DEPARTMENT OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Switch Design for Microfluidic Large-Scale Integration

Yanlu Ma

DEPARTMENT OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Switch Design for Microfluidic Large-Scale Integration

Author:	Yanlu Ma
Supervisor:	Prof. Dr. -Ing. Ulf Schlichtmann
Advisor:	Dr. -Ing. Tsun-Ming Tseng
Submission Date:	10.8.2020

Master's thesis statement of originality

Herewith I confirm that I have done my master's thesis by myself, without contributions from any sources other than those cited in the text and bibliography.

Munich, 10.8.2020

Place and date

Signature

Abstract

In the past few decades, continuous-flow microfluidic large-scale integration (mLSI) is more and more important in biological/chemical fields. This is driven largely by the need for greater, more accurate and faster access to information on the composition of substances in the development of environmental and materials science.

However, the prior research underestimates the microfluidic layer interactions in the design. With that in mind, a co-layout synthesis tool for continuous-flow microfluidic biochips, which is called Columba, is proposed by Dr. Tsun-Ming Tseng. This tool has several functional modules such as mixers, reaction chambers and switches. By combining these elements it can form different physical structures.

Switches guide fluid direction when flow channels cross. In some cases, two different microfluids need to avoid contamination. But the tool does not support this feature. This work proposes to design a new module model for switch, in order to adapt to different circumstances. Proposed method is based on a linear programming model that optimizes physical structure

of switch. Experimental results represent that the proposed method can efficiently avoid contamination.

Acknowledgements

I would like to thank the following individuals, without whom I would not have been able to complete my thesis, and without whom I would not have made it through my Master's degree!

Firstly, I would like to thank my supervisor, Dr. Tsun-Ming Tseng, for his support throughout this project. I am extremely grateful for our friendly chats during our meetings.

Secondly, I must thank Prof. Ulf Schlichtmann for giving me the opportunity to finish my thesis at Chair of Electronic Design Automation. I would also like to thank Alexandre Carvalho Truppel. His work on WRONoCs inspired me a lot.

Finally, I would like to thank my boyfriend, Teng Wang. And my biggest thank to my family for all the support. I love you all.

Yanlu Ma

Contents

List of Figures

List of Tables

Abbreviations

CPU	Central Processing Unit
EDA	Electronic Design Automation
IC	Integrated Circuit
GRU	General Routing Unit
LP	Linear Programming
mLSI	microfluidic Large-Scale Integration
MIP	Mixed Integer Programming

1 Introduction

In this chapter, we will have an overview of flow-based microfluidic biochips. First, physical architecture of microfluidic biochips is described. Second, the structure of this thesis is given.

1.1 Physical architecture of microfluidic biochips

In the past decades, flow-based microfluidic biochips [1] are rapidly developed for biological experiments. Soft lithography is used to produce two-layer continuous-flow microfluidic biochips [2]. The structure of flow-based microfluidic biochips is shown in Fig. 1.1 [1].

Control layer. This layer is used to activate/deactivate the flow paths in accordance with requirements.

Flow layer. Under/Above the control layer is the flow layer, which is constructed for transportation of samples and reagents.

Channel. Each layer contains several channels, which allow fluids to flow through.

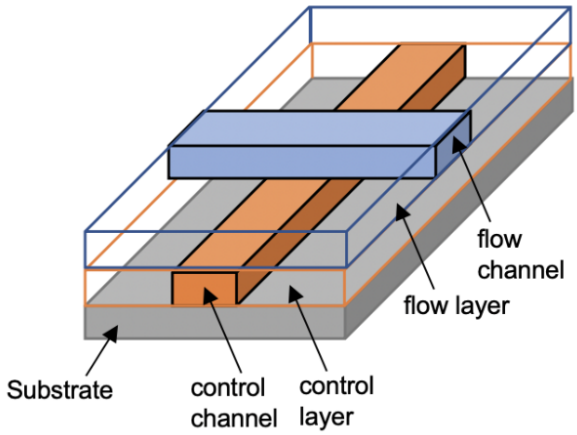

Figure 1.1: Structure of flow-based biochips [1]

Valve. Continuous-flow microfluidic biochips use valves to control the movement of samples and reagents [1]. When fluid fills the control channel, membrane will be pressed upwards. Thus, the flow channel is blocked. The structure of valve is shown in Fig. 1.2 [2].

1.2 Thesis structure

This thesis starts with a short review of background in the second chapter, mentioning why Columba necessary and how Columba works. It also compares the difference between Columba 2.0 and Columba S.

Chapter 3 discusses the general concept of this thesis. A formal definition of new design is given and the optimization problem is studied. Among

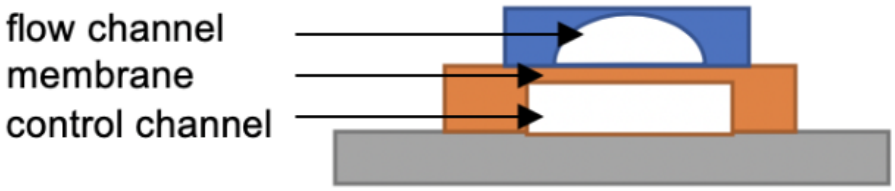

Figure 1.2: Structure of valve [2]

others, chapter 3 shows the reason of choosing linear programming based optimization algorithm.

Chapter 4 implements the details of an optimization algorithm.

Chapter 5 shows the results and compares this new method with previous work, depending on different number of in-/outlets and requirements to avoid contamination.

Finally, Chapter 6 draws a conclusion for this thesis and proposes space for future development.

2 Background

Having given a review of flow-based microfluidic biochips, this chapter gives the introduction on a co-layout synthesis tool - Columba [3]. It also gives the starting point for this thesis, since this research focuses on the switch design, which is an important module model of Columba.

2.1 Columba

It is due to error rates in manual design that an automation design for continuous-flow microfluidic large-scale integration (mLSI) increases significantly [3]. Compared to previous design automation, Columba takes interaction between different layers into consideration. As shown in Fig. 2.1 [4], the design of a nucleic acid processor is generated by Columba.

From the graph we can see that this design has one control layer (green lines) and one flow layer (red lines). Valves in width of control layer are intersection of two lines. Columba design is based on different functional microfluidic components. These components provide the physical struc-

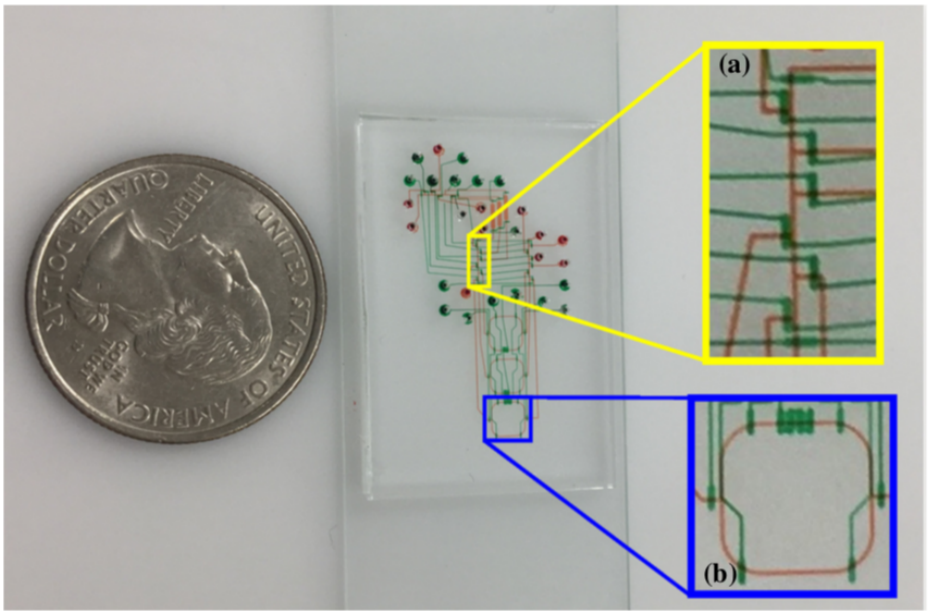

Figure 2.1: Design of a nucleic acid processor by Columba [3] (a) Switch (b) Mixer

tures of microfluidics. For instance, mixers (a) are used for mixing operation. Switches (b) are used to control the direction of fluid flow in intersecting flow channels. In addition, liquid is added von inlets and it flows out of the outlets. The design rules of those components are described in plain text. Then Columba takes netlist descriptions from text as inputs. Finally, an AutoCAD-compatible design is generated by Columba. As shown in Fig. 2.2, this design can be directly used for mask fabrication [3].

2.2 Columba 2.0 vs Columba S

Columba has two different optimized versions, Columba 2.0 and Columba S. Both of them are optimized for lesser run time and also for fewer number of inlets/outlets. In particular, Columba S is optimized for shorter length of flow channel.

Here we only focus on the switch structure. Different physical structures are illustrated in Fig. 2.3 [3] and in Fig. 2.4 [5]. Control channels are green lines. Flow channels are blue lines. Valves are indicated with orange lines. The general proposed method to actuate valves is pin selection. Table 2.1 [3] shows pin options in Columba 2.0 to actuate valves.

Switch in Columba 2.0 has one main flow channel and several flow channel flow channel junctions. The distance between adjacent flow channel junctions is constant. Only one direction of the paired flow channel junctions can be used as flow channel, the other one as well as the corre-

