DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CHAIR OF ELECTRONIC DESIGN AUTOMATION

TECHNISCHE UNIVERSITÄT MÜNCHEN

Contamination-Free Switch Design and Synthesis for Microfluidic Large-Scale Integration

Duan Shen

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CHAIR OF ELECTRONIC DESIGN AUTOMATION

TECHNISCHE UNIVERSITÄT MÜNCHEN

Contamination-Free Switch Design and Synthesis for Microfluidic Large-Scale Integration

Master Thesis

Author:	Duan Shen
Supervisor:	Dr. -Ing. Tsun-Ming Tseng
Supervising Professor:	Prof. Dr. -Ing. Ulf Schlichtmann
Submission Date:	June 05 2021

I confirm that this is my own work and I have documented all sources and material used.

Munich, June 05 2021 Duan Shen

Acknowledgments

First of all, I would like to thank my supervisor Dr. Tsun-Ming Tseng for his constant advice about my thesis, his mental support, encouragement and trust in my abilities.

Secondly, I would like to express my gratitude to Prof. Ulf Schlichtmann for the opportunity to make my thesis at Chair of Electronic Design Automation. I also want to thank my advisor Zhidan for her helps in building the Gurobi models and using the remote server.

Most importantly, I would not have been able to afford to undertake this endeavour without the support by my parents and my friends.

Abstract

Continuous-flow microfluidic large-scale integration (mLSI) biochips have rapidly been developed and widely been used in recent decades. The gap between design efficiency and application complexity leads to a growing interest in mLSI design automation. State-of-the-art in design automation for continuous-flow microfluidics focuses on simultaneous co-optimization of both flow and control layers but neglects the potential contamination between flows. Functioning as the fluid router, microfluidic switches are likely among the most polluted modules since they locate at the intersection of flow channels. State-of-the-art tools design the switches as a spine with junctions, which makes the spine prone to fluid pollution.This thesis proposes an 8-pin, a 12-pin and a 16-pin reconfigurable microfluidic switch designs. I propose a synthesis method to produce application-specific switches reduced from the proposed switch designs. This thesis also proposes a scheduling and binding method to determine the contamination-free fluid routing paths on the switch. The proposed method groups the fluid flows that can be routed through the switch in parallel, and minimizes the number of fluid flow groups to minimize the time spent on fluid routing tasks. To save control resources, pressure sharing is considered among valves within the switch. The problem is solved with integer quadratic programming (IQP). Experimental results show that the application-specific switch designs synthesized by the proposed method are always able to avoid fluid contamination.

Contents

1 Introduction

Microfluidics is the technology that precisely manipulates reagent fluids at a nanoliter-scale or below. Continuous-flow *microfluidic large-scale integration* (mLSI) refers to the technology of microfluidic chips containing a large number of micro-mechanical valves and control components, enabling hundreds of assays to be automatically performed on a single miniaturized chip [1][2]. The common structure of an mLSI device is composed of two layers: a flow layer and a control layer. As shown in Figure 1.1, valves are formed at the intersections of the two layers [1]. By changing the pressure in the control channels through control inlets, the state of valves lying between the control layer and the flow layer can be switched, which enables the manipulation of fluids flowing on the flow layer. Since mLSI takes much less experiment space and uses less amount of reagents, it can significantly save experiment materials and time costs. Thanks to its convenient manipulation, high efficiency, spared costs and accuracy, mLSI is applied in a wide range of applications in the fields of chemistry and biology, such as polymerase chain reaction (PCR) [3], single cell mRNA isolation [4], chromatin immunoprecipitation (ChIP) [5], nucleic acid processor [6], etc.

As microfluidic experiments become more complex, the number of components integrated on the chip is increasing dramatically, which incapacitates the manual design as it would be error prone and consume much more effort and time when dealing with a large assay. Therefore, automated design tools for mLSI are highly needed to fit the growing design complexity. So far, there has been significant effort devoted to design automation research for mLSI. However, the focus has been placed on optimizing microfluidic layers separately, neglecting the interactions between the two layers. This would result in a less global and less practical solution. Recently, several attempts have been made to the physical co-design of both layers. For example, Wang et al. proposed a placement algorithm to seamlessly integrate the two interacting layers [7], and Tseng et al. proposed a design tool named Columba, a top-down approach that applies a library of microfluidic components specifying the interactions between two layers to simplify the co-layout design [8].

Despite these advances, there have been insufficient discussions to avoid fluid contamination on mLSI biochips. This problem should be given much attention considering the following

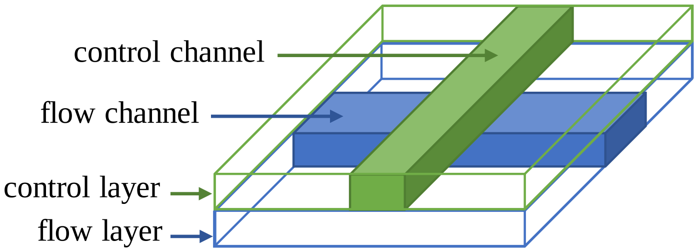


Figure 1.1: The valve structure.

facts: First, reagent fluids leave their residues when passing through the flow channels [9]. Therefore, a subsequent fluid that is conflicting with the residues may be polluted when routed through the same channel. Second, since the bioassays are manipulated with significantly reduced amounts of fluids, the resultant concentration of contamination is amplified, causing seriously erroneous outcomes and thus drastically reducing the accuracy. Taking PCR as an example, as described in [9], if flows of different DNA samples go through the same channel segment, the residual left by the former sample will pollute the later one, leading to an unexpected experiment result. Third, fluids in an mLSI device tend to use the same flow segment, especially within the microfluidic switches. In the current synthesis tools for continuous-flow microfluidics, the switch is among the modules most likely to be polluted. For example, in Columba [8], the switch is designed as a spine with junctions, which leaves inadequate space for conflicting fluids to avoid forming overlapping transportation paths. However, the previous literature either demonstrated a contamination-free method for a specific bioassay [10] or introduced washing operations into the biochip design to clean up the flow channels [9]. So far, there has been no attempt to solve this problem by optimizing the switch module.

This thesis aims to propose a contamination-free switch design methodology for automation synthesis tools of mLSI. The contributions include:

- This thesis proposes to synthesize a switch that avoids the fluid contamination. The switch structure is designed reduced from a crossbar, providing more routing possibilities than a spine. The switch is given in three sizes, 8-pin, 12-pin and 16-pin, to be applied to different sizes of switch inputs. It is configurable by removing the unnecessary channels and valves.

- This thesis proposes a switch input scheduling method to efficiently arrange the flows by grouping the flows that can simultaneously go through the switch. Each flow group is executed at different time to prevent fluid collisions or undesired flow routing.

- This thesis provides multiple binding policies between the flow pins and the connected modules. The flow pins are connected to other modules that are serving as the sources or the destinations of the flows. Different binding policies can increase the design flexibility.

- This thesis provides pressure sharing as an optional feature. Since control inlets occupy much more chip area compared to microfluidic channels, the number of control inlets is strongly limited. In addition to removing the unnecessary valves, the pressure sharing method minimizes the number of control inlets. This groups the valves that can share the same pressure sequence, so that they are able to reuse the same control inlet. Yet the control channel routing of pressure sharing lies beyond the scope of this thesis.

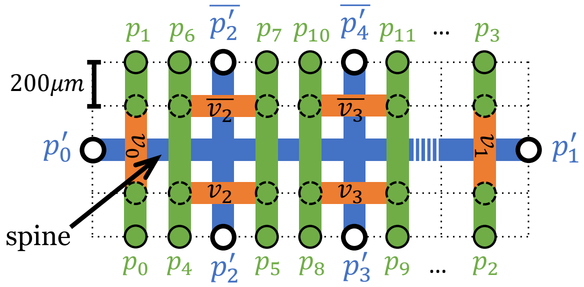
The proposed methods take flow inputs and contamination flow pairs as inputs, and synthesizes the optimal design by solving an integer quadratic programming (IQP) model.

The rest of this paper is organized as follows. Section 2 introduces the switch of the prior studies, describes the general structure, and formulates the problem model to be solved in this thesis. Section 3 then explains the proposed methods in detail. Section 4 shows and analyzes the results of multiple test cases. Section 5 discusses the results. A general conclusion is summarized in Section 6.

2 Switch Model

This section first introduces the switch structure that is designed by other studies. Then the general structure of the proposed switch is described. And the problem is formulated.

