret the destination

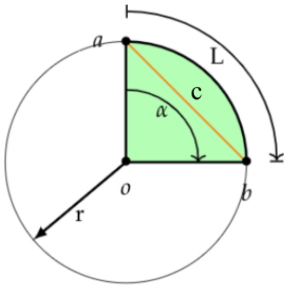


Figure 4.3: A circle with radius r. L denotes the arc length, while α is its central angle.

as the terminal point of the target path in the following context. The position of the destination varies with the specified length calculated by the multiplication of the total length of the original parsed path and percentage.

The first step to acquire the flow path at a specified length is comparing it with the length of each composed edge in the sequence and meanwhile to update itself regularly. In Figure 4.2, if the specified length L is greater than the length of the first edge $(1,\ a)$, this edge will be viewed as a sub-path of target path and deducted from the further specified path. Accordingly, the specified length L gets smaller. Otherwise, it indicates that the destination of the target path lies at a distance L away from the starting point of the current edge.

Then, the next step aims to calculate the accurate coordinate of the destination on the straight and curved edge. Since the coordinate of the destination is the offset from the initial point, different directions would result in different calculations.

4.5.1 Acquire the terminal point on a straight edge

A straight directed edge could be horizontal, vertical, and oblique. Its type and direction could be differentiated by comparing x-coordinate and y-coordinate of two endpoints.

horizontal

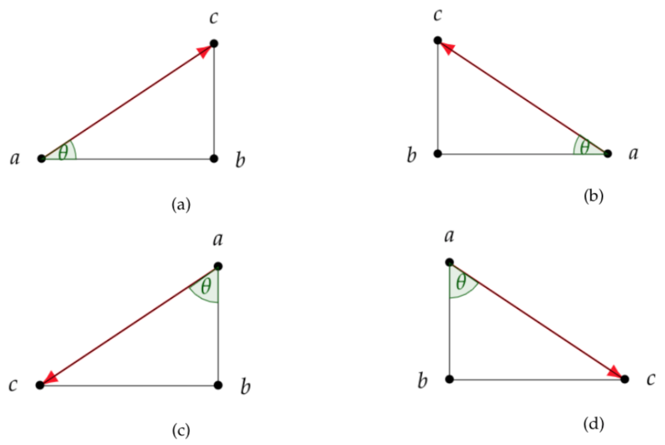
When the edge is horizontal, the destination has the same y-coordinate as the initial point. If the starting point is at the left side of the terminal end, then the x-coordinate of the destination is the x-coordinate of the starting point plus specified length; otherwise, minus.

vertical

As the edge is vertical, the x-coordinate of the destination and starting point are identical. If the starting point is below the terminal end, then the y-coordinate of the destination is the y-coordinate of the starting point plus specified length; otherwise, minus.

oblique

For an oblique edge, the offset distance is the projection of the specified length. As Figure 4.4 shows, different relative positions of the initial point and the terminal point classify the obliques edges into four groups.



(a)

(b)

(c)

(d)

Figure 4.4: Four different oblique edges starting from point a (x1, y1) to point c (x2, y2).

(a) The starting point a is on the left-below of the terminal point c in Figure 4.4(a). According to trigonometric function, a target path with specified length L on edge$(a,\ c)$ would ends at point:

$$(x1 + \cos\theta * L,\ y1 - \sin\theta * L)$$

where θ could be acquired using inverse function:

$$\tan\theta = \frac{(y1 - y2)}{(x2 - x1)} => \theta = \arctan\frac{(y1 - y2)}{(x2 - x1)}.$$

(b) The starting point a is on the right-below of the terminal point c in Figure 4.4(b). The destination would locate at:

$$(x1 - \cos\theta * L,\ y1 - \sin\theta * L)\ \text{, with } \theta = \arctan\frac{(y1 - y2)}{(x1 - x2)}.$$

(c) The starting point a is on the left-above of the terminal point c in Figure 4.4(d). The destination would locate at:

$$(x1 - \sin\theta * L,\ y1 + \cos\theta * L)\ \text{, with } \theta = \arctan\frac{(x1 - x2)}{(y2 - y1)}.$$

(d) The starting point a is on the right-above of the terminal point c in Figure 4.4(c).The destination would locate at:

$$(x1 + \sin\theta * L,\ y1 + \cos\theta * L)\ \text{, with } \theta = \arctan\frac{(x2 - x1)}{(y2 - y1)}.$$

4.5.2 Acquire the terminal point on an arc

All curved segments in the design of bio-chip are sub-segments of quarter circles. Thus, calculating the position of a destination is the same as a point on the circle.