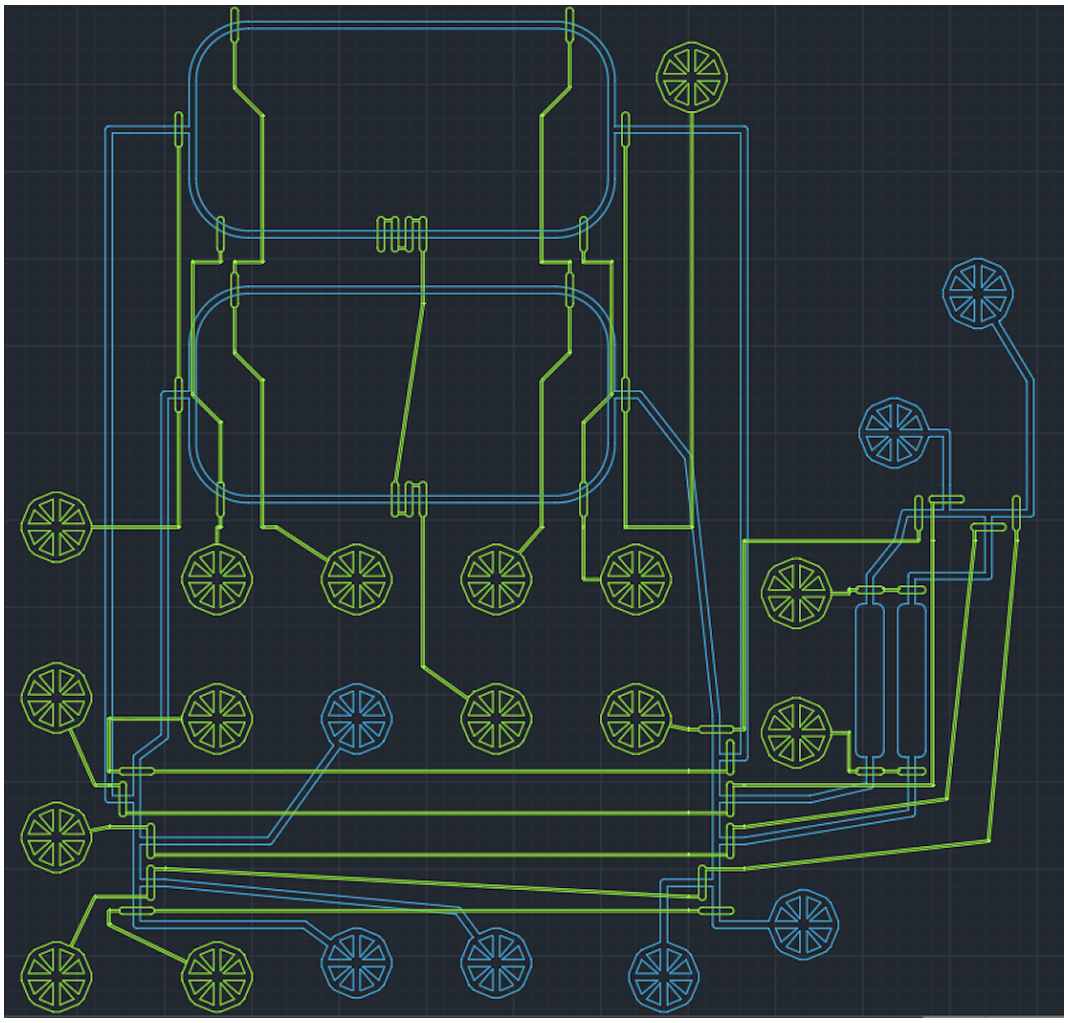

Figure 2.2: mLSI Design implemented in AutoCAD [3]

sponding valve and control channel will be removed. In other words, once Columba chooses one flow channel junction connected with pin *p2'* as flow path, then the opposite flow channel junction connected with pin $p2''$ and valve $\overline{v2}$ are automatically deleted. Fig. 2.3 shows the principle modul (the upper structure) and one possible binding option (the structure below).

Switch in Columba S consists of one flow channel spine and several flow channel junctions. Due to the characteristics of spine, the distance between adjacent flow channel junctions can be changed to suit different needs. Therefore flow channels can access from different locations, in order to minimize the total channel length. Compared to switch structure in Columba 2.0, only one side of pins can be reserved. As a result, valves can be actuated either from the top or from the bottom [5].

Table 2.1: Pin selection for switch

v0	$\{p0\}, \{p1\}, \{p0, p1\}$
v1	$\{p2\}, \{p3\}, \{p2, p3\}$
v2	$\{p4\}, \{p5\}, \{p6\}, \{p7\}, \{p4, p5\}, \{p4, p7\}, \{p5, p6\}, \{p6, p7\}$
$\overline{v2}$	$\{p4\}, \{p5\}, \{p6\}, \{p7\}, \{p4, p5\}, \{p4, p7\}, \{p5, p6\}, \{p6, p7\}$
$\cdots$	(analogous to *v2* and $\overline{v2}$)

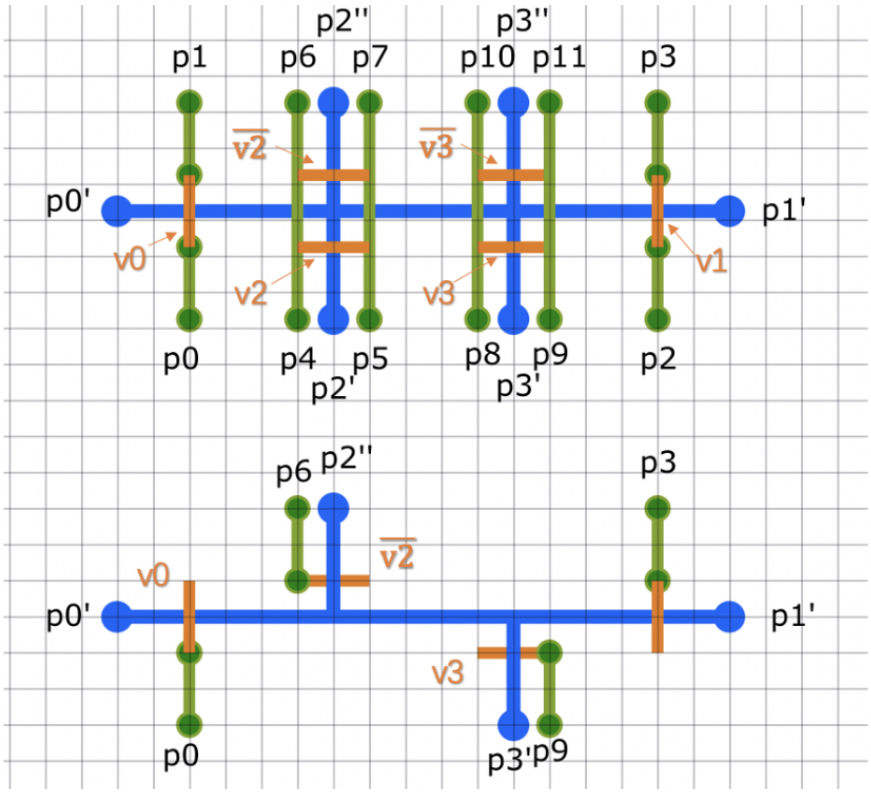

Figure 2.3: Switch structure in Columba 2.0 [3]

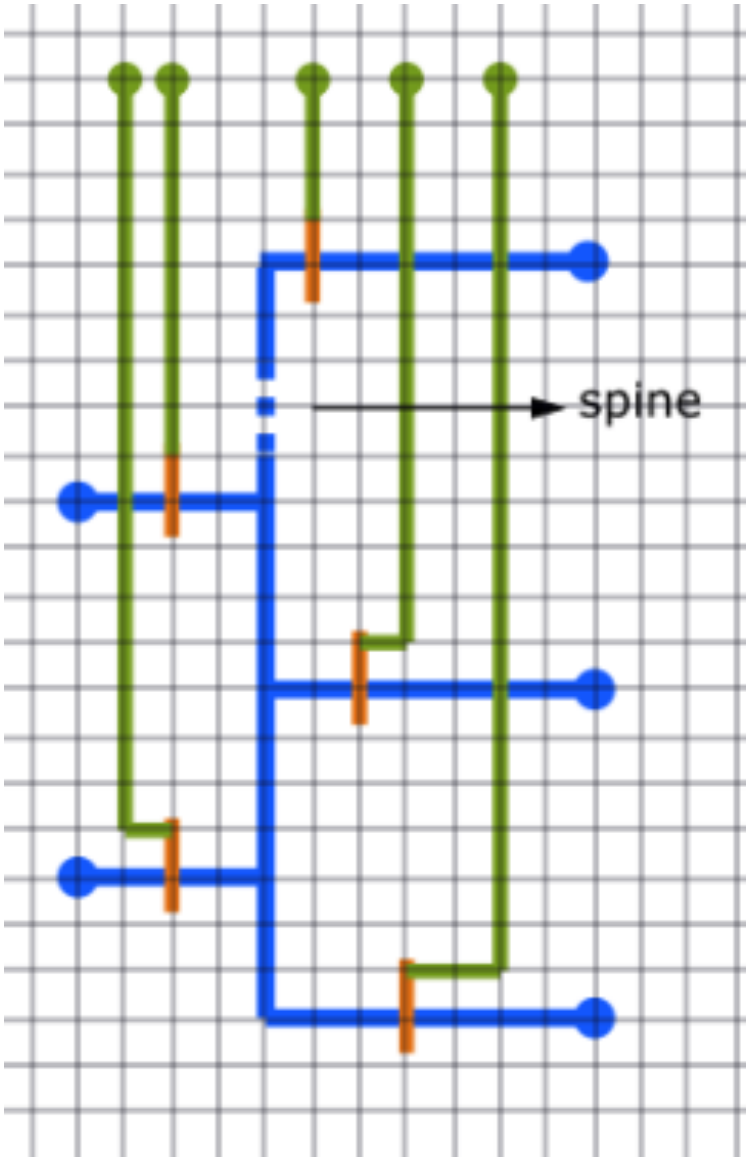

Figure 2.4: Switch structure in Columba S [5]

2.3 Switch design problem

As the need for biological experimentation grows, which enables the design of a platform for large and complex applications. Different samples and reagents might flow through same switch structure. Therefore, cross-contamination of reagents may occur in any of the following situations:

1. The reagent contains the substrate to be measured in the next test.
2. The current reactant has an effect on the latter reaction.

These might result in the contamination. However, these two versions of Columba do not provide much attention to avoidance of contamination, since current switch consists of one main flow channel. All fluids flowing through same switch share this path, which may causes contamination. A major goal of this thesis is to design a new switch structure with follows:

Contamination Avoidence Discussed before.

Structural Diversity Achieved through pin assignment and physical model.

3 Design Concept

In this chapter, the concept and design flow of switch design will be discussed. First, an approach to the physical construction of switch is presented with examples from published research. Then, the proposed method is described presenting the reason for choosing Mixed Integer Programming (MIP) and for choosing corresponding software. Finally, the necessary steps of software setup are listed.

3.1 Design flow

The design of switch starts with the following prerequisites:

• The goal of this thesis is the switch design based on Columba. All design of functional components in Columba relies on pin assignment. For the sake of consistency, pin assignment is also considered in this design. Therefore, we have to find new physical topology first.

With the prerequisites, the design concept of switch includes these three aspects:

- Design of physical model for switch.
- Possible optimization algorithm.
- Software setup.

3.2 Physical Model

The basic topology of this model (Fig. 3.3) is inspired by published research. Tan Yan and Martin. F. Wong proposed a network flow model in Fig. 3.1 [6]. They focused on the traditional escape routing problem, due to limitation of the wire-number between two orthogonal or diagonal adjacent pin. Then, they built an algorithm based on this model. Fig. 3.2 [6] shows the details inside a tile. Four pins on the boundary can be used as flow entrances. Five nodes (N, W, S, E, C) form an inner-structure. Compared to previous switch model, this structure has an advantage in the structure diversity. The main flow channel can be replaced by this inner-structure.

I extend this model with more pins on the boundary. It includes TL (Top left), T (Top), TR (Top right), L (Left), R (Right), BL (Bottom left), B (Bottom), BR (Bottom right). Those pins can be used as inlets/outlets of fluids. Edges between pins and nodes are recognized as flow paths. Fig. 3.3 shows the pins, nodes and connections between them. For example, T - N - C - S - BR indicates one possible path of fluid. Furthermore, this model is symmetric. I call it General Routing Unit (GRU) for the physical topology. The relevant essay was published by Alexandre Truppel [7].