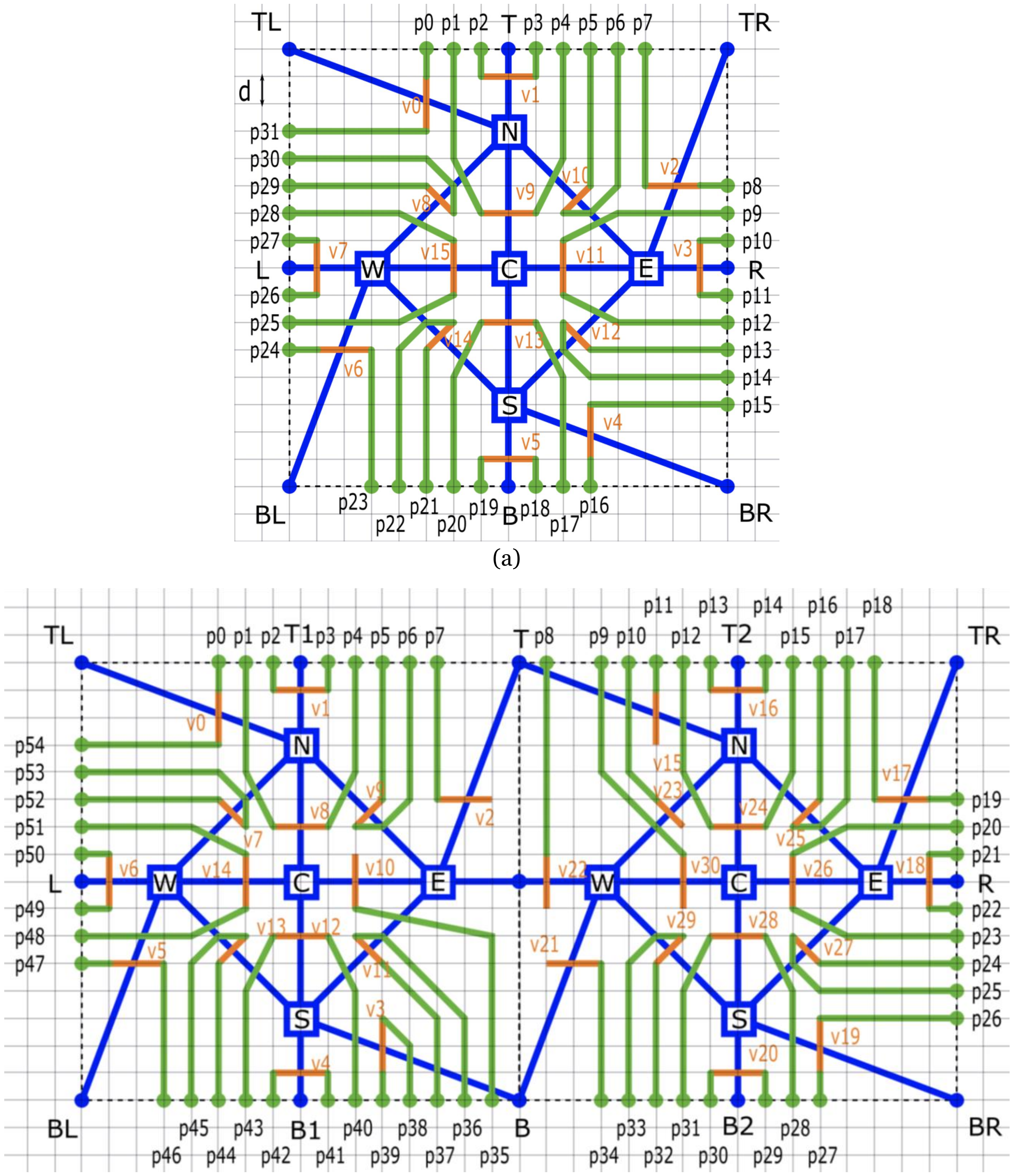
2.1 Switch Module of Microfluidics



p_1 p_6 $\overline{p_2'}$ p_7 p_{10} $\overline{p_4'}$ p_{11} ... p_3
200μm
$\overline{v_2}$
$\overline{v_3}$
p_0'
v_0
v_1
p_1'
v_2
v_3
spine
p_0 p_4 p_2' p_5 p_8 p_3' p_9 ... p_2

Figure 2.1: Spine-like switch structure in Columba [8].

Currently, there has been a new trend to design microfluidic synthesis tools in a top-down methodology, which builds a library of microfluidic components to simplify the physical synthesis [11]. Taking Columba [8][12][13] as an example, it is a top-down physical synthesis tool for continuous-flow microfluidics that provides a library of module models, including mixers, reaction chambers, etc. The module models precisely describe the inner module structure as well as the layer interactions. Figure 2.1 shows the switch module model provided by the module library of Columba, which draws the flow channels in blue, control channels in green, and valves as orange rectangles. As presented, Columba designed the switch basically as a spine with junctions, which cannot support contamination-free fluid transmissions as discussed previously in Section 1.


TL
p0 p1 p2 T p3 p4 p5 p6 p7
TR
d
p31
p30
p29
p28
p27
L
p26
p25
p24
p8
p9
p10
R
p11
p12
p13
p14
p15
N
W
C
E
S
v0
v1
v2
v3
v4
v5
v6
v7
v8
v9
v10
v11
v12
v13
v14
v15
BL
p23 p22 p21 p20 p19 B p18 p17 p16
BR
TL
p0 p1 p2 T1 p3 p4 p5 p6 p7
T p8
p9 p10 p11 p12 p13 T2 p14 p15 p16 p17 p18
TR
p54
p53
p52
p51
p50
L
p49
p48
p47
p19
p20
p21
R
p22
p23
p24
p25
p26
v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14
v15 v16 v17 v18 v19 v20 v21 v22 v23 v24 v25 v26 v27 v28 v29 v30
BL
p46 p45 p44 p43 p42 B1 p41 p40 p39 p38 p37 p36 p35 B
p34 p33 p32 p31 p30 B2 p29 p28 p27
BR

(a)

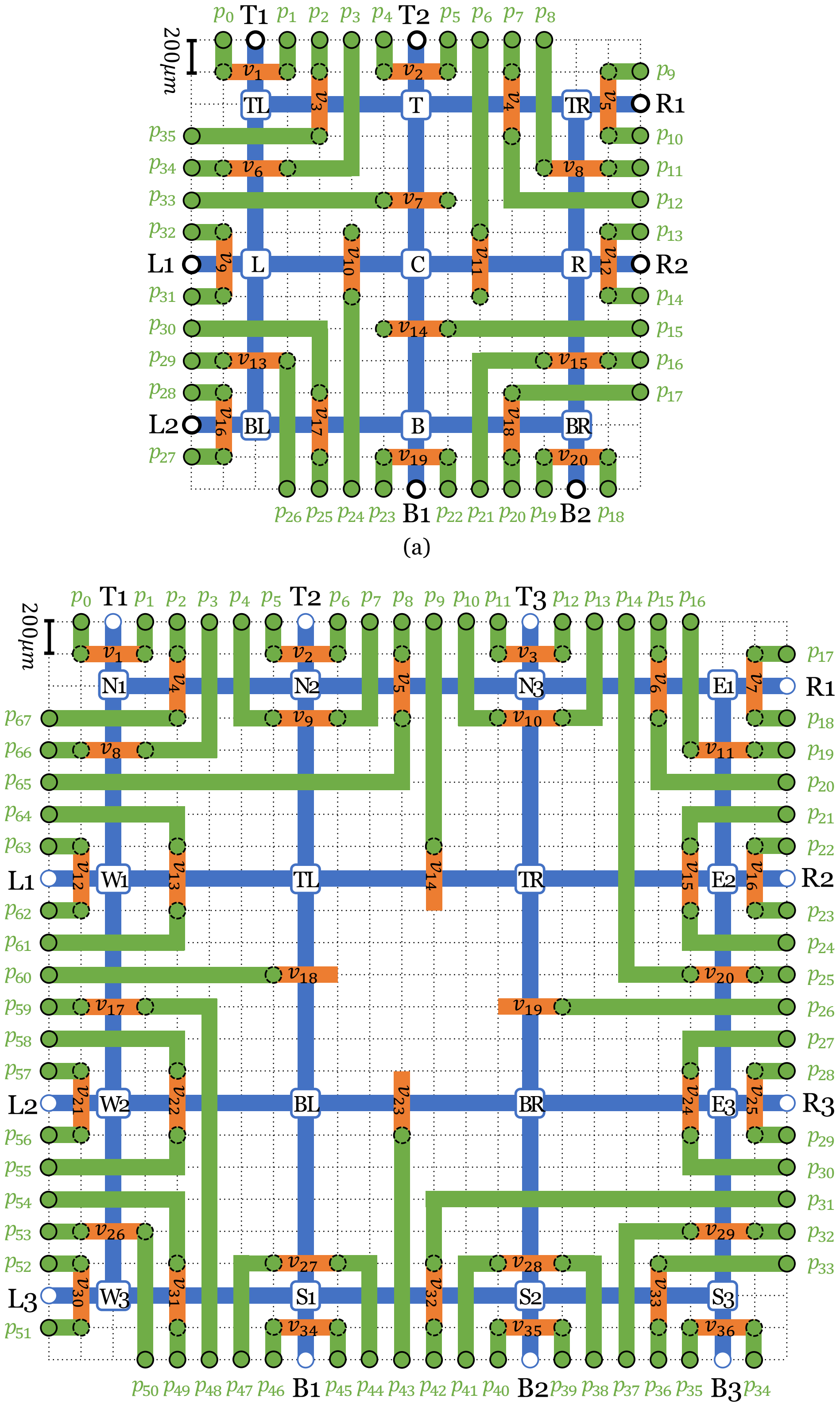
(b)

Figure 2.2: Switch structures designed by a previous study[14]: (a) of one GRU. (b) of two GRUs.