Staring point above terminal point

There are four different arcs on the quarter circles possessing different relative positions to the origin $o(x,\ y)$, which influences the coordinate of the destination.

- top-right

  If the arc is the section of a quarter circle on the top-right corner(Figure 4.5(a), the destination c will locate at the position:

  $$(x + r * \cos\beta,\ y - r * \sin\beta)\ where\ \beta = \theta - \alpha.$$

  Besides, with adjacent side od and hypotenuse side oa, θ can be measured via trigonometric function:

  $$\cos\theta = \frac{x_a - x}{r} => \theta = \arccos\frac{x_a - x}{r}$$

- top-left

  The destination c in Figure 4.5(b) positions at:

  $$(x - r * \cos\beta,\ y - r * \sin\beta)\ where\ \beta = \theta - \alpha \text{ and } \theta = \arccos\frac{x - x_a}{r}.$$

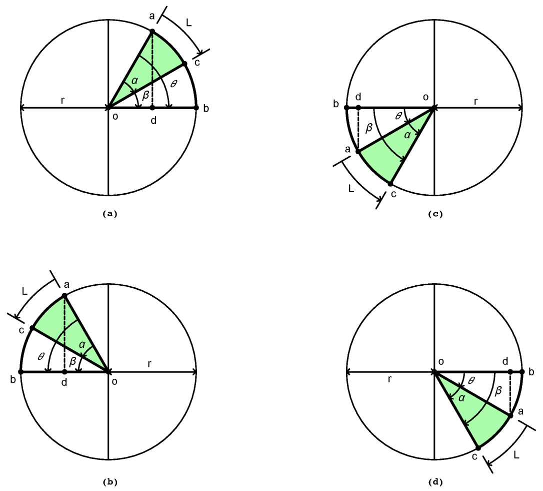


Figure 4.5: A sub-path $(a,\ c)$ with length L on the edge $(a,\ b)$ is part of the target path

- bottom-left

  The destination c in Figure 4.5(c) positions at:

  $$(x - r * \cos\beta,\ y + r * \sin\beta)\ \mathit{where}\ \beta = \theta + \alpha \text{ and } \theta = \arccos\frac{x - x_a}{r}.$$

- bottom-right

  The destination c in Figure 4.5(d) positions at:

  $$(x + r * \cos\beta,\ y + r * \sin\beta)\ \mathit{where}\ \beta = \theta + \alpha \text{ and } \theta = \arccos\frac{x_a - x}{r}.$$

As for α, since the radian of a central angle is defined as the ratio of the length of the subtended arc L divided by the radius of the circle r:

$$\alpha == \frac{L}{r} * \frac{180^\circ}{\pi}$$

Staring point below terminal point

Even if the starting point sits below the terminal point, the formula to calculate the destination remains the same, but with different β and θ.

β refers to the angle between oc and ob, while θ defines the angle between oa and ob. When the starting point a switches the position with the terminal point c, angle β would also swap with angle θ. Consequently, if β is initially bigger than θ, after the reversion, β would be smaller than θ. Therefore,

$$\beta = \begin{cases} \theta + \alpha, & \text{if arc}(a,\ c) \text{ has angle } \ \beta = \theta - \alpha \\ \theta - \alpha, & \text{if arc}(a,\ c) \text{ has angle } \ \beta = \theta + \alpha \end{cases}$$

4.6 Animate transportation operation

4.6.1 Regulate the path direction

This simulation tool is independent of any high-level protocols. In the first layer, the selection of the flow paths ignores the physical properties of fluids, and some paths even begin with outlets. For this reason, the flow path would be modified to align with the entered transportation flow routes.

When the protocol requests for 15% of the segment $(2,\ 1)$ but only with path $(1,\ 2)$ available, the JavaScript program will start searching reversely. After obtaining the target path $(2,\ b)$, a key symbolising the direction from landmark 2 to landmark 2 and a mapped value consisting of a series of edge $(b,\ a)$ and edge $(a,\ 1)$ would replace the initial element representing path $(1,\ 2)$ in the flow map(Figure 4.2). In this case, not only the sequence of edges in the path but also the orientation of the directed each edge is reversed. Hence, the next path could immediately begin at the first remaining edge towards the matching direction.