Similar to flow model in Fig. 3.1, I propose a network consisting two GRUs. Fig. 3.4 shows the combination of them. It contains twelve pins and eight nodes adapting many situations. The combination of three or more GRUs is not considered in this thesis.

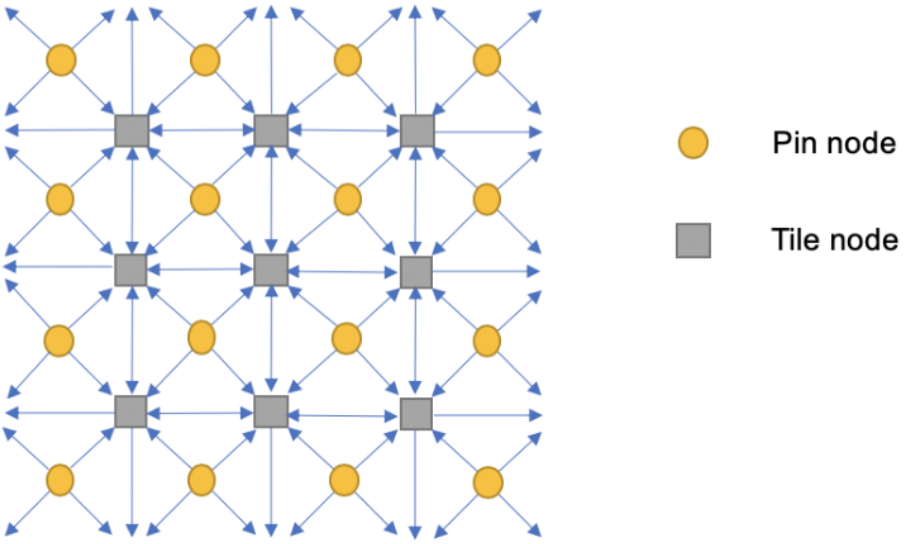

Figure 3.1: The traditional network flow model [6]

3.3 Mathematical model

In order to build a mathematical model for switch, Mixed Integer Programming (MIP) is an advanced method. All pins and connections in the physical model of switch can be described as variables and linear con-

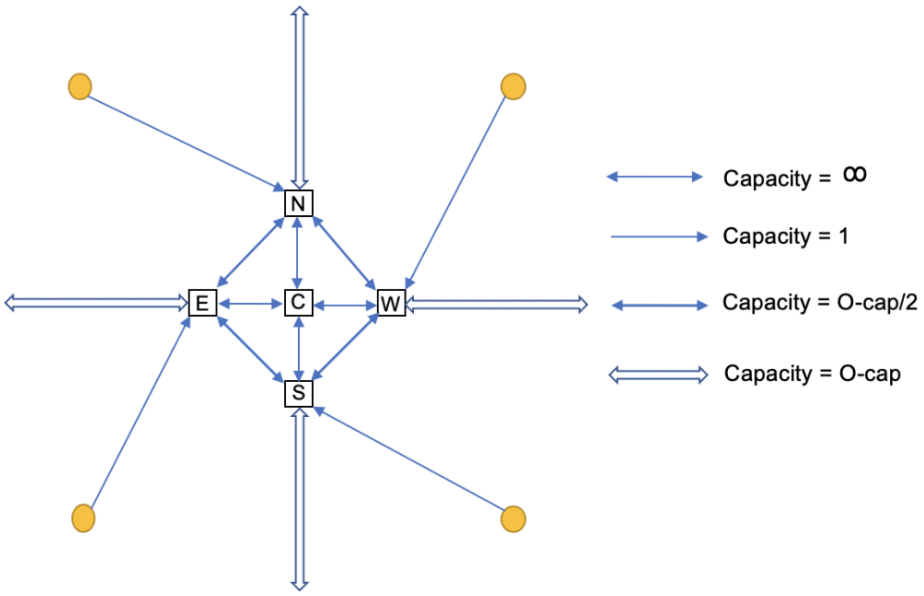

Figure 3.2: Network flow model inside a tile [6]

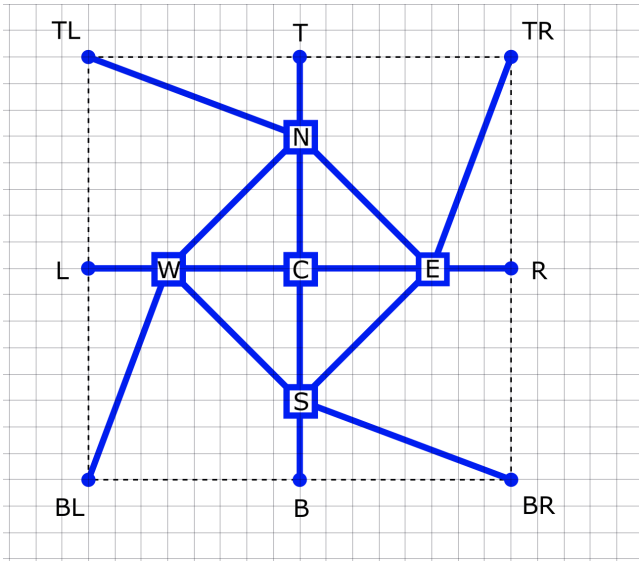

Figure 3.3: Flow model of one GRU

straints. The optimization problem is simplified into linear optimization functions. Besides, a mixed integer linear program (MIP) is of the form:

$$\begin{aligned} \text{min} \quad & c^T \cdot x \\ \text{subject to} \quad & A \cdot x = b \\ & x_i \in Z \; \forall i \in I \end{aligned}$$

Gurobi is one of the advanced MIP solvers. Mathematical problems that Gurobi can solve: linear problems and mixed integer linear and quadratic problems.

Gurobi has several technical advantages:

1. Multi-objective optimization.

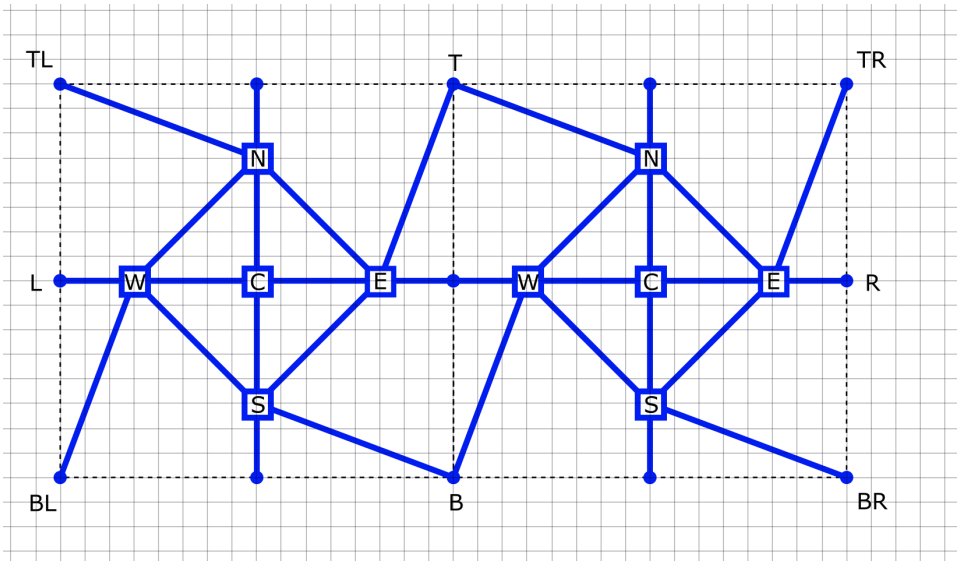

Figure 3.4: Flow model of two GRUs

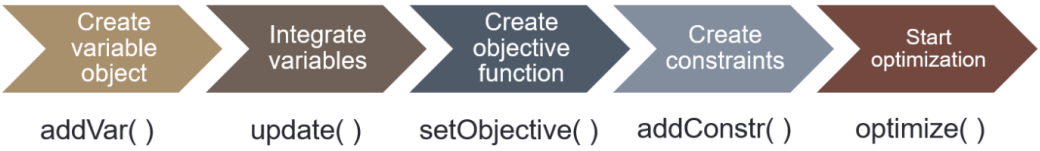

Figure 3.5: Optimization Process of Gurobi

2. It uses the latest optimization to take the full advantage of multi-core processors.

3. The problem scale is limited only by the amount of computer memory, not by the number of variables or the number of constraints.

4. It is available on multiple platforms, including Windows, Linux, Mac OS X.

5. A convenient and lightweight interface is provided which supports C++, Java, Python, Matlab and R.

6. Low memory consumption

3.3.1 Algorithm

Transport path varies with different pin assignments. The contamination-avoidance pin assignment problem is essentially an algorithmic design problem. In practical modeling, we have to consider path conflict and settings of valves. The pin-assignment problem is a difficult NP problem that is difficult to solve quickly at large volumes. We have 8 pins of inlets/outlets for one GRU. But using MIP directly is too restrictive, and many conditions are not well represented. If we go through all the solutions at once to find the optimal solution, the program will not be able to finish, and if we the use greedy algorithm, we may not be able to get the optimal solution.

Where the fluid enters and exits is uncertain. Then I convert this uncertainty to path allocation, which means a liquid can choose one of the

established pathway options. Let's start with a list of all possible paths and move on to path allocation. Of course, as long as all points through which the path passes are determined, the corresponding length of path can also be calculated. Chapter 4 implements the details.

3.3.2 Design Workflow

In this subsection, inputs and outputs of this model are specified.

Input:	Communication rules.
Step:	Pin assignment and path allocation.
Output:	• Path of each liquid.
	• An image file to represent switch model.
Minimization objective:	Total length of path

Communication rules define a set of inlets I and a set of outlets O, start points and end points of fluids (Set IO), conflict of fluids (Set AC, indicating which liquids need to avoid contaminating each other). Randomly generated inputs should conform to the following:

1: One-to-one principle for set I and set O. Each inlet $i \in \mathbb{I}$ and each outlet $o \in \mathbb{O}$ can only correspond to one pin. The total number of used pins equals to the sum of number of inlets and outlets.

$$|I| + |O| \leq \text{Number of pins (8 for one GRU, 12 for two GRUs)}$$

2: $|I| \geq |O|$. This is why we need a switch.

3: All inlets $i \in I$ have corresponding outlets $o \in O$.

4: Elements in set *IO* are in pairs of inlet and outlet. For example, (i_2, o_1).

5: $|AC|$ can be zero.

6: Conflict between inlets $i_1 \in I$, $i_2 \in I$.

$\rightarrow$ The same inlet cannot conflict with itself. $i_1 \neq i_2$

$\rightarrow$ The corresponding outlet is not shared. Either element (i_1, o_1) or element (i_2, o_1) exists in set *IO*.

7: No conflict between outlets.

8: Conflict between inlet and outlet. $i_1 \in I$, $o_1 \in O$.

$\rightarrow$ Element (i_1, o_1) in set *IO* can not exist.