My colleage Ma has recently developed a microfluidic switch that attempted to avoid the fluid contamination [14]. Her switch structure was inspired by a network flow model [15] and it was constructed as General Routing Units (GRUs) which was proposed by Truppel [16]. As shown in Figure 2.2 (a), the switch can be an 8-pin switch consisting of one GRU. It can also be extended by connecting multiple GRUs. For example, Figure 2.2 (b) shows a 12-pin switch consisting of two GRUs. However, this design is suffering from considerable design limitations. First, the structure provides insufficient routing space for contamination avoidance. Each node is connected to two pins. Here I call those intersections of flow segments as nodes, such as C, N, E, W and S. I call those flow channel ends that can be connected by other modules as flow pins, such as TL, T, TR, R, BR, B, BL and L. It can be observed from Figure 2.2 that the flow pins TL and T are connected to the same and only node N. Problem occurs when two conflicting flows are from pin TL and T, passing by the node N without other routing choices. Second, the lack of routing choice could also result in a flow collision even though the collided flows do not cause contamination. For example, if two flows are going from pin L and pin BL simultaneously, they would come across with each other at the intersection node W. With the valves open for corresponding flows, these flows might be routed into wrong flow channels. Third, the flow channels are placed too close. For example, in one-GRU structure, the angle between the flow segments N-W and W-C is about 45°. Such closed channels could increase the possibility of reagent residual at the turning nodes. Fourth, the control channels are placed too close, which does not meet the standards. According to the design rule of Stanford Foundry [17], the minimum space between channels should be $100\mu m$. While, for instance in the one-GRU structure, the control channels extending from pins $p6$ and $p9$ are beyond the standard.

2.2 General Switch Structure

To overcome the contamination problem, this thesis designs the switch module as a crossbar-like structure, as presented in Figure 2.3 and Figure 2.4. After the physical synthesis, the unused channel segments and valves will be removed to generate an application-specific switch. The switch structure design follows the design rule of Stanford Foundry [17], which defines the flow channel width and valve length to be $100\mu m$, the valve channel width to be $300\mu m$, and the minimum space between channels to be $100\mu m$. As shown in Figure 2.3(a)(b) and Figure 2.4, this thesis provides three sizes, 8-pin, 12-pin and 16-pin, for this switch, so that this switch design is able to support different sizes of applications.


200μm
p_0 T1 p_1 p_2 p_3 p_4 T2 p_5 p_6 p_7 p_8
p_9 R1 p_{10} p_{11} p_{12} p_{13} R2 p_{14} p_{15} p_{16} p_{17}
p_{35} p_{34} p_{33} p_{32} L1 p_{31} p_{30} p_{29} p_{28} L2 p_{27}
p_{26} p_{25} p_{24} p_{23} B1 p_{22} p_{21} p_{20} p_{19} B2 p_{18}
TL T TR L C R BL B BR
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_{15} v_{16} v_{17} v_{18} v_{19} v_{20}
200μm
p_0 T1 p_1 p_2 p_3 p_4 p_5 T2 p_6 p_7 p_8 p_9 p_{10} p_{11} T3 p_{12} p_{13} p_{14} p_{15} p_{16}
p_{17} R1 p_{18} p_{19} p_{20} p_{21} p_{22} R2 p_{23} p_{24} p_{25} p_{26} p_{27} p_{28} R3 p_{29} p_{30} p_{31} p_{32} p_{33}
p_{67} p_{66} p_{65} p_{64} p_{63} L1 p_{62} p_{61} p_{60} p_{59} p_{58} p_{57} L2 p_{56} p_{55} p_{54} p_{53} p_{52} L3 p_{51}
p_{50} p_{49} p_{48} p_{47} p_{46} B1 p_{45} p_{44} p_{43} p_{42} p_{41} p_{40} B2 p_{39} p_{38} p_{37} p_{36} p_{35} B3 p_{34}
N1 N2 N3 E1 W1 TL TR E2 W2 BL BR E3 W3 S1 S2 S3
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_{15} v_{16} v_{17} v_{18} v_{19} v_{20} v_{21} v_{22} v_{23} v_{24} v_{25} v_{26} v_{27} v_{28} v_{29} v_{30} v_{31} v_{32} v_{33} v_{34} v_{35} v_{36}

(a)

(b)

Figure 2.3: Switch structures of the present thesis: (a) 8-pin. (b) 12-pin.

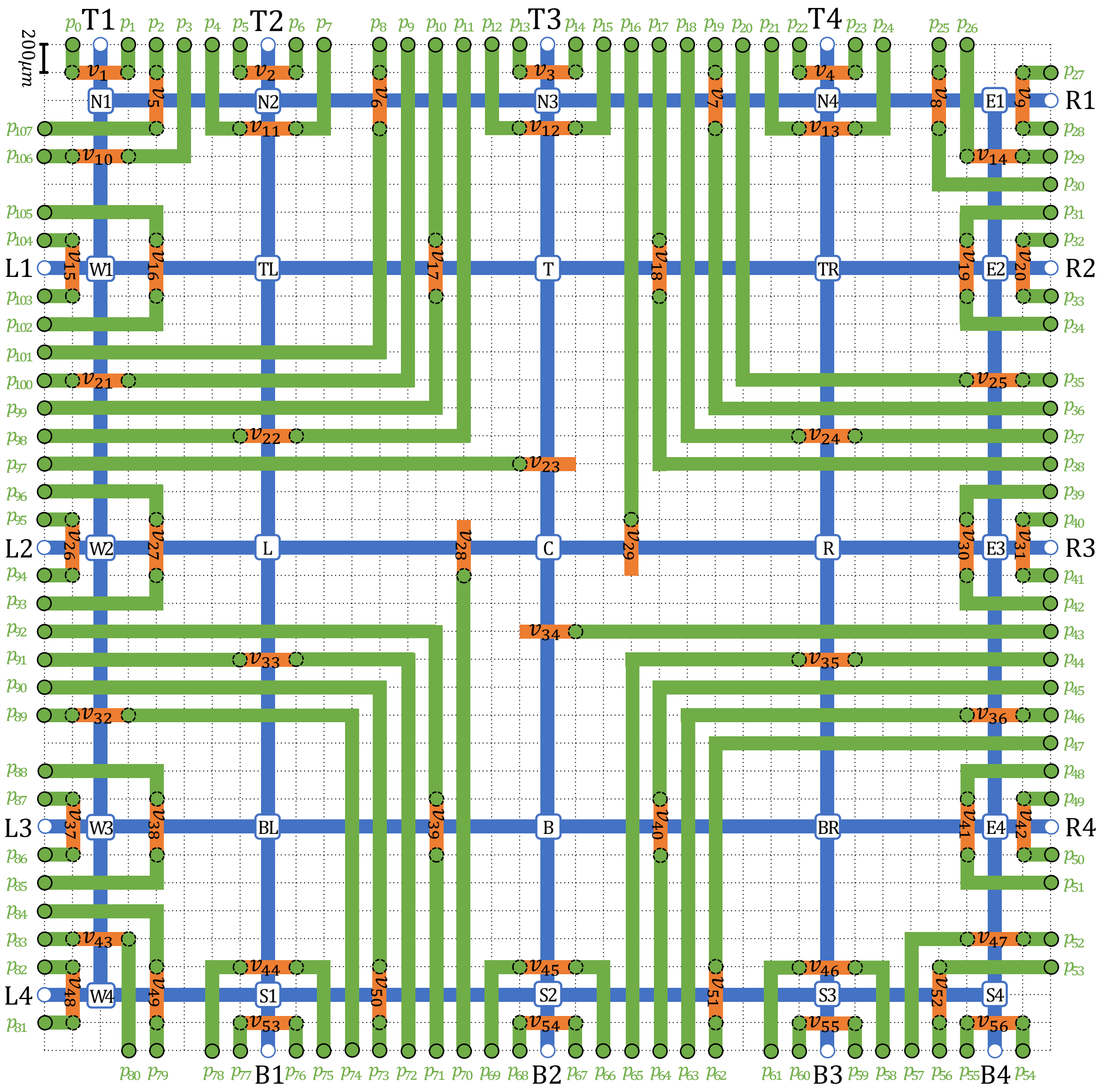
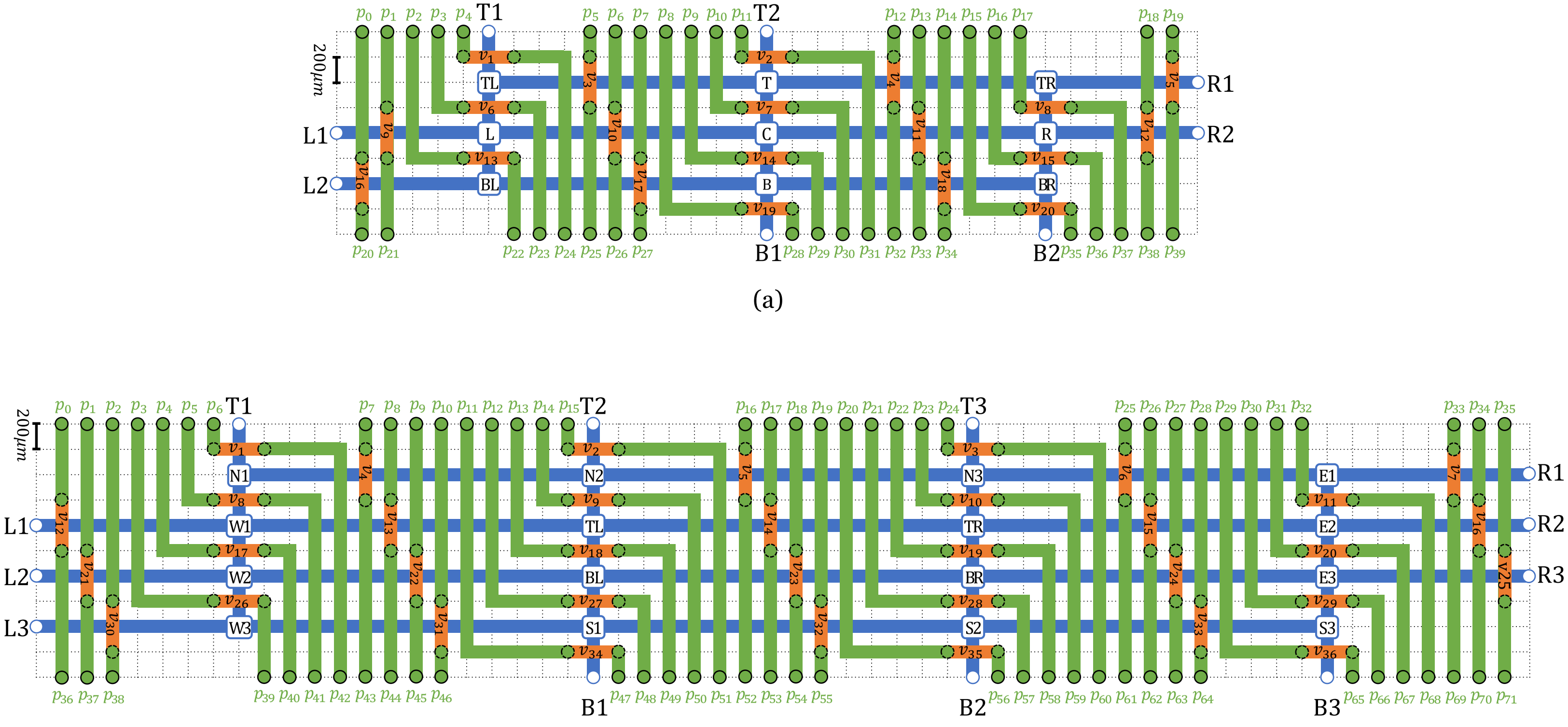