Straight edge could be converted in the opposite direction by swapping two endpoints, while the reversion of curves also engenders a contrary sweeping-flag and different angles.

4.6.2 Animate fluids and gases in the channels

This program appends a new <path> element with a mapped control path to animate the instant diffusion process of gases. In terms of the flow path, I utilize the style property of HTML and asynchronization function to simulate the motion of liquids.

5 Experiment results

Image

This simulation device will immediately show the design of the bio-chip with the index after it receives the script of the analyzed design.

In the following, I will exhibit the images of the dynamically generated designs in three different scale (Figure 5.1 5.2 5.3) from Columba 2.0[15]. Table 5.1 explains their input features and results after parsing and analyzing from the original design in AutoCAD in the first layer, respectively. From the result, we can find out that the portion of the analyze time grows with the number of edges to be analyzed.

- The first design is the most basic design of the biochip, which contains a mixer, a chamber, and a switch.
- The second design consists of two mixers, two chambers, and three switches.
- The third design has five mixers, whose ring-shaped channel is symmetrical with the horizontal axis, four chambers, and two complex switches.

Table 5.1: Input features and analyze results in the first layer.

Id	|L|	|V|	|B|	|F|	|C|	|E|	|Analyze time|	|Total time|
1	24	16	4	4	13	24	0.005381	0.017562s
2	58	38	13	7	15	61	0.028899s	0.094293s
3	116	74	25	17	26	176	0.101829s	0.34692s

|L|: the number of landmarks in the design; |V|: the number o valves in the design; |B|: the intersection points/branches in the design; |F|: the number of flow inlets/outlets in the design; |V| = |B| + |F|; |C|: the number of control inlets in the design; |E|: the number of edges bounded by two landmarks extracted from the first layer; |Analyze time|: the program time to analyze the parsed design of biochip in C++; |Total time|: the program time to parse and analyze the original design of biochip in AutoCAD.

In these images, landmarks, including valves, intersection points, and flow inlets/outlets are in the same numbering group colored in blue, while control inlets are labeled separately in green. Besides, I implement the stroke of a segment to reflect the

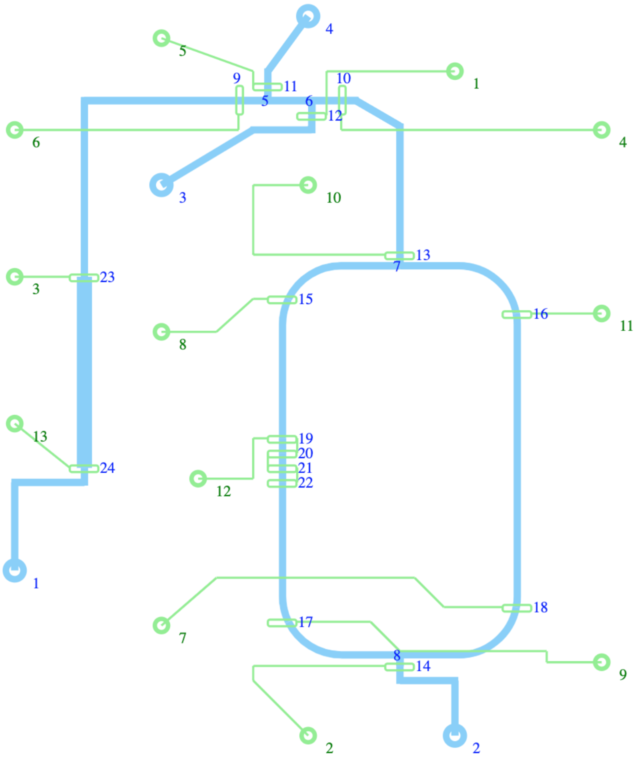


Figure 5.1: Image with Id 1.