3.4 Software setup

Figure 3.6: Software platform [8]

Here I choose Anaconda as the software platform for programming. Anaconda is an open source distribution for the programming language Python. As shown in the last section, Python is supported by Gurobi. It'll take a few steps, so that Gurobi can support Python for Mac [9].

1: Add the Gurobi channel to default search list with command

conda config –add channels http://conda.anaconda.org/gurobi

2: Install the Gurobi package with command *conda install gurobi*. Fig. 3.7 shows the results in terminal window that Gurobi packages are installed.

3: Get a Gurobi License.

4: Set an environment variable in terminal by adding the following line to

shell startup file. Open the file in terminal with command *vi /.bashrc*. Then, the gurobi.lic file is placed for platform.

$$\text{export } GRB\_LICENSE\_FILE = /Users/gurobi.lic$$

```
Downloading and Extracting Packages
ca-certificates 2020.1.1: ################################################# | 100%
gurobi 9.0.2: ############################################################# | 100%
certifi 2020.4.5.1: ######################################################### | 100%
openssl 1.0.2u: ############################################################## | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
YanludeMacBook-Pro:~ emma$
```

Figure 3.7: Installation of Gurobi packages

4 Optimization algorithm

After having randomly generated communication rule as input, this chapter presents the details about an optimization algorithm. As discussed in previous chapter, here we focus on the path allocation. I will define some necessary variables and constraints related to them.

4.1 Switch model

Fig. 4.1 and Fig. 4.2 show the design of GRU consisting of flow channels, control channels and valves. Each valve has two different paths to be activated, except for v_2, v_{10}, v_{15}, v_{21} and v_{22} in Fig. 4.2, since there is little space for more channels. If a channel segment is not connected to any selected pin [3], it will be removed in the final design. These two structures are basic designs, which mean the unused flow channels and valves will also be removed. d in Fig. 4.1 represents the minimum distance between two channels.

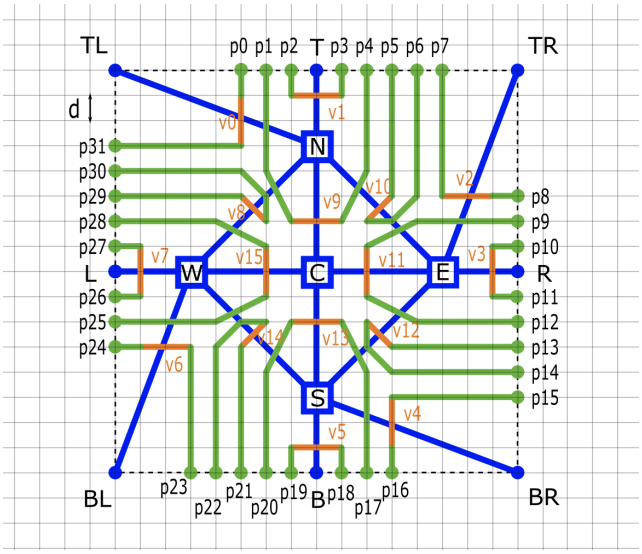

Figure 4.1: Switch model - One GRU

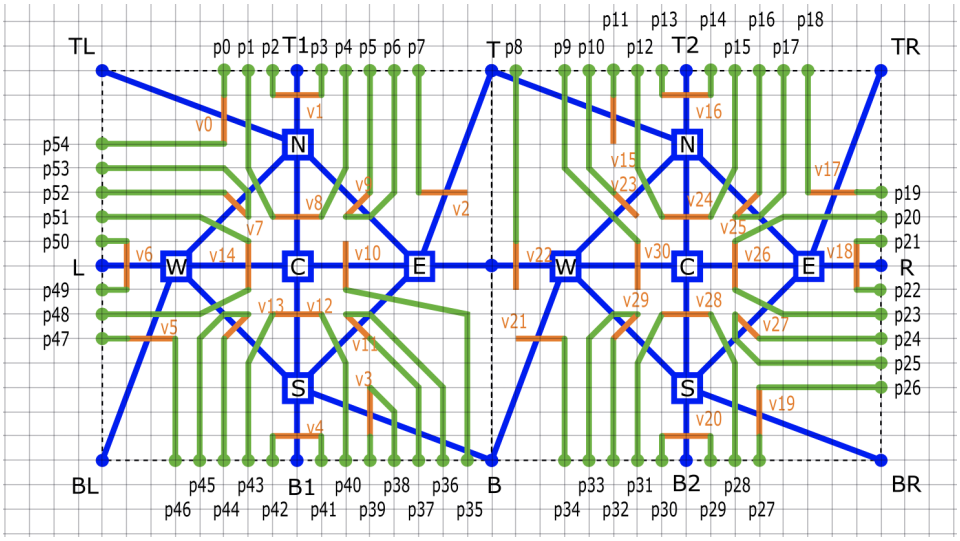

Figure 4.2: Switch model - Two GRUs

4.1.1 Constants and indices

The general indices are points $i \in \mathbb{P} \cup \mathbb{Q}$. The constants are as followed as below:

The total number of inlets N_{inlet} and outlets N_{outlet} are given as input, which also defines the number of GRUs N_{gru}.

L_{fh} represents the length of flow channel. l_1 is the length between T and N, l_2 for N and C, l_3 for N and W, l_4 for N and TL.

Table 4.1: Constants and Indices

N_{inlet}	The total number of inlets
N_{outlet}	The total number of outlets
N_{gru}	The total number of GRUs
$L_{fh} \in \{l_1, l_2, l_3, l_4\}, \ l_1 \leq l_2 < l_3 \leq l_4$	Length of flow channel fh
$\mathbb{P} = \{TL, T, TR, L, R, BL, BR, B\}$	Pins of GRU
$\mathbb{Q} = \{N, W, S, E, C\}$	Nodes of GRU
$fh_{\mathbb{Q}}^{P}$	Channel connected to pins
$fh_{\mathbb{Q}}^{Q'}$	Channel between nodes

4.1.2 Variables and constraints

The variables and constraints are as follows:

Table 4.2: Variables

Binary variable	
$ffh_{f,\,fh,\,g}$	Fluid f goes through flow channel fh in GRU g.
$fac_{f1,\,f2}$	Fluid $f1$ and $f2$ use same flow channel
$b_{i,\,g}$	point I of GRU g is accessed
$fb_{f,\,i,\,g}$	Fluid f contains point of GRU g in its path
$v \in V$	Valve : 0 - not activated ; 1 - activated
Continuous variable	
$l_{f,\,g}$	Total length of all used flow channels

We introduce an binary variable $b_{i,\,g}$, $\forall i \in \mathbb{P} \cup \mathbb{Q}$, $g = 1...\,N_{gru}$ to indicate whether point I of GRU g is accessed. $b_{p,\,g} = 1$, $\forall p \in \mathbb{P}$ means pin p of GRU g is used as an in-/outlet.

1. Each pin of GRUs is connected to exactly one inlet/outlet. One fluid direction consists of one pair of inlet/outlet. Thus, the sum of all variables $b_{p,\,g}$ is:

$$\sum_{p \in \mathbb{P}} \sum_{g = 1 \ldots N_{gru}} b_{p,g} = N_{inlet} + N_{outlet} \tag{1}$$

Which can be also written as

$$b_{TL,g} + b_{T,g} + b_{TR,g} + b_{R,g} + b_{BR,g} + b_{B,g} + b_{BL,g} + b_{L,g} = N_{inlet} + N_{outlet} \tag{1.1}$$

$$\forall g = 1 \ldots N_{gru}$$

2. Once one pin is chosen as an in-/outlet, the corresponding node is also determined. This is the uniqueness of the GRU structure.

$$b_{N,g} = b_{T,g} \cup b_{TL,g} \tag{2.1}$$

$$b_{W,g} = b_{L,g} \cup b_{BL,g} \tag{2.2}$$

$$b_{S,g} = b_{B,g} \cup b_{BR,g} \tag{2.3}$$

$$b_{E,g} = b_{R,g} \cup b_{TR,g} \tag{2.4}$$

To describe the path of each fluid the set of binary variables $ffh_{f,\,fh,\,g}$, $\forall f = 1... N_f$, $fh = 1... N_{fh}$, $g = 1... N_{gru}$ is created, where $ffh_{f,\,fh,\,g} = 1$ means fluid f goes through channel fh in GRU g.

3. If we try to avoid the contamination of 2 different fluids, following constraints must be created:

$$ffh_{f1,\,fh,\,g} + ffh_{f2,\,fh,\,g} \leq 1 \tag{3}$$

4. The path must be continuous from the inlet to the responding outlet. It is taken into consideration when we list all possible paths. At least one node is included in one path.

4.2 Path allocation

Fig. 4.3 describes the details of input. We define a few symbols that are common throughout next text:

```
Communication Rule:
Inlet: [1, 2, 3, 4, 5]
Outlet: [6, 7, 8]
In-Out: [[1, [8]], [2, [6]], [3, [8]], [4, [6]], [5, [7, 6]]]
AC: [[1, [2, 6]], [3, [4, 7]]]
```

Figure 4.3: Example of Input

• *Inlet, Outlet*: Use different arabic numbers to represent inlets and outlets.

• IO $(In - Out)$: Combination of in-/outlets is also added into the input. Inlet corresponds to start point and outlet corresponds to end point.

• AC: The liquid flowing through different inlets and outlets is at risk of contamination.

• $i \in \mathbb{I}$: The total number of combination IO

α_i represents start point of each IO.

β_i represents end point of each IO.

• $d \in \mathbb{D}$

D = [0, 1, 2, 3, 4, ..., 407]. Full paths with index 0 - 407 (Case "one GRU")

D = [0, 1, 2, 3, 4, ..., 15571]. Full paths with index 0 - 15571 (Case "two GRUs")

Based on physical templates, we can list all possible paths with calculated length. Fig. 4.4 - Fig. 4.6 show the start points and end points of all paths. The first column is the index of all paths for one GRU. It has 56 combinations of start and end points. The second column shows the total number of paths with same start and end points. Rest columns indicate whether path d includes pin p and node n. Fig. 4.7 - Fig. 4.9 show the total points (pins p and nodes n) that paths go through in case of two GRUs. Due to the large volume of paths in this case only a few paths are shown here.