Figure 2.4: Switch structure of the present thesis, 16-pin.

In the aspect of scalability, Columba S [13] has developed a feasible implementation. It modified its module models to be accessed horizontally by flow channels and vertically by control channels, and applies multiplexers for valve control. To be compatible to Columba S, this work also draws the switch structure for the consideration of scalability, as presented in Figure 2.5 and Figure 2.6.

Throughout this paper, the term *pins* will refer to the switch pins that connect to the other modules by flow channels. The term *nodes* will refer to the intermediate nodes of a switch in the flow layer. The term *flow segments* will refer to the flow channel edges between nodes or between nodes and pins. For example, in the 8-pin switch, the pins are $T1, T2, R1, R2, B2, B1, L2, L1$. The nodes are C, T, L, R, B. There are 20 flow segments in the 8-pin switch, such as $T1$-TL and TL-T. Also, each valve is designed to be accessed by at least one control channel. Those valves with two possible connected control channels can be accessed by either or both of the channels.

Moreover, the switch distributes its flow pins nearly evenly on the border, so that it can be reached by the connected modules from all directions. The modules serve as the sources or the destinations of the flows. They should have a one-to-one binding relation with the flow pins. This thesis provides three different binding policies: fixed, clockwise and unfixed. The fixed policy binds the connected modules strictly to specified flow pins according to the input. This setting has the smallest problem space but also the least flexibility. The clockwise policy binds the modules with the flow pins according to a user-defined order of modules. The modules are sequentially assigned to the flow pins so that they are arranged in clockwise direction around the switch. For example, modules A, B, C, D are the connecting modules to an 8-pin-switch. The specified order is $\{A, B, C, D\}$. The available flow pins are $\{T1, T2, R1, R2, B2, B1, L2, L1\}$ in clockwise order around the switch. A clockwise policy may bind the modules $\{A, B, C, D\}$ to $\{T1, T2, R1, R2\}$ respectively. It may also skip pins and make the modules bind to $\{R1, B1, L1, T1\}$ respectively. Same as most other module models, the switch of this thesis is allowed to be rotated. This policy can be applicable to those design tools, for instance Columba [8] [12] [13], which locate the modules before routing, while leaving rotation space for the modules. These synthesis tools determine the relative location order of the modules after the placement phase. The clockwise policy therefore takes the order as input, and routes the modules to the flow pins, thus enabling a relatively more flexible routing solution than the fixed policy. The unfixed policy does not assign the modules to specific flow pins. This policy has the largest problem space and also the largest flexibility, whereas it might require the synthesis tool to conduct the module placement afterwards. The switch model is supposed to be able to support any application using the unfixed policy.



(a)

(b)

Figure 2.5: Scalable switch structures compatible to Columba S. (a) of 8-pin switch. (b) of 12-pin switch.

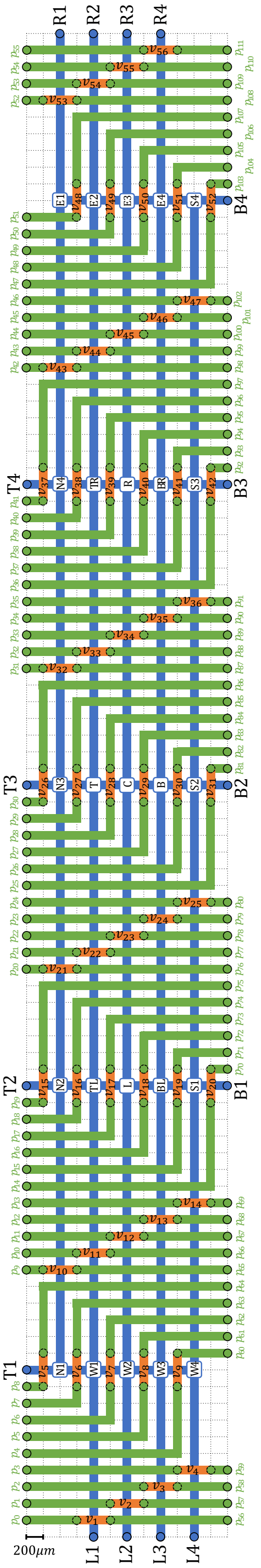


Figure 2.6: Scalable 16-pin switch structure compatible to Columba S.

2.3 Problem Formulation

The general model for the contamination-free switch is formulated as follows:

Input:

- all groups of flows to be executed,
- the conflicting flow pairs,
- the binding option (either fixed, clockwise or unfixed),
- the order of connected input/output modules (if using clockwise binding policy).

Output:

- sets of parallel-executable flows,
- routing paths of the flows without risk of contamination,
- binding pairs between modules and switch pins,
- used flow channels, and the total flow channel length.
- used valves, and the information of pressure-sharing valves.
- the program runtime.

Objective:

- to minimize the flow channel length,
- to minimize the number of sets of flows.

The mathematical model will be described in detail in the next section.

3 Mathematical model

This thesis proposes to synthesize application-specific switches reduced from the switch models. This thesis constructs an integer quadratic programming (IQP) model for the synthesis problem and send the model to an optimization solver to calculate the optimal solution. This section explains the variables, constraints, and objectives introduced in the IQP model to implement the contamination-free switch with optimized flow scheduling and minimized flow channel length.

3.1 Path Assignment For Flows

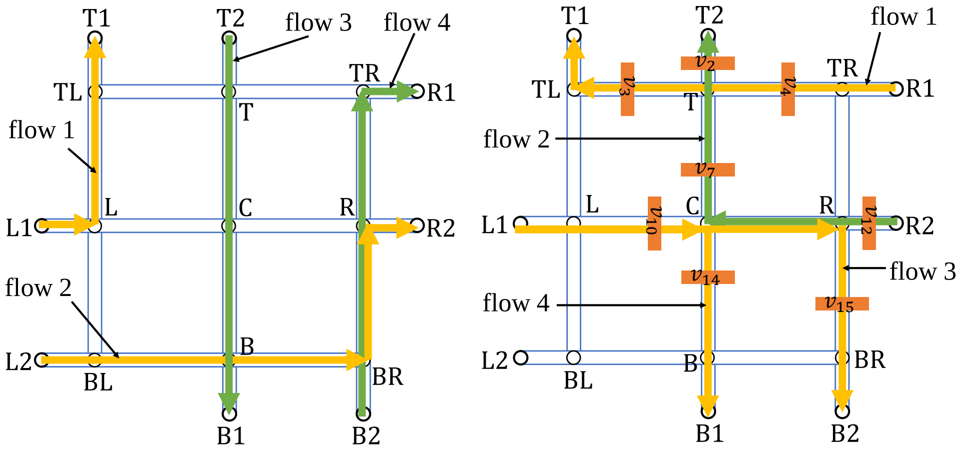
Switches are deployed to route the flows when the flow channels intersect. The first general task of a switch is thus to assign a switch flow path to each flow. To avoid flow detour, this thesis generates in advance a set of shortest paths that route between each pair of flow pins, and then assigns each flow to one of the paths. A binary variable $x_{i,d}$ is introduced to indicate whether flow i chooses path d, for each $i \in Flows$ and $d \in Paths$, where $Flows$ indicates the set of flow indices and $Paths$ indicates the set of path indices. Each flow should choose exactly one path. Each path should be chosen at most once. The corresponding constraints are as follows:

$$\sum_{d \in Paths} x_{i,d} = 1, \forall i \in Flows, \tag{3.1}$$

$$\sum_{i \in Flows} x_{i,d} \leq 1, \forall d \in Paths. \tag{3.2}$$

3.2 Fluid Contamination Avoidance

Fluid contamination happens when conflicting flows use the same flow segment or node. The polluted segment or node can result in erroneous outcomes. The contamination avoidance is implemented by forcing the conflicting flows to choose paths without intersections or