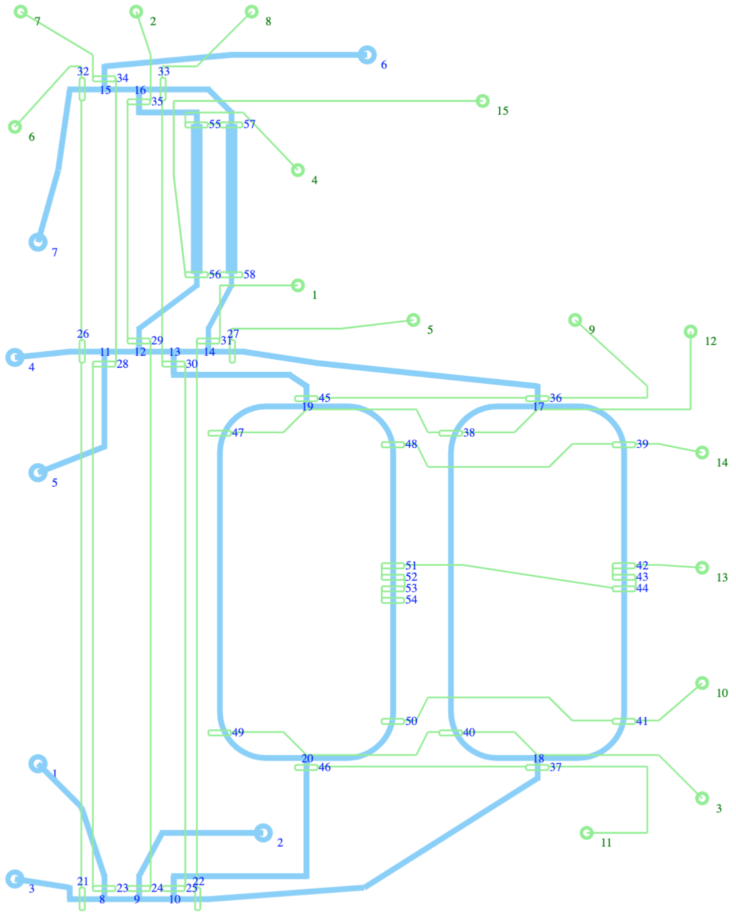


Figure 5.2: Image with Id 2.

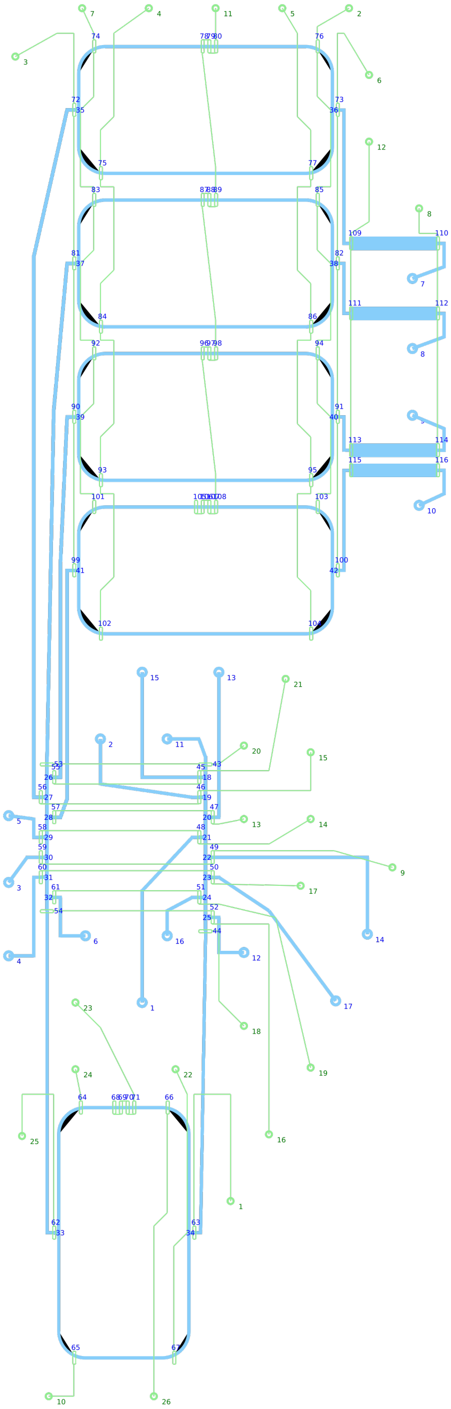


Figure 5.3: Image with Id 3.

width of the channel. The flow channel is thus thicker than the control channel, and the chamber is the thickest segment among the flow segments. The channels are painted in light green and light blue, which indicates the initial status.