The above paths are split into an encoding matrix that contains the location or not. The advantage of this processing is that the constraints can be read directly from the matrix, which is convenient for programming. In addition, it is convenient to count the total length of all selected paths. We

	All_Paths	Path	TL	T	TR	BR	B	BL	R	L
0	1	(TL, T)	1	1	0	0	0	0	0	0
1	8	(TL, TR)	1	0	1	0	0	0	0	0
2	8	(TL, R)	1	0	0	0	0	0	1	0
3	9	(TL, BR)	1	0	0	1	0	0	0	0
4	9	(TL, B)	1	0	0	0	1	0	0	0
5	8	(TL, BL)	1	0	0	0	0	1	0	0
6	8	(TL, L)	1	0	0	0	0	0	0	1
7	1	(T, TL)	1	1	0	0	0	0	0	0
8	8	(T, TR)	0	1	1	0	0	0	0	0
9	8	(T, R)	0	1	0	0	0	0	1	0
10	9	(T, BR)	0	1	0	1	0	0	0	0
11	9	(T, B)	0	1	0	0	1	0	0	0
12	8	(T, BL)	0	1	0	0	0	1	0	0
13	8	(T, L)	0	1	0	0	0	0	0	1
14	8	(TR, TL)	1	0	1	0	0	0	0	0
15	8	(TR, T)	0	1	1	0	0	0	0	0
16	1	(TR, R)	0	0	1	0	0	0	1	0
17	8	(TR, BR)	0	0	1	1	0	0	0	0

Figure 4.4: Path matrix 1 - 1

18	8	(TR, B)	0	0	1	0	1	0	0	0
19	9	(TR, BL)	0	0	1	0	0	1	0	0
20	9	(TR, L)	0	0	1	0	0	0	0	1
21	8	(R, TL)	1	0	0	0	0	0	1	0
22	8	(R, T)	0	1	0	0	0	0	1	0
23	1	(R, TR)	0	0	1	0	0	0	1	0
24	8	(R, BR)	0	0	0	1	0	0	1	0
25	8	(R, B)	0	0	0	0	1	0	1	0
26	9	(R, BL)	0	0	0	0	0	1	1	0
27	9	(R, L)	0	0	0	0	0	0	1	1
28	9	(BR, TL)	1	0	0	1	0	0	0	0
29	9	(BR, T)	0	1	0	1	0	0	0	0
30	8	(BR, TR)	0	0	1	1	0	0	0	0
31	8	(BR, R)	0	0	0	1	0	0	1	0
32	1	(BR, B)	0	0	0	1	1	0	0	0
33	8	(BR, BL)	0	0	0	1	0	1	0	0
34	8	(BR, L)	0	0	0	1	0	0	0	1
35	9	(B, TL)	1	0	0	0	1	0	0	0
36	9	(B, T)	0	1	0	0	1	0	0	0
37	8	(B, TR)	0	0	1	0	1	0	0	0

Figure 4.5: Path matrix 1 - 2

37	8	(B, TR)	0	0	1	0	1	0	0	0
38	8	(B, R)	0	0	0	0	1	0	1	0
39	1	(B, BR)	0	0	0	1	1	0	0	0
40	8	(BR, BL)	0	0	0	1	0	1	0	0
41	8	(BR, L)	0	0	0	1	0	0	0	1
42	8	(BL, TL)	1	0	0	0	0	1	0	0
43	8	(BL, T)	0	1	0	0	0	1	0	0
44	9	(BL, TR)	0	0	1	0	0	1	0	0
45	9	(BL, R)	0	0	0	0	0	1	1	0
46	8	(BL, BR)	0	0	0	1	0	1	0	0
47	8	(BL, B)	0	0	0	0	1	1	0	0
48	1	(BL, L)	0	0	0	0	0	1	0	1
49	8	(L, TL)	1	0	0	0	0	0	0	1
50	8	(L, T)	0	1	0	0	0	0	0	1
51	9	(L, TR)	0	0	1	0	0	0	0	1
52	9	(L, R)	0	0	0	0	0	0	1	1
53	8	(L, BR)	0	0	0	1	0	0	0	1
54	8	(L, B)	0	0	0	0	1	0	0	1
55	1	(L, BL)	0	0	0	0	0	1	0	1

Figure 4.6: Path matrix 1 - 3

	End	Index	Length	Path	Start	TL	TR	BR	BL	R	L
0	T	0	11.544	[TL, N, T]	TL	1	0	0	0	0	0
1	TR	1	34.158	[TL, N, W, C, E, TR]	TL	1	1	0	0	0	0
2	TR	1	41.228	[TL, N, W, C, S, E, TR]	TL	1	1	0	0	0	0
3	TR	1	41.228	[TL, N, W, S, C, E, TR]	TL	1	1	0	0	0	0
4	TR	1	38.298	[TL, N, W, S, E, TR]	TL	1	1	0	0	0	0
5	TR	1	27.088	[TL, N, C, E, TR]	TL	1	1	0	0	0	0
6	TR	1	34.158	[TL, N, C, S, E, TR]	TL	1	1	0	0	0	0
7	TR	1	41.228	[TL, N, C, W, S, E, TR]	TL	1	1	0	0	0	0
8	TR	1	24.158	[TL, N, E, TR]	TL	1	1	0	0	0	0
9	R	2	28.614	[TL, N, W, C, E, R]	TL	1	0	0	0	1	0
10	R	2	35.684	[TL, N, W, C, S, E, R]	TL	1	0	0	0	1	0
11	R	2	35.684	[TL, N, W, S, C, E, R]	TL	1	0	0	0	1	0
12	R	2	32.754	[TL, N, W, S, E, R]	TL	1	0	0	0	1	0
13	R	2	21.544	[TL, N, C, E, R]	TL	1	0	0	0	1	0
14	R	2	28.614	[TL, N, C, S, E, R]	TL	1	0	0	0	1	0
15	R	2	35.684	[TL, N, C, W, S, E, R]	TL	1	0	0	0	1	0
16	R	2	18.614	[TL, N, E, R]	TL	1	0	0	0	1	0

Figure 4.7: Path matrix 2 - 1

188	BL	26	28.614	[R, E, N, C, W, BL]	R	0	0	0	1	1	0
189	BL	26	28.614	[R, E, C, N, W, BL]	R	0	0	0	1	1	0
190	BL	26	28.614	[R, E, C, S, W, BL]	R	0	0	0	1	1	0
191	BL	26	21.544	[R, E, C, W, BL]	R	0	0	0	1	1	0
192	BL	26	25.684	[R, E, S, W, BL]	R	0	0	0	1	1	0
193	BL	26	35.684	[R, E, S, C, N, W, BL]	R	0	0	0	1	1	0
194	BL	26	28.614	[R, E, S, C, W, BL]	R	0	0	0	1	1	0
195	L	27	20.140	[R, E, N, W, L]	R	0	0	0	0	1	1
196	L	27	30.140	[R, E, N, C, S, W, L]	R	0	0	0	0	1	1
197	L	27	23.070	[R, E, N, C, W, L]	R	0	0	0	0	1	1
198	L	27	23.070	[R, E, C, N, W, L]	R	0	0	0	0	1	1
199	L	27	23.070	[R, E, C, S, W, L]	R	0	0	0	0	1	1
200	L	27	16.000	[R, E, C, W, L]	R	0	0	0	0	1	1
201	L	27	20.140	[R, E, S, W, L]	R	0	0	0	0	1	1
202	L	27	30.140	[R, E, S, C, N, W, L]	R	0	0	0	0	1	1
203	L	27	23.070	[R, E, S, C, W, L]	R	0	0	0	0	1	1
204	TL	28	31.228	[BR, S, W, N, TL]	BR	1	0	1	0	0	0
205	TL	28	34.158	[BR, S, W, C, N, TL]	BR	1	0	1	0	0	0
206	TL	28	41.228	[BR, S, W, C, E, N, TL]	BR	1	0	1	0	0	0
207	TL	28	27.088	[BR, S, C, N, TL]	BR	1	0	1	0	0	0

Figure 4.8: Path matrix 2 - 2

389	R	52	23.070	[L, W, S, C, E, R]	L	0	0	0	0	1	1
390	R	52	20.140	[L, W, S, E, R]	L	0	0	0	0	1	1
391	BR	53	35.684	[L, W, N, C, E, S, BR]	L	0	0	1	0	0	1
392	BR	53	28.614	[L, W, N, C, S, BR]	L	0	0	1	0	0	1
393	BR	53	35.684	[L, W, N, E, C, S, BR]	L	0	0	1	0	0	1
394	BR	53	32.754	[L, W, N, E, S, BR]	L	0	0	1	0	0	1
395	BR	53	35.684	[L, W, C, N, E, S, BR]	L	0	0	1	0	0	1
396	BR	53	28.614	[L, W, C, E, S, BR]	L	0	0	1	0	0	1
397	BR	53	21.544	[L, W, C, S, BR]	L	0	0	1	0	0	1
398	BR	53	18.614	[L, W, S, BR]	L	0	0	1	0	0	1
399	B	54	30.140	[L, W, N, C, E, S, B]	L	0	0	0	0	0	1
400	B	54	23.070	[L, W, N, C, S, B]	L	0	0	0	0	0	1
401	B	54	30.140	[L, W, N, E, C, S, B]	L	0	0	0	0	0	1
402	B	54	27.210	[L, W, N, E, S, B]	L	0	0	0	0	0	1
403	B	54	30.140	[L, W, C, N, E, S, B]	L	0	0	0	0	0	1
404	B	54	23.070	[L, W, C, E, S, B]	L	0	0	0	0	0	1
405	B	54	16.000	[L, W, C, S, B]	L	0	0	0	0	0	1
406	B	54	13.070	[L, W, S, B]	L	0	0	0	0	0	1
407	BL	55	11.544	[L, W, BL]	L	0	0	0	1	0	1

Figure 4.9: Path matrix 2 - 3

use df to represent the data matrix in Fig. 4.6.

We introduce an auxiliary binary variable $x_{i,d}$ to indicate whether one pair of Inlet-Outlet chooses path d. With this new variable we can rewrite the constraints: Each pair of Inlet-Outlet chooses exactly one path.

$$\sum_{d \in \mathbb{D}} x_{i,d} = 1 \; \forall i \in \mathbb{I} \tag{4.1}$$

One path d can be chosen at most once.

$$\sum_{i \in \mathbb{I}} x_{i,d} \leq 1 \; \forall d \in \mathbb{D} \tag{4.2}$$

The path must start and end at correct points.

pin = ['TL', 'TR', 'BR', 'BL', 'R', 'L' ,'T', 'B'] in case "one GRU"

pin = ['TL', 'T', 'TR', 'BR', 'B', 'BL', 'R', 'L', 'T1', 'T2', 'B1', 'B2'] in case "two GRUs"

The order of elements in the array is not fixed. The programming code will change the order until it finds the optimal solution. Otherwise, match of in-/outlets and start-/endpoints has only one possibility, which may result in infeasible model.

$$\sum_{d \in D} x_{i,d} \times pin.index\left(df_{d,\text{"Start"}}\right) = \alpha_i \; \forall i \in \mathbb{I} \tag{4.3}$$

$$\sum_{d \in D} x_{i,d} \times pin.index\left(df_{d,\text{"End"}}\right) = \beta_i \; \forall i \in \mathbb{I} \tag{4.4}$$

If we have to avoid contamination of different paths, following constraints must be fulfilled:

Node n in those paths can be selected at most 1 time.

Matrix O, which has D rows and N columns, is part of matrix df.