(a) (b)

Figure 3.1: Switch models of example cases (in the flow layer) for (a) contamination avoidance, where the fluid flows with the same color indicate a high risk of contamination. (b) flow scheduling, where the fluid flows with the same color indicate that they are scheduled to be routed in the same flow set and they are executed at the same time.

overlapping. In other words, each node in the conflicting paths should be used at most once. Here a binary constant $hasNode_{d,n}$ is indicated to indicate whether path d uses node n, for each $d \in Paths$ and $n \in Nodes$, where $Nodes$ indicates the set of nodes of the switch. For example, $Nodes$ of an 8-pin switch is $\{C, T, R, B, L\}$. The set of conflicting flow pairs is denoted as CF. For example, the flow indices are $Flows = \{1, 2, 3, 4\}$. If there are conflicts between flow 1 and 2, and between 3 and 4, the set CF is therefore $\{\{1, 2\}, \{3, 4\}\}$. The switch shown in Figure 3.1(a) is a possible solution for the example. The paths of flow 1 and flow 2 do not share the same node or segment. The flow 3 and flow 4 are also routed separately apart. Therefore, no consequent fluid contaminations will occur. The constraint is as follows:

$$\sum_{d\in Paths}\sum_{i\in CF} x_{i,d} \cdot hasNode_{d,n} \le 1, \forall n \in Nodes. \tag{3.3}$$

3.3 Flow Scheduling

Considering a great number of flows that should be routed through the switch, it is necessary to schedule these flows into multiple sets to be executed separately. It is also important that each flow segment cannot simultaneously be occupied by multiple flows unless for the branching flows from the same inlet. The term *flow sets* will be used in this paper to refer to such sets of parallel executable flows. Flow scheduling allows the flow segments and nodes to be reused in different flow sets, which means, each node should be used by the flows from at most one inlet in each flow set. The switch shown in Figure 3.1(b) shows an example of flow scheduling. Flows 1, 2, 3 and 4 are routed through different paths, but flow 2 is sharing the same segment *C*-*R* with flow 3, sharing the same node *T* with flow 1 and the same node *C* with flow 4. If flow 2 is executed together with any of the other three flows, an unexpected collision may occur. Furthermore, since the corresponding valves along the paths of flows 1, 3 and 4 are open, flow 2 may go to pins *T*1, *B*1 or *B*2. To avoid this, flow 2 should be assigned to a different flow set. While the flows 3 and 4 are the branches stemming from inlet *L*1, they are allowed to be executed in the same flow set.

An integer variable $K_{n,s}$ is introduced to count the usage of node n by all flows in flow set s, and an integer variable $k_{p,n,s}$ is introduced to count the usage of node n by flows from inlet pin p in flow set s. The value of $K_{n,s}$ is a summation of $k_{p,n,s}$ over inlet pin p. When $k_{p,n,s}$ is nonzero, the node n is used by flows from inlet pin p in the flow set s. In this case, flows from all other inlet pins $p^{'}$ cannot use this node, leading the $k_{p^{'},n,s}$ value to be zero, and thus the value of $K_{n,s}$ is equal to $k_{p,n,s}$. The problem is equal to model the conditional relation that if $k_{p,n,s} \geq 1$, then $k_{p,n,s} = K_{n,s}$. An auxiliary binary variable $q^{'}_{p,n,s}$ is introduced for each inlet pin p, at each node n in each flow set s to indicate whether the node n is used by flows from pin p in flow set s.

$$k_{p,n,s} \geq 1 - q^{'}_{p,n,s} \cdot N\_Pins, \tag{3.4}$$

$$k_{p,n,s} \leq K_{n,s} + q^{'}_{p,n,s} \cdot N\_Pins, \tag{3.5}$$

$$k_{p,n,s} \geq K_{n,s} - q^{'}_{p,n,s} \cdot N\_Pins, \tag{3.6}$$

with respect to $\forall p \in InletPins, \forall s \in FlowSets$, and $\forall n \in Nodes$, where $FlowSets$ is the set of flow set indices. N_Pins is an integer constant indicating the number of flow pins. It is the smallest constant that ensures the correctness of constraints (4)–(6). Based on the previous experience, a smaller constant usually results in a faster optimization. When $q^{'}_{p,n,s}$ is set to 1, constraints (4)–(6) become $k_{p,n,s} \geq 1 \wedge k_{p,n,s} = K_{n,s}$, but when $q^{'}_{p,n,s}$ is set to 0, constraints (4)–(6) become tautology regardless of the values of $k_{p,n,s}$ and $K_{n,s}$.

It is worth mentioning that for the flows that have no risk to pollute each other, they can share the same flow segments and nodes as long as they are assigned to different flow sets to pass the switch at different time. However, for contamination-prone flows, they cannot occupy the same flow segments or nodes even when they are assigned to different flow sets.

The whole functional model is now completed. The model is formulated as follows:

$$Minimize : \sum \alpha \cdot N\_Sets + \beta \cdot L_{flow} \tag{3.7}$$

$$Subject\,to : constraint\,(1) - (6). \tag{3.8}$$

where *N_Sets* represents the number of flow sets. L_{flow} represents the total length of flow channels. α and β are the weight coefficients, which are defined by the application input.

3.4 Binding Policies

This thesis provides multiple binding policies between the flow pins and the connected modules. Each module is either the source or the destination of a flow. And each module should be bound to a unique flow pin, which means that each flow pin should be used by at most one module. A binary variable $y_{m,p}$ is introduced to indicate whether module m is bound to switch pin p, for each $m \in Modules$ and $p \in Pins$, where *Modules* indicates the set of connected modules and *Pins* indicates the set of flow pins. For example, *Pins* for an 8-pin switch is $\{T1, T2, R1, R2, B2, B1, L2, L1\}$. Each module should choose exactly one flow pin, while each pin should be chosen at most once by the modules. The corresponding constraints are as follows:

$$\sum_{p \in Pins} y_{m,p} = 1, \forall m \in Modules, \tag{3.9}$$

$$\sum_{m \in Modules} y_{m,p} \leq 1, \forall p \in Pins. \tag{3.10}$$

The current design examines three policies of module-to-pin binding: **fixed**, **clockwise** and **unfixed**. A fixed policy specifies the module-pin pairs as the switch input, which requires a constraint to bind the modules to the specified pins.

$$y_{m,p} = 1, \forall (m,p) \in MP, \tag{3.11}$$

where (m,p) is a module-pin pair of the set of fixed module-pin pairs MP.

For the clockwise policy, the module indices are first read in a clockwise order, and then the following constraints are constructed to assign modules to the pins sequentially around the switch. An integer variable pin_m is introduced to indicate the pin index to which module m is bound, for each $m \in Modules$. The indices of pins $T1, T2, R1, R2, B2, B1, L2, L1$ are respectively 1 to 8. An auxiliary binary variable q_m is introduced for each module m to indicate whether the assigned pin index is the largest among the assigned pins. There should be exactly one module that satisfies $q_m = 1$.

$$pin_a \leq pin_b - 1 + q_a \cdot N\_Pins, \forall a \in Modules, \tag{3.12}$$

$$\sum_{m \in Modules} q_m = 1, \tag{3.13}$$

where b is the module that is successive to a in the order. If a is the last in the order, then b is the first module. To give an example, four modules are bound to flow pins of an 8-pin switch, with the specified order m_1, m_2, m_3, m_4. A possible binding result is m_1-$T2$, m_2-$R1$, m_3-$B1$, m_4-$L1$, with the pin indices as 2, 3, 6, and 8. In this case, according to constraint (12), q_{m_1}, q_{m_2}, and q_{m_3} can all be set to 0 or 1, while q_{m_4} must be set to 1 or constraint (12) would be violated. This, according to constraint (13), implies that q_{m_1}, q_{m_2}, and q_{m_3} all must be 0. Another result may be m_1-$B1$, $m2$-$L2$, m_3-$T1$, m_4-$T2$, with the pin indices as 6, 7, 1 and 2. The pin index assigned to module m_2 is the largest and thus the variable q_{m_2} is set to be 1.

The unfixed binding policy requires no extra constraints.