Animation

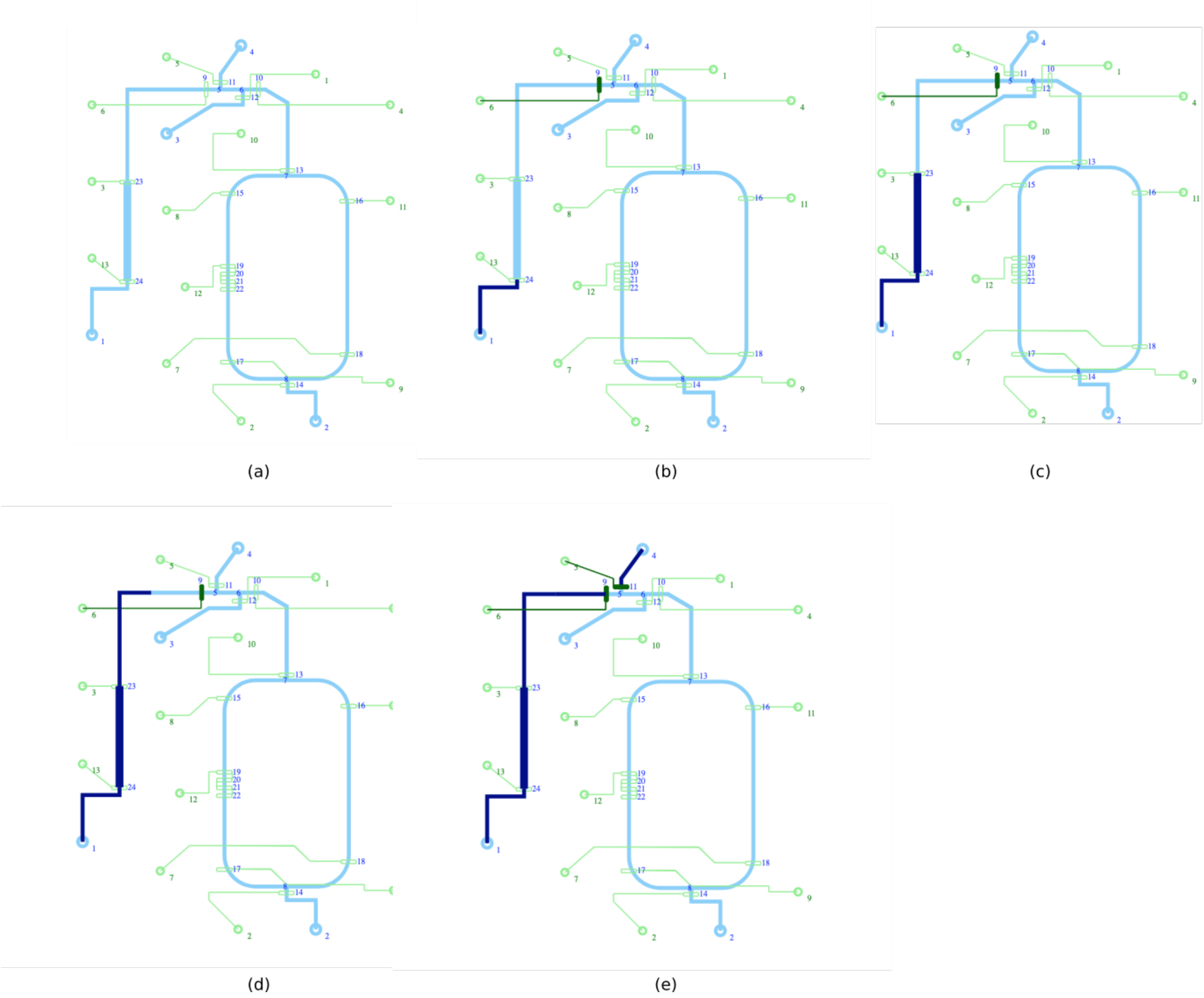
In the animation phase, after the transportation operation is entered, the channel will be then colored with blue and green to imply the fluid movement and the introduction of gases.

Each transportation operation specifies the status in the specified time. As a command, it is structured following specific rules. A command contains three parts: time in seconds, a set of labels of pressurized control inlets, and a set of flow transportation paths, each of which is defined with the label of the starting point and the terminal point, and a specified decimal denoting the percentage. Every two parts are separated with a comma and a space. Every two elements in the same part are separated with a comma.