N = [N, W, S, C, E] in case "one GRU"

N = [N1, W1, S1, C1, E1, N2, W2, S2, C2, E2] in case "two GRUs"

$$\sum_{i \in AC} \sum_{d \in D} x_{i,d} \times O_{d,n} \leq 1 \; \forall n \in \mathbb{N} \tag{4.5}$$

Optimization objective:

Minimize the total length of all selected paths.

$$\min L = \sum_{i \in I} \sum_{d \in D} x_{i,d} \times df_{d,"Length"} \tag{4.6}$$

subject to: (4.1) (4.2) (4.3) (4.4) (4.5)

5 Result and Analysis

In this chapter, the approach of optimization is fully tested. First, results of 4 different inputs will be implemented. Then, a comparison is made between different input protocols. Finally, performance of the optimization algorithm will be discussed.

We implement the physical synthesis models in Python, and solve them using Gurobi [24], a mixed integer linear programming (MILP) solver. For this switch design, we tested different input protocols:

• Case 1: Only one GRU is needed and there is no contamination to be avoided.

• Case 2: Only one GRU is needed but we have to avoid contamination between inlets and outlets.

• Case 3: Two GRUs are needed and there is no contamination to be avoided.

• Case 4: Two GRUs are needed but we have to avoid contamination between inlets and outlets.

5.1 Result

In this part, Fig. 5.1 - Fig. 5.19 show details of inputs and the corresponding results. In the following we analyze the results with details:

I have tested the model with four different inputs. For each input, the selected paths are expressed in the form of picture and text. Unused paths in figures are blocked with valves (orange lines). Input 1 has only 3 inlets and 2 outlets, which is the simplest example. As shown in Fig. 5.3 and in Fig. 5.4, compared to model in Columba 2.0, the model in this research has no obvious advantage in physical structure. As the number of IOs increases, the complexity has increased accordingly.

```
Communication Rule:
Inlet: [1, 2, 3]
Outlet: [4, 5]
In-Out: [[1, [4]], [2, [4, 5], [3, 5]]]
AC: [[1, [3]]]
```

Figure 5.1: Input 1 - one GRU

```
In-Out: 1 - 4    Number: 0        Path: ['TL', 'N', 'T']
In-Out: 2 - 4    Number: 263      Path: ['L', 'W', 'N', 'T']
In-Out: 2 - 5    Number: 304      Path: ['L', 'W', 'S', 'B']
In-Out: 3 - 5    Number: 136      Path: ['BR', 'S', 'B']
Total Length:49.228
```

Figure 5.2: Input 1 with AC - Output

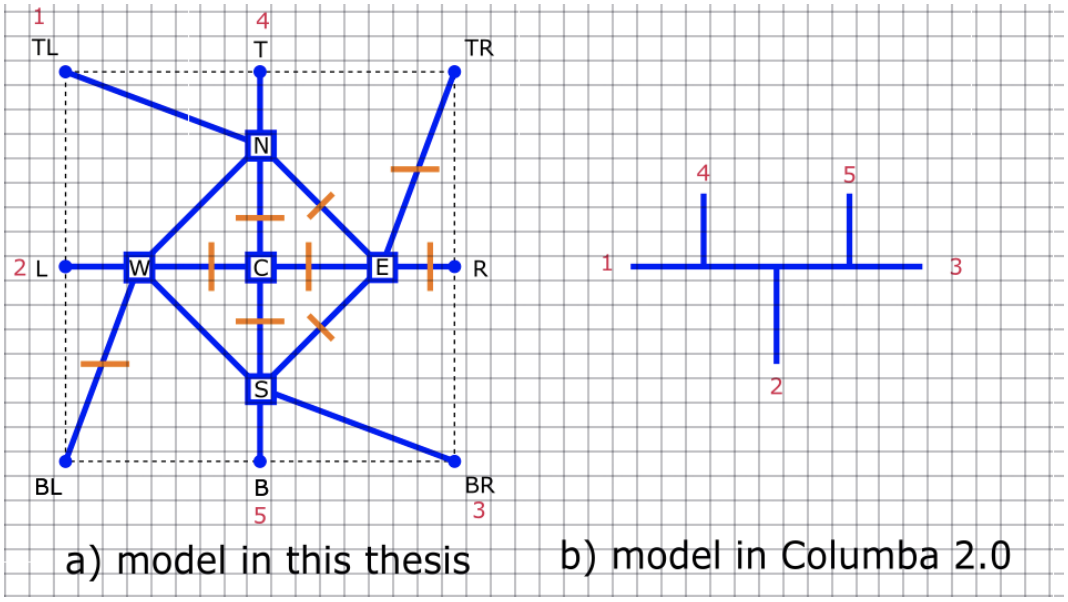

Figure 5.3: Input 1 - Path visualization

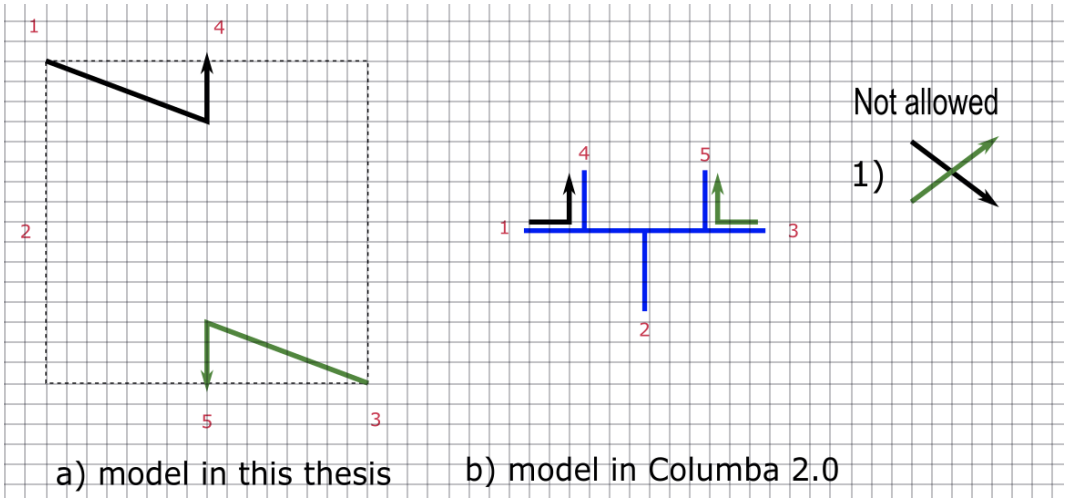

Figure 5.4: Input 1 - Comparison between different path allocation

As shown in Fig. 5.5, input 2 has five inlets and three outlets. Compared to input 1, more flow junctions are unblocked. For the same IO but with different ACs, pin assignment and path allocation are different. Fig. 5.8 visualizes the complete path allocation but Fig. 5.9 focuses on the paths which should be separated in oder to avoid contamination. Model on the left is without AC. In-Out 1 and In-Out 2 have one shared point S. Model on the right is with AC. It can be seen clearly that In-Out 1 and In-Out 2 have no shared points.

```
Communication Rule:
Inlet: [1, 2, 3, 4, 5]
Outlet: [6, 7, 8]
In-Out: [[1, [7]], [2, [6]], [3, [8]], [4, [6]], [5, [6, 7]]]
AC: [[1, [2]], [3, [4]]]
```

Figure 5.5: Input 2 - one GRU

```
In-Out: 1 - 7     Number: 369      Path: ['B', 'S', 'C', 'N', 'T']
In-Out: 2 - 6     Number: 145      Path: ['BR', 'S', 'W', 'L']
In-Out: 3 - 8     Number: 67       Path: ['TR', 'E', 'R']
In-Out: 4 - 6     Number: 203      Path: ['BL', 'W', 'L']
In-Out: 5 - 6     Number: 43       Path: ['TL', 'N', 'W', 'L']
In-Out: 5 - 7     Number: 0        Path: ['TL', 'N', 'T']
Total Length:87.860
```

Figure 5.6: Input 2 without AC - Output

The situation is the similar in example 3 and example 4. As shown in Fig. 5.14a, In-Out 3 and In-Out 4 has common point S, when we don't consider AC. In comparison, the paths of In-Out 3 and In-Out 4 are separated in Fig.

```
In-Out: 1 - 7     Number: 161      Path: ['BL', 'W', 'N', 'T']
In-Out: 2 - 6     Number: 236      Path: ['R', 'E', 'S', 'B']
In-Out: 3 - 8     Number: 8        Path: ['TL', 'N', 'E', 'TR']
In-Out: 4 - 6     Number: 136      Path: ['BR', 'S', 'B']
In-Out: 5 - 6     Number: 304      Path: ['L', 'W', 'S', 'B']
In-Out: 5 - 7     Number: 263      Path: ['L', 'W', 'N', 'T']
Total Length:93.526
```

Figure 5.7: Input 2 with AC - Output

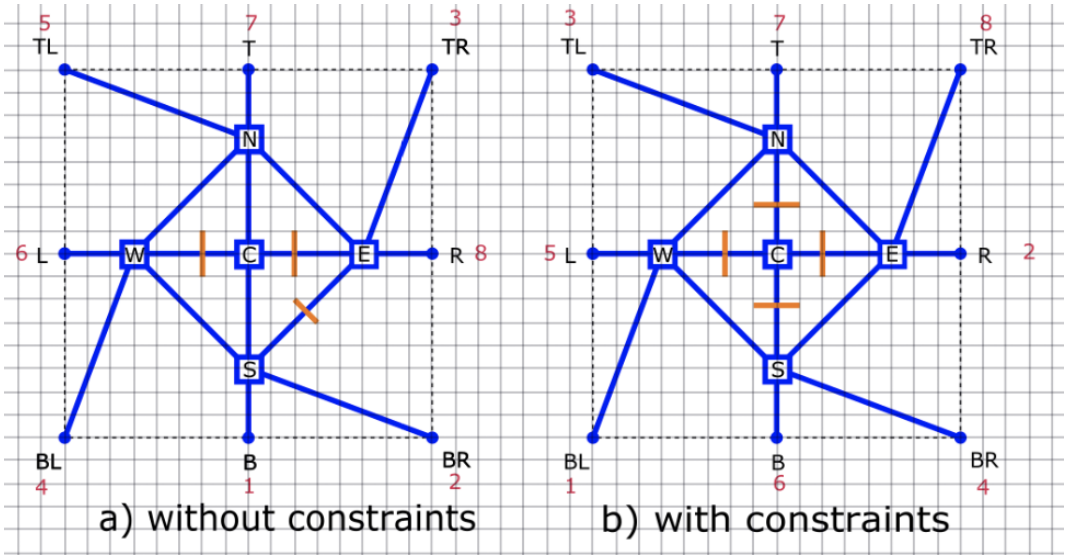

Figure 5.8: Input 2 - Path visualization

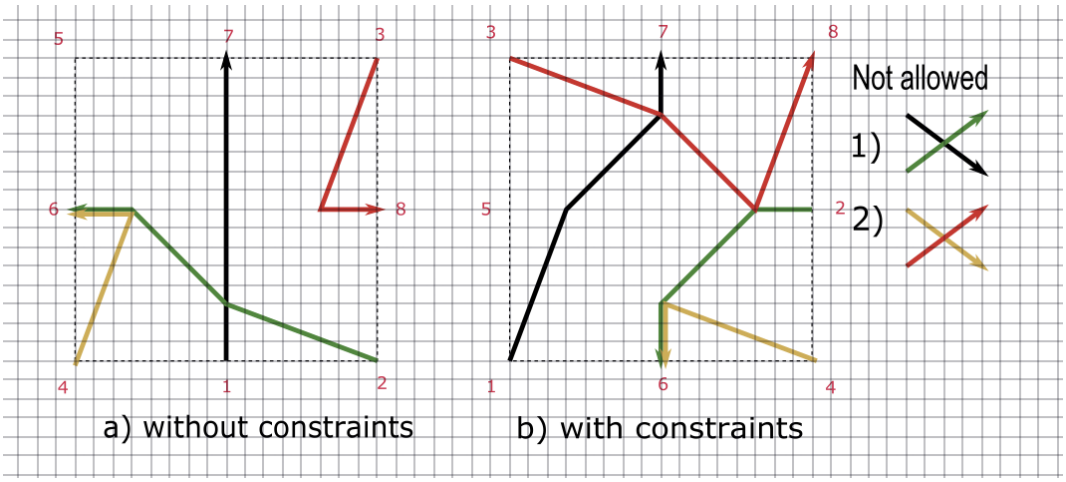

Figure 5.9: Input 2 - Comparison between different path allocation

5.14b. As shown in Fig. 5.19a, point E1 and point W2 and point C2, which are not allowed. After optimization with AC constraints, there is no more common points.

```
Communication Rule:
Inlet: [1, 2, 3, 4, 5]
Outlet: [6, 7, 8]
In-Out: [[1, [8]], [2, [6]], [3, [8]], [4, [6]], [5, [7, 6]]]
AC: [[1, [2, 6]], [3, [4, 7]]]
```