3.5 Pressure Sharing

Before minimizing the number of pressure sources, the essential valves are identified and the unnecessary ones are removed. An essential valve is recognized by comparing the flows passing through the valve and the flows passing through the neighbor segments of the valve. Here, this thesis defines an action *carry* of a valve if the valve is open for flows to pass it. If the valve can carry all flows in the neighbor segments, the valve can always be at the open status. Removing the valve does not affect the flow routing, hence such a valve is regarded as an unnecessary valve. Taking the case drawn in Figure 3.1(b) as an example, the valve on the segment C-R carries the flows 2 and 3, coming from the inlet pins $R2$ and $L1$. Its neighbor segments are L-C, T-C, R-$R2$, C-B, after removing the unused segment TR-R. The neighbor segments carry the flows from either pin $R2$ or $L1$. The valve on segment C-R is regarded as unnecessary, since it can always be at the open status.

As the valves are controlled by control inlets, with a standard size of 1mm^2, a large number of control inlets can take considerable chip area, compared to standard microfluidic channels that extend 0.1mm in width [17]. This thesis provides pressure sharing as an optional feature by grouping the valves that can share the same pressure source, with the goal of minimizing

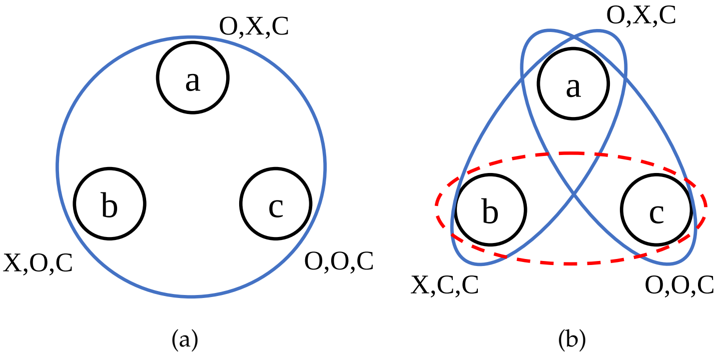


Figure 3.2: Clique examples. The blue circle indicates a valid clique. The red circle indicates an invalid clique. The status sequence is aside each valve node, with *O* indicating the status open, *C* indicating the status closed, and *X* indicating the status *don't care*. (a) The three valves can be covered by the same clique. (b) The valve *a* can construct a clique either with valve *b* or valve *c*, however valves *b* and *c* cannot be covered by the same clique.

the number of such valve groups. Two valves are said to be able to share pressure when one's state schedule do not affect the other one. Besides open and closed, a valve status X is introduced to indicate the status *don't care*. A valve is at status X when its status is not affecting the microfluidic device at that time, which can be compatible to other either open valves or closed ones [18].

To solve the pressure sharing problem, the valves are regarded as the vertices, and the edges are used to connect the valves that can share a pressure sequence. Then a graph model is built to model the pressure sharing relations of all valves. The pressure sharing problem is transferred to a clique covering problem which searches the minimum number of cliques that cover all valves. If a group of valves can share their pressure source, they can be considered to make up a clique. For example, as drawn in Figure 3.2(a), status sequence of valve *a* is open, X, closed. Status sequence of valve *b* is X, open, closed. And status sequence of valve *c* is open, open, closed. The first and the second status of these valves is either open or X. The third status of each valve is the same to be closed. This means, they can be controlled by the same pressure source open, open, closed. A clique can thus be constructed to cover all the three valves. The other example in Figure 3.2(b) presents the situation that the valve *a* can form a clique either with valve *b* or valve *c*, but valves *b* and *c* cannot be covered by the same clique. In this case, all three valves should be covered by at least two cliques. For instance, one clique to cover valves *a* and *b*, the other to cover valve *c*. At the end, the number of cliques is the number of control inlets that should be integrated. The modeling of this problem is established on the solution of the previous switch model.

To model this problem, this thesis constrains each valve to belong to exactly one clique in constraint (3.14). A binary variable $z_{v,c}$ is introduced to indicate whether valve v belongs to clique c, for all c in $Cliques$, and for all v in $Valves$, where $Cliques$ is the set of clique indices with the initial size to be the number of valves. $Valves$ is the set of all essential valves. Another binary variable $clique_c$ is introduced to denote whether the clique c is occupied. If any of $z_{v,c}$ is 1, $clique_c$ is forced to be 1.

$$\sum_{c \in Cliques} z_{v,c} = 1, \forall v \in Valves, \tag{3.14}$$

$$clique_c \geq z_{v,c}, \forall c \in Cliques, \forall v \in Valves. \tag{3.15}$$

All valves in the same clique should be able to share the pressure source. This means, if two valves cannot share their pressure, they are not allowed to be in the same clique. A binary constant $ps_{v1,v2}$ is introduced to indicate whether the valves $v1$ and $v2$ are able to share the pressure. The constraints are modeled as follows:

$$z_{v_1,c} + z_{v_2,c} \leq 1 + ps_{v_1,v_2}, \tag{3.16}$$

$\forall c \in Cliques, \forall v_1, v_2 \in Valves, v_1 \neq v_2$. When valves v_1 and v_2 are able to share their pressure, $ps_{v1,v2}$ is equal to 1. The right-hand side of constraint (3.16) becomes 2. In this case, they are allowed to belong to the same clique. The left-hand side can be either 0, 1 or 2, which holds the inequality. Otherwise, if $ps_{v1,v2} = 0$, the valves should not belong to the same clique. Variables $z_{v_1,c}$ and $z_{v_2,c}$ cannot be 1 with the same clique c. The left-hand side is thus forced to be either 0 or 1. Last, the optimization goal of this problem is to minimize the number of occupied cliques.

$$Minimize \sum_{c \in Cliques} clique_c. \tag{3.17}$$

4 Result

The method of this thesis is implemented in C++ on a computer with 900 MHz CPU. The proposed mathematical model is an integer quadratic programming (IQP) model, and is solved by Gurobi [19], an optimization solver.This thesis synthesizes the switches in widely used applications as test cases: kinase activity [20], single cell mRNA isolation [4], chromatin immunoprecipitation (ChIP) [5] and nucleic acid processor [6]. The switch inputs are as defined by Columba [21], which can be accessed online [22]. The following tests take the optimization weight of number of flow sets $\alpha = 1$ and weight of flow channel length $\beta = 100$. The three different binding policies are tested on each switch case.

4.1 Contamination Avoidance

To investigate that the synthesis results in a contamination-free switch design, this thesis first tests the switches that contain conflicting flows. Table 4.1 presents the input features: number of connected modules #*m*, the size of the switch model sw. size, as well as the result feature values : the program runtime *T* in second, total flow channel length *L* in *mm*, the number of valves #*v*, and the number of flow sets #*s*. Figure 4.1 shows the synthesized application-customized switch structures for the switch in ChIP with different binding policies: Figure 4.1(a) with fixed binding, Figure 4.1(b) with clockwise binding and Figure 4.1(c) with unfixed binding. Figure 4.2(a) shows the result for the switch case of nucleic acid processor. Figure 4.2(b) shows the result for the switch case of mRNA. By contrast, Figure4.1(d), Figure 4.2(c) and Figure 4.2(d) present the corresponding switch (marked and amplified with red rectangles) synthesized in another synthesis tool Columba [8][12][13].

As shown in Table 4.1, the switch of ChIP can be synthesized within all three binding policies, while the switch of the other two cases are only synthesized within the unfixed policy. This can be explained by the fact that the solution spaces of fixed and clockwise binding policies are much smaller. Flow conflicting constraints may let the solver fail to find solutions. Although applying the unfixed policy is able to handle all these test cases, it takes

Table 4.1: Feature results of test cases with contamination avoidance

id	application	$\#m$	sw. size	binding	$T\,(s)$	$L\,(mm)$	$\#v$	$\#s$
1	ChIP[5]	9	12-pin	clockwise	48.413	13.6	6	2
				fixed	0.273	16.4	6	2
				unfixed	8336.38	13.6	6	2
2	nucleic acid processor[6]	7	8-pin	clockwise	no solution			
				fixed	no solution			
				unfixed	7.083	9.8	6	2
3	mRNA isolation[4]	10	12-pin	clockwise	no solution			
				fixed	no solution			
				unfixed	13448.9	17.8	8	2

$\#m$: number of connected modules; sw.: switch; T: runtime in second; L: flow channel length in mm; $\#v$: number of valves; $\#s$: number of flow sets.

much more time for optimization with a larger problem space. It is worth mentioning that the unfixed binding policy needs the cooperation of the physical design tool to place connected modules according to the pin assignment results for flows.

Figure 4.1 (a), (b) and (c) show the synthesized switch structures on the flow layer. All green lines are flows of the same flow set, and the yellow lines are of the other flow set. The rectangles located across the flow channels are the essential valves. Likewise, the valves in the same color are able to share the same pressure source. The first test case is a switch of ChIP. It has conflicts between flows coming from flow inlets i_10 and i_11. The flow from i_10 is routed to Mixer $M2$, while the flows from i_11 are distributed to Mixers $M3, M4$ and $M5$. As Figure 4.1(b) displays, these flows, which are drawn as lines in yellow, are separated by the channel segment $S2$-$S3$ with its valve v_{33} closed. The result successfully avoids the potential contamination. On the contrary, a spine-like switch in this case is polluted at the junctions and segments of the spine. As presented in Figure 4.1(d), the fluid flows from i_10 and i_11 intersect at the horizontal spine. In addition, since there are no valves except at the ends along the spine, the flow from i_10 to $M2$, for example, may flow to $M3$ if the flow from i_11 to $M3$ is routed simultaneously. Besides, it deploys 9 valves, which do not share pressure with each other, quadrupling the number of control inlets compared to this design.