Figure 5.2 introduces a sequence of the commands to animate the transportation in the first seven seconds with the image in different phases displayed in Figure 5.4. In Table 5.2, (a) is the initial state of the biochip, and the first command starts with (b), which means that in the 2. second, the fluid from flow inlet numbered with 1 could arrive at the landmark 24. As the control channel is immediately filled with gases, the coloring process of the control channel and all valves along connected with the control inlet 6 happens right after the initial state. In (e), the fluids from two different resources start the movement in the sixth second and then reach the destination in the seventh second.

(a)	// initial state
(b)	2; 6; 1 24 1
(c)	4; ; 24 23 1
(d)	6; ; 23 9 0.7
(e)	7; 5; 23 9 0.3, 4 11 1

Table 5.2: The transportation path entered as the input.



(a)

(b)

(c)

(d)

(e)

Figure 5.4: The simulation of the transportation operation.

List of Figures

List of Tables

Bibliography

[1] William SN Trimmer. “Microrobots and micromechanical systems.” In: *Sensors and actuators* 19.3 (1989), pp. 267–287.

[2] Saurabh Vyawahare, Andrew D Griffiths, and Christoph A Merten. “Miniaturization and parallelization of biological and chemical assays in microfluidic devices.” In: *Chemistry & biology* 17.10 (2010), pp. 1052–1065.

[3] Petra S Dittrich, Kaoru Tachikawa, and Andreas Manz. “Micro total analysis systems. Latest advancements and trends.” In: *Analytical chemistry* 78.12 (2006), pp. 3887–3908.

[4] Petra S Dittrich and Andreas Manz. “Lab-on-a-chip: microfluidics in drug discovery.” In: *Nature Reviews Drug Discovery* 5.3 (2006), p. 210.

[5] Todd M Squires and Stephen R Quake. “Microfluidics: Fluid physics at the nanoliter scale.” In: *Reviews of modern physics* 77.3 (2005), p. 977.

[6] George M Whitesides. “The origins and the future of microfluidics.” In: *Nature* 442.7101 (2006), p. 368.

[7] Marc A Unger et al. “Monolithic microfabricated valves and pumps by multilayer soft lithography.” In: *Science* 288.5463 (2000), pp. 113–116.

[8] Jessica Melin and Stephen R Quake. “Microfluidic large-scale integration: the evolution of design rules for biological automation.” In: *Annu. Rev. Biophys. Biomol. Struct.* 36 (2007), pp. 213–231.

[9] Kwang W Oh and Chong H Ahn. “A review of microvalves.” In: *Journal of micromechanics and microengineering* 16.5 (2006), R13.

[10] Tsun-Ming Tseng et al. “Storage and caching: Synthesis of flow-based microfluidic biochips.” In: *IEEE Design & Test* 32.6 (2015), pp. 69–75.

[11] Tsun-Ming Tseng et al. “Columba: Co-layout synthesis for continuous-flow microfluidic biochips.” In: *Proceedings of the 53rd Annual Design Automation Conference*. ACM. 2016, p. 147.

[12] Mengchu Li et al. “Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations.” In: *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2016, pp. 624–629.

[13] Mengchu Li et al. "Component-oriented high-level synthesis for continuous-flow microfluidics considering hybrid-scheduling." In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2017, pp. 1–6.

[14] Mengchu Li et al. "VOM: Flow-Path Validation and Control-Sequence Optimization for Multilayered Continuous-Flow Microfluidic Biochips." In: ().

[15] Tsun-Ming Tseng et al. "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.8 (2017), pp. 1588–1601.

[16] Tsun-Ming Tseng et al. "Cloud Columba: Accessible Design Automation Platform for Production and Inspiration." In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2016).

[17] Stanford Foundry. *Basic Design Rules*. URL: `http://web.stanford.edu/group/foundry`.

[18] IEEE Xplore Digital Library. *IEEE Xplore*. URL: `http://ieeexplore.ieee.org`.

[19] Hailong Yao, Tsung-Yi Ho, and Yici Cai. "PACOR: practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips." In: *Proceedings of the 52nd Annual Design Automation Conference*. ACM. 2015, p. 142.

[20] Kai Hu et al. "Control-layer routing and control-pin minimization for flow-based microfluidic biochips." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.1 (2016), pp. 55–68.