Figure 5.10: Input 3 - one GRU

```
In-Out: 1 - 8     Number: 83       Path: ['TR', 'E', 'S', 'B']
In-Out: 2 - 6     Number: 203      Path: ['BL', 'W', 'L']
In-Out: 3 - 8     Number: 236      Path: ['R', 'E', 'S', 'B']
In-Out: 4 - 6     Number: 400      Path: ['BR', 'S', 'W', 'L']
In-Out: 5 - 7     Number: 0        Path: ['TL', 'N', 'T']
In-Out: 5 - 6     Number: 43       Path: ['TL', 'N', 'W', 'L']
Total Length:92.000
```

Figure 5.11: Input 3 without AC - Output

```
In-Out: 1 - 8     Number: 284      Path: ['L', 'W', 'C', 'E', 'R']
In-Out: 2 - 6     Number: 391      Path: ['B', 'S', 'BR']
In-Out: 3 - 8     Number: 67       Path: ['TR', 'E', 'R']
In-Out: 4 - 6     Number: 194      Path: ['BL', 'W', 'S', 'BR']
In-Out: 5 - 7     Number: 306      Path: ['T', 'N', 'TL']
In-Out: 5 - 6     Number: 327      Path: ['T', 'N', 'C', 'S', 'BR']
Total Length:96.334
```

Figure 5.12: Input 3 with AC - Output

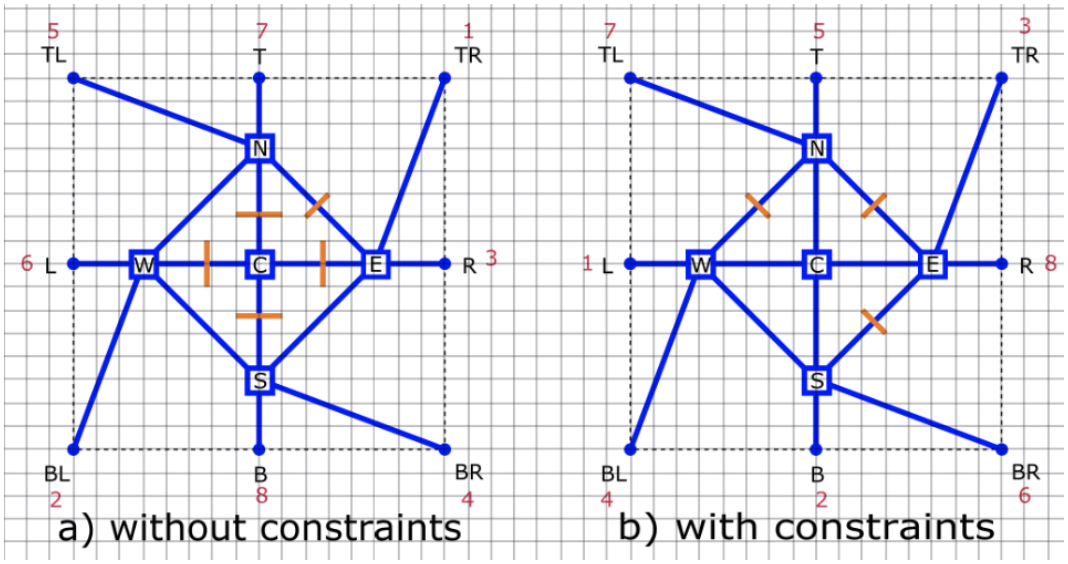

Figure 5.13: Input 3 - Path visualization

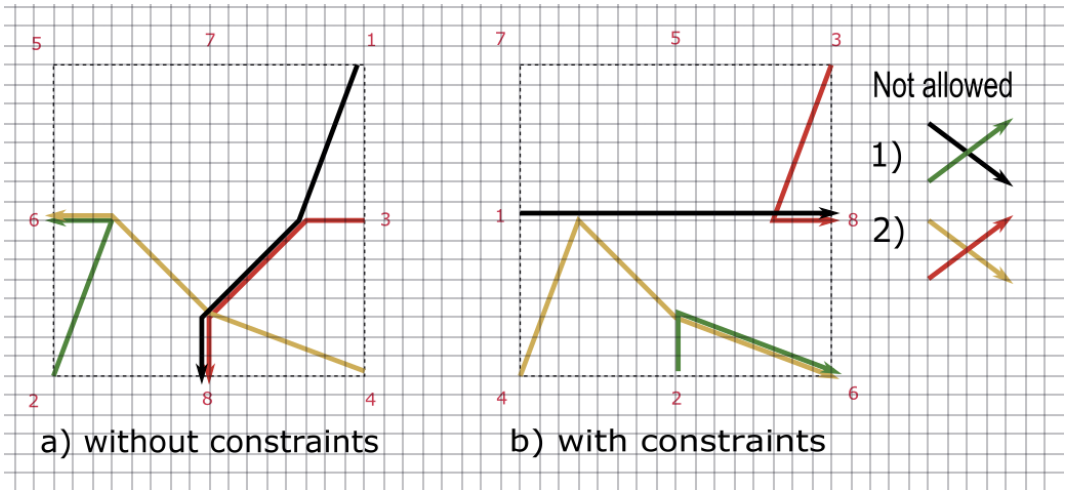

Figure 5.14: Input 3 - Comparison between different path allocation

```
Communication Rule:
Inlet: [1, 2, 3, 4, 5, 6, 7]
Outlet: [8, 9, 10, 11]
In-Out: [[1, [10, 8]], [2, [11]], [3, [10, 9]], [4, [9]], [5, [9, 8]], [6, [8, 11, 9]], [7, [9, 11, 8]]]
AC: [[1, [4, 11]], [2, [3, 5, 9]]]
```

Figure 5.15: Input 4 - two GRUS

```
In-Out: 1 - 10    Number: 9504    Path: ['L', 'W1', 'C1', 'E1', 'T']
In-Out: 1 - 8     Number: 10440   Path: ['L', 'W1', 'S1', 'B1']
In-Out: 2 - 11    Number: 8511    Path: ['BL', 'W1', 'C1', 'E1', 'W2', 'C2', 'E2', 'R']
In-Out: 3 - 10    Number: 14418   Path: ['B2', 'S2', 'C2', 'N2', 'T']
In-Out: 3 - 9     Number: 15548   Path: ['B2', 'S2', 'C2', 'N2', 'T2']
In-Out: 4 - 9     Number: 11952   Path: ['T1', 'N1', 'E1', 'W2', 'N2', 'T2']
In-Out: 5 - 9     Number: 7810    Path: ['B', 'W2', 'N2', 'T2']
In-Out: 5 - 8     Number: 7664    Path: ['B', 'S1', 'B1']
In-Out: 6 - 8     Number: 6622    Path: ['BR', 'S2', 'W2', 'E1', 'S1', 'B1']
In-Out: 6 - 11    Number: 5783    Path: ['BR', 'S2', 'E2', 'R']
In-Out: 6 - 9     Number: 6774    Path: ['BR', 'S2', 'C2', 'N2', 'T2']
In-Out: 7 - 9     Number: 1134    Path: ['TL', 'N1', 'E1', 'W2', 'N2', 'T2']
In-Out: 7 - 11    Number: 574     Path: ['TL', 'N1', 'E1', 'W2', 'C2', 'E2', 'R']
In-Out: 7 - 8     Number: 1004    Path: ['TL', 'N1', 'C1', 'S1', 'B1']
Total Length:325.684
```

Figure 5.16: Input 4 without AC - Output

```
In-Out: 1 - 10    Number: 10333    Path: ['L', 'W1', 'S1', 'B']
In-Out: 1 - 8     Number: 10766    Path: ['L', 'W1', 'S1', 'B', 'W2', 'S2', 'B2']
In-Out: 2 - 11    Number: 9014     Path: ['BL', 'W1', 'S1', 'B1']
In-Out: 3 - 10    Number: 1797     Path: ['T', 'E1', 'W2', 'B']
In-Out: 3 - 9     Number: 2178     Path: ['T', 'E1', 'N1', 'T1']
In-Out: 4 - 9     Number: 13921    Path: ['T2', 'N2', 'T', 'E1', 'N1', 'T1']
In-Out: 5 - 9     Number: 3653     Path: ['TR', 'E2', 'C2', 'W2', 'E1', 'N1', 'T1']
In-Out: 5 - 8     Number: 3982     Path: ['TR', 'E2', 'S2', 'B2']
In-Out: 6 - 8     Number: 5408     Path: ['R', 'E2', 'S2', 'B2']
In-Out: 6 - 11    Number: 5261     Path: ['R', 'E2', 'C2', 'W2', 'E1', 'S1', 'B1']
In-Out: 6 - 9     Number: 5079     Path: ['R', 'E2', 'C2', 'W2', 'E1', 'N1', 'T1']
In-Out: 7 - 9     Number: 6399     Path: ['BR', 'S2', 'W2', 'E1', 'N1', 'T1']
In-Out: 7 - 11    Number: 6622     Path: ['BR', 'S2', 'W2', 'E1', 'S1', 'B1']
In-Out: 7 - 8     Number: 6798     Path: ['BR', 'S2', 'B2']
Total Length:345.666
```

Figure 5.17: Input 4 with AC - Output

5.2 Analysis

• Path allocation

In example 2, 3 and 4, the path allocation varies with different AC. Compared with models in Columba 2.0 and Columba S, if the number of inlets and outlets is less, the generated structure is similar in this research. As the situation gets more complicated, Columba 2.0 and Columba S cannot generate an effective structure. The advantage of this switch design is as followed as below:

One structure can be used for different inputs. There is no need to regenerate new structures. We can point out routes that can be used to avoid contamination in advance. This is necessary for biological experiments.