The second test case is a switch of nucleic acid processor. The mixture from each mixer should be sent to a dedicated reaction chamber. If any mixtures pollute each other, the single-cell experiment run on the nucleic acid processor is a failure. The tests generate a result with unfixed binding policy. The result design is shown in Figure 4.2(a). The conflicting flows from different mixers to their destination reaction chambers are drawn in yellow lines. The design separates apart the flows from *M*1 to *RC*1, from *M*2 to *RC*2 and from *M*3 to *RC*3, which achieves the goal of contamination avoidance. On the contrary, the switch design of Columba 2.0 in Figure 4.2(c) suffers from relatively heavy contamination among the spine. The spine segment marked red is the most polluted, as it is used by every flow from the mixers. Since the flows cannot avoid passing through this segment, the contamination occurs.

The third test case is a switch of mRNA isolation. Similar to the second test case, *RC*1, *RC*2, *RC*3, and *RC*4 send fluids to their dedicated fluid outlets, *p_c*1, *p_c*2, *p_c*3, and *p_c*4, respectively. As shown in Figure 4.2(b), the synthesized switch result routes these flows apart, thus avoiding the pollution. This time, the switch design of Columba S shown in Figure 4.2(d) successfully arranges the flows apart. However, similar to the first test case, since there is no valve along the spine, the fluids cannot be controlled to flow to the correct destinations if some flows are routed in parallel. For example, if the flow from *RC*1 to *p_c*1 and the flow from *RC*2 to *p_c*2 are routed in parallel, some of the fluids from *RC*1 may go to *p_c*2, polluting the fluids collected from *RC*2.

The switch is also able to be synthesized in a scalable version that is compatible to Columba S. The synthesized scalable switches for ChIP with different binding policies are presented in Figure 4.3.

4.2 Flow Scheduling

Another goal of this work is to schedule the input flows into minimum number of flow sets. Since the above referred experiments do not have many flows to be scheduled, 90 artificial switch input cases have been tested, with different input features: switch size, number of flows, number of connected modules, number of conflicting constraints, number of initial sets of flows, and binding policies.

This thesis takes the following settings as default. First, each flow pin of the switch can be bound to at most one module. The module should be either the inlet or the outlet to the switch. For example, if module *A* is connected to the pin *T*1 of the switch as an inlet, no flows can be routed to the pin *T*1. Second, each outlet pin can be accessed at most once. For example, it is not allowed to execute flows *T*1 to *T*2 and *R*1 to *T*2 in the same switch case, where *T*2 is the outlet pin that tries to be accessed twice.

Table 4.2: Input and output features of the example case

input flows	$1 \rightarrow (7, 10, 11), 2 \rightarrow (5, 8, 9), 3 \rightarrow (4, 6, 12)$
connected module order	1,2,3,4,5,6,7,8,9,10,11,12
conflicting flows	none
switch size	12-pin
binding policy	clockwise
scheduled flows	$[3 \rightarrow (4, 5, 6)], [2 \rightarrow (5, 8, 9)], [1 \rightarrow (7, 10, 11)]$
#flow sets	3
#valves	15
$L\,(mm)$	21.2

All of the test cases achieve the goal of flow scheduling. It is observed from the statistics that applying the clockwise or fixed policy of module-pin binding cannot handle some of the cases that are with contamination constraints, while the unfixed policy can always synthesize a solution to schedule all flows into one set. Another finding is that, for the same test case but tested on both 8-pin and 12-pin switches, the case on the smaller size performs better regarding the program runtime and flow channel length. But choosing the 8-pin or the 12-pin switch as the starting point has almost no impact on the scheduling performance.

Here, one of the cases is illustrated as example. The input and output features are as shown in Table 4.2. The synthesized switch structure and the flow paths on the switch are shown in Figure 4.4. This example takes all flows non-scheduled as input, and results in 3 sets of flows after synthesis. Flows of the same flow set are drawn in the same color. The flows from input 3 to outputs 4, 6 and 12 can be scheduled together (colored in yellow). It is the same for the flows from input 1 to outputs 7, 10, 11 (colored in blue), and the flows from input 2 to outputs 5, 8, 9 (colored in green). The switch schedules its input flows so that a flow channel segment cannot be occupied by flows from different inlets at the same time.

4.3 Binding Policies

To compare the difference between binding policies, further tests are conducted. The result features are as illustrated in Table 4.3, which shows the runtime T in seconds and the total flow channel length L in mm. These cases have no constraints on conflicting flows, and therefore, they have solutions for all binding policies. Regarding the flow channel length,

Table 4.3: Result features using different binding policies

id	application	$\#m$	sw. size	binding	$T\,(s)$	$L\,(mm)$
1	ChIP sw.1 [5]	9	12-pin	clockwise	48.413	13.6
				fixed	0.273	16.4
				unfixed	8336.38	13.6
2	ChIP sw.2 [5]	10	12-pin	clockwise	209.919	154
				fixed	0.513	180
				unfixed	3282.29	154
3	kinase activity sw.1 [20]	4	12-pin	clockwise	0.927	46
				fixed	1.867	46
				unfixed	21.85	46
4	kinase activity sw.2 [20]	6	12-pin	clockwise	2.197	60
				fixed	0.05	60
				unfixed	8.3	60

$\#m$: number of modules; sw.: switch; T: runtime in second; L: flow channel length in mm; $\#v$: number of valves; $\#s$: number of flow sets.

fixed binding policy in general results in the largest flow channel length L in almost all cases. This results from the smaller routing flexibility compared with the clockwise or unfixed binding policy. It is also noticeable that the fixed binding policy takes much less program runtime since the optimization solver does not need to spend effort on binding. Additionally, it is observed that for each test case, the runtime T becomes larger as the number of connected modules increases, which means, the program runtime of the switch designs increases with the complexity of the input application.

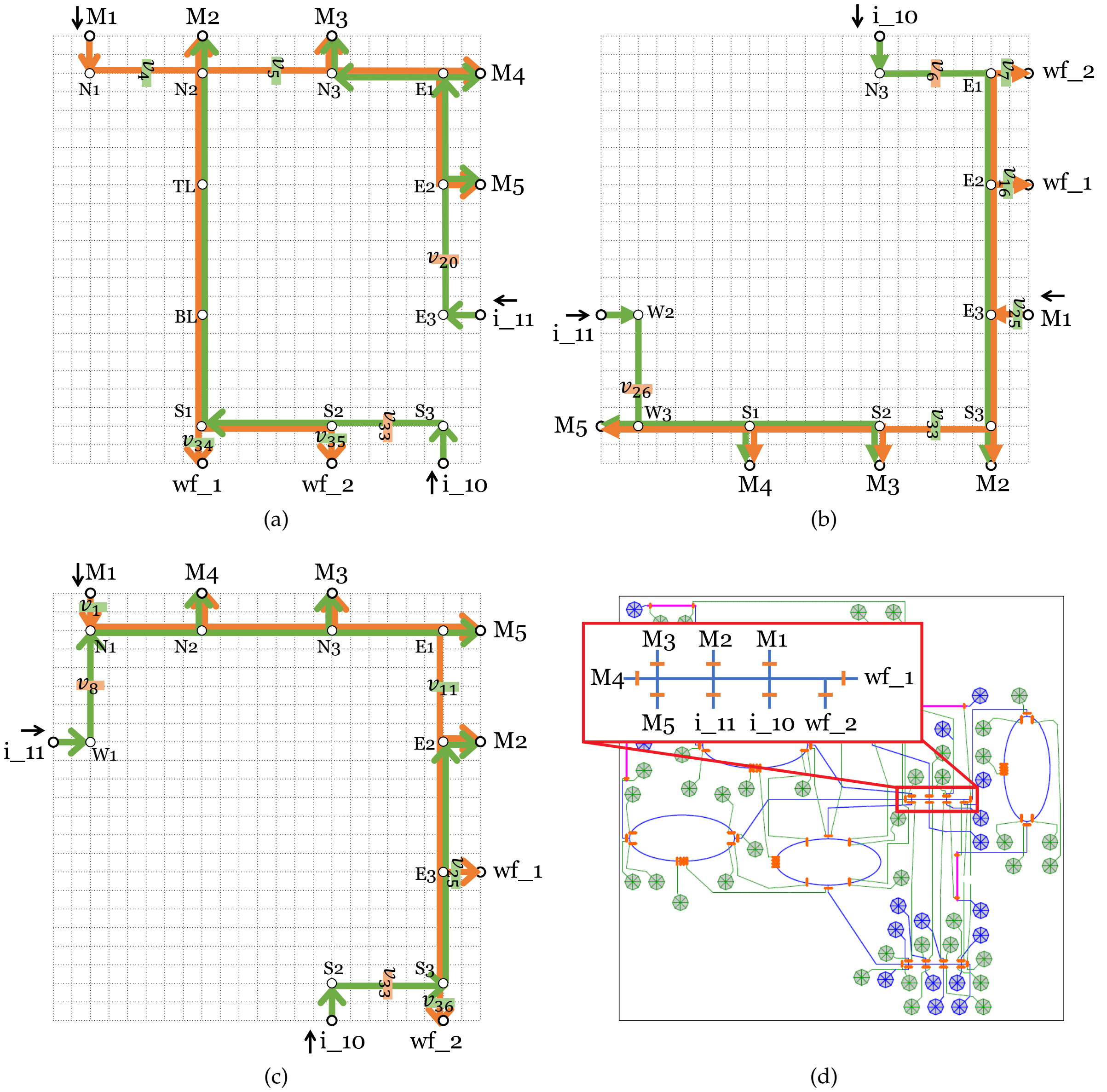


Figure 4.1: Synthesized switch results (in the flow layer) for ChIP[5]: (a) by the present thesis with fixed binding policy. (b) by the present thesis with clockwise binding policy. (c) by the present thesis with unfixed binding policy. (d) by Columba [8].