• Run Time

For all input protocols, Gurobi synthesizes them within seconds. The

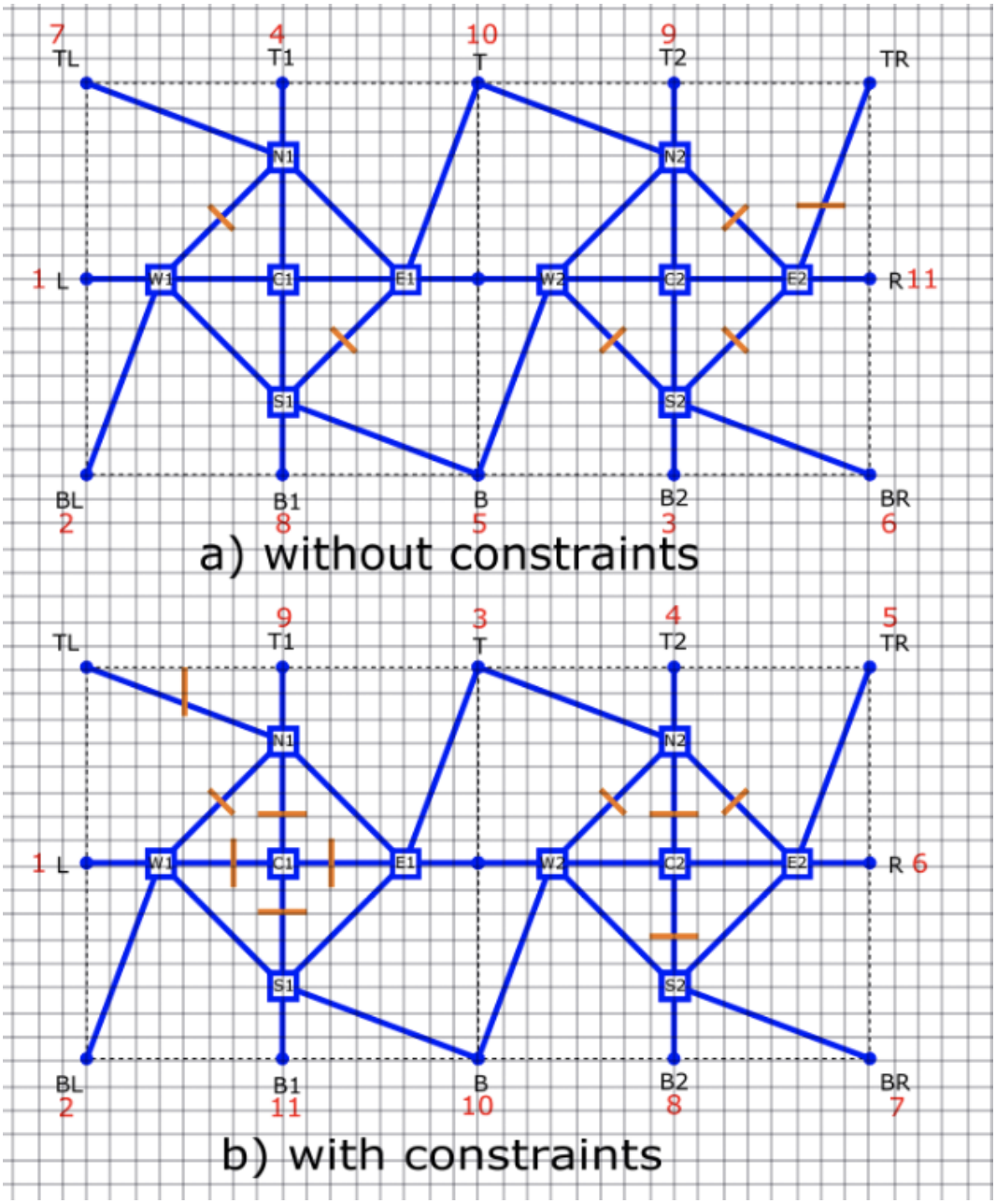

Figure 5.18: Input 4 - Path visualization

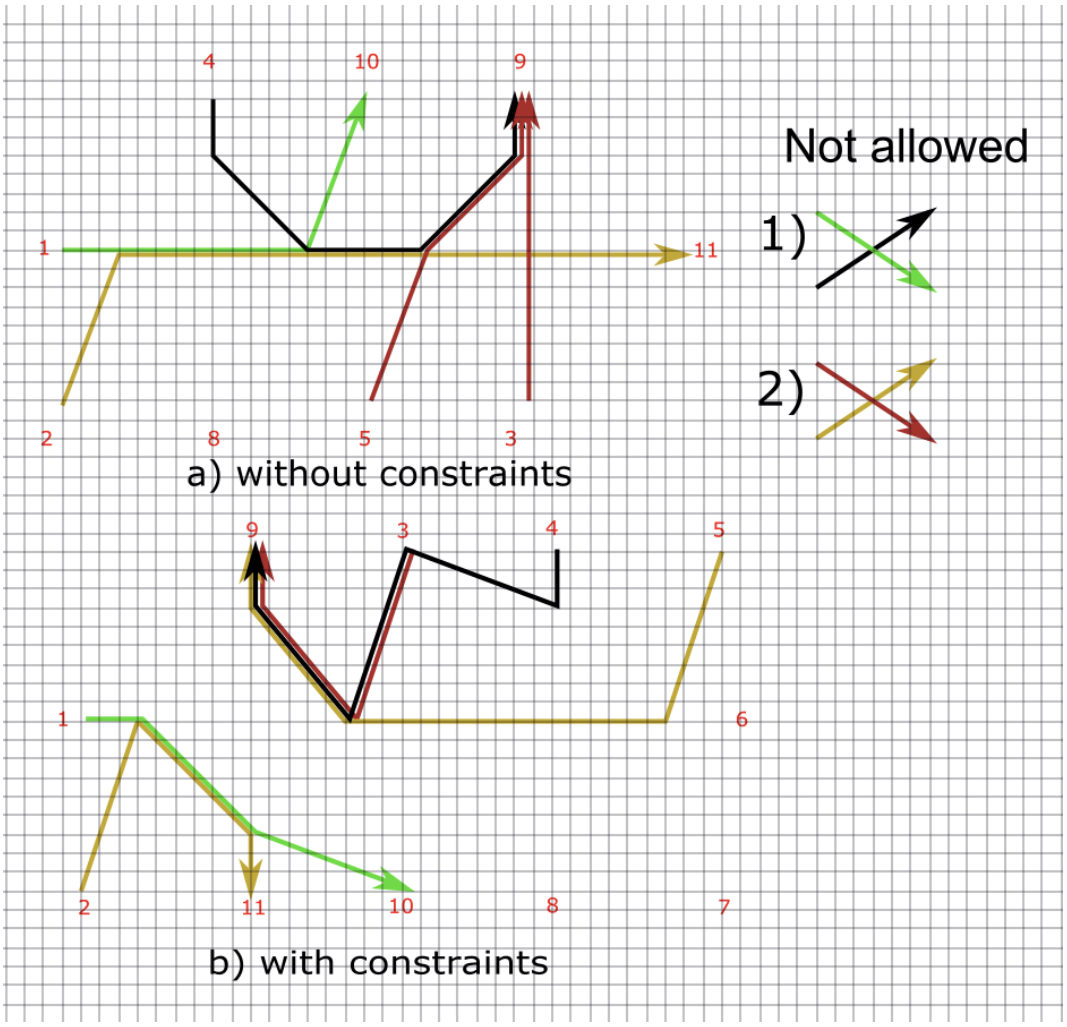

Figure 5.19: Input 4 - Comparison between different path allocation

program run time increases with the scale of GRU and the protocol. For the same IO with different AC, there is a significant improvement in time efficiency. However, for the same IO with and without AC, there is no significant improvement in time efficiency. The run time of different inputs is shown in Table 5.1 .

Table 5.1: Run time of different inputs

I + O	AC	Run time with AC	Run time without AC
3 + 2	1	0.02s	0.02s
5 + 3	1	0.04s	0.04s
5 + 3	2	0.04s	0.04s
7 + 4	2	0.35s	0.33s
7 + 4	4	0.40s	0.40s

6 Conclusion and Prospect

6.1 Conclusion

The technological progress of flow-based microfluidic biochips is continuous in the last decade. Previous studies, however, still follow the simple architectural model framework without considering contanmination avoidance. It actually becomes a bottleneck. In this paper, we propose a new architectural framework of switch, which is a place prone to cross contamination.

To construct a new switch structure a linear programming model was developed. First, the idea of General Routing Unit was used for reference. Different from only one main flow channel in the previous design, General Routing Unit provides several flow channels as main flow channels. With the help of Gurobi, which is the fastest math programming solver, the optimization of the programming model was also demonstrated. Finally, we used several tests to characterize the performance of the solver and the quality of results was obtained.

In general, this research provides feasible flow paths for each liquid considering contanmination avoidance. We can tell in advance which routes that the liquids flow through in oder to avoid cross contamination. It is conducive for experimenters doing biological experiments.

6.2 Prospect

In the future there is a huge prospect of following areas.

GRU designs. More GRU structure can be designed and they can form different physical structure to adapt to different biological experiment situation. As an example, a 3D-Structure, which might effectively reduce the crossover.

Solve times. In this paper the run time of the solver is in half of a second. As the complexity of the physical structure increases, run consumption increases.

Bibliography

[1] C. Liu, B. Li, H. Yao, P. Pop, T.-Y. Ho, and U. Schlichtmann, *Transport or store? Synthesizing flow-based microfluidic biochips using distributed channel storage*. 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, 2017.

[2] M. Li, T.-M. Tseng, Y. Ma, T. Ho, and U. Schlichtmann, *VOM: Flow-Path Validation and Control-Sequence Optimization for Multilayered Continuous-Flow Microfluidic Biochips*. 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 2019, pp. 1-8, doi: 10.1109/ICCAD45719.

[3] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, *Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37.8: 1588-1601, 2017.

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

[5] T.-M. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T.-Y. Ho, and U. Schlichtmann, *Columba S: A scalable co-layout design automation tool for microfluidic large-scale integration*. Proceedings of the 55th Annual Design Automation Conference, 2018.

[6] T. Yan and M. D. F. Wong, *A Correct Network Flow Model for Escape Routing*. Proceedings of the 46th Annual Design Automation Conference, 2009.

[7] A. Truppel, T.-M. Tseng, D. Bertozzi, J. C. Alves, and U. Schlichtmann, *PSION+: Combining logical topology and physical layout optimization for Wavelength-Routed ONoCs*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020.

[8] Anaconda, *Anaconda logo*. https://www.anaconda.com, 2004.

[9] Gurobi, *Gurobi Optimizer Quick Start Guide*. https://www.gurobi.com, 2019.