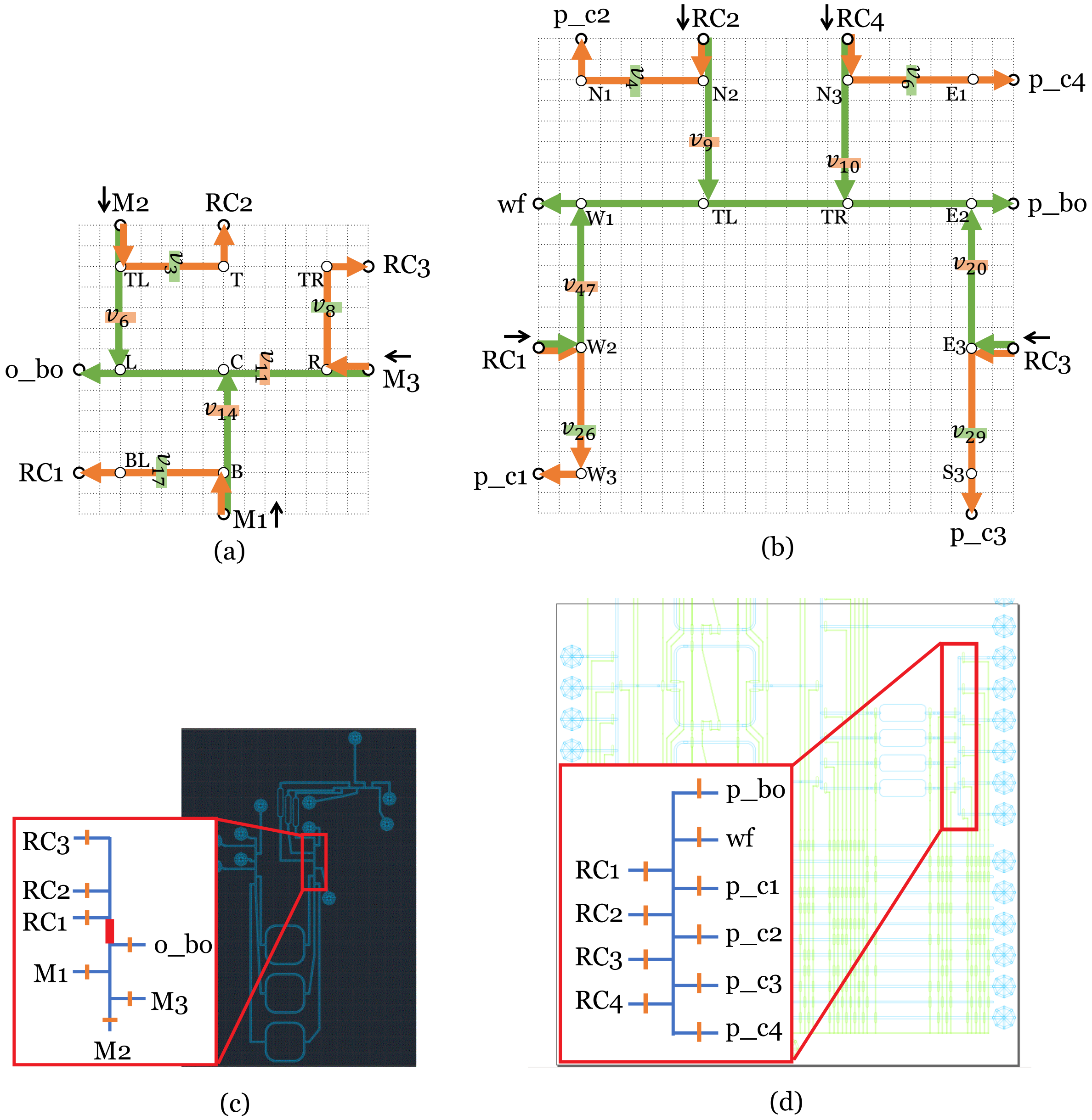


Figure 4.2: Switch designs of: (a) nucleic acid processor.(b) mRNA isolation. (c) nucleic acid processor synthesized by Columba 2.0 [12]. (d) mRNA isolation synthesized by Columba S [13].

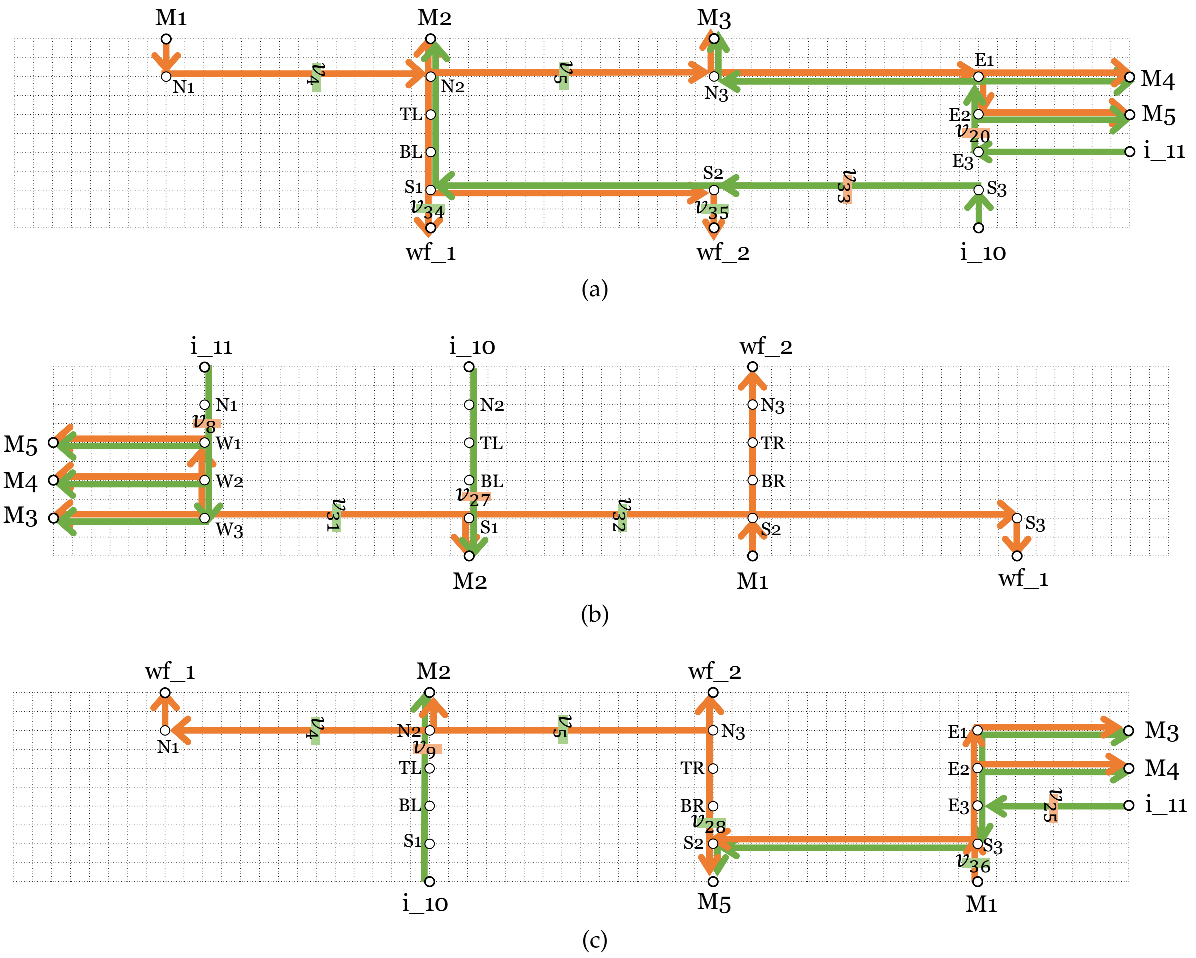


Figure 4.3: Synthesized scalable switch result (in the flow layer) for ChIP by the present thesis: (a) with fixed binding policy. (b) with clockwise binding policy. (c) with unfixed binding policy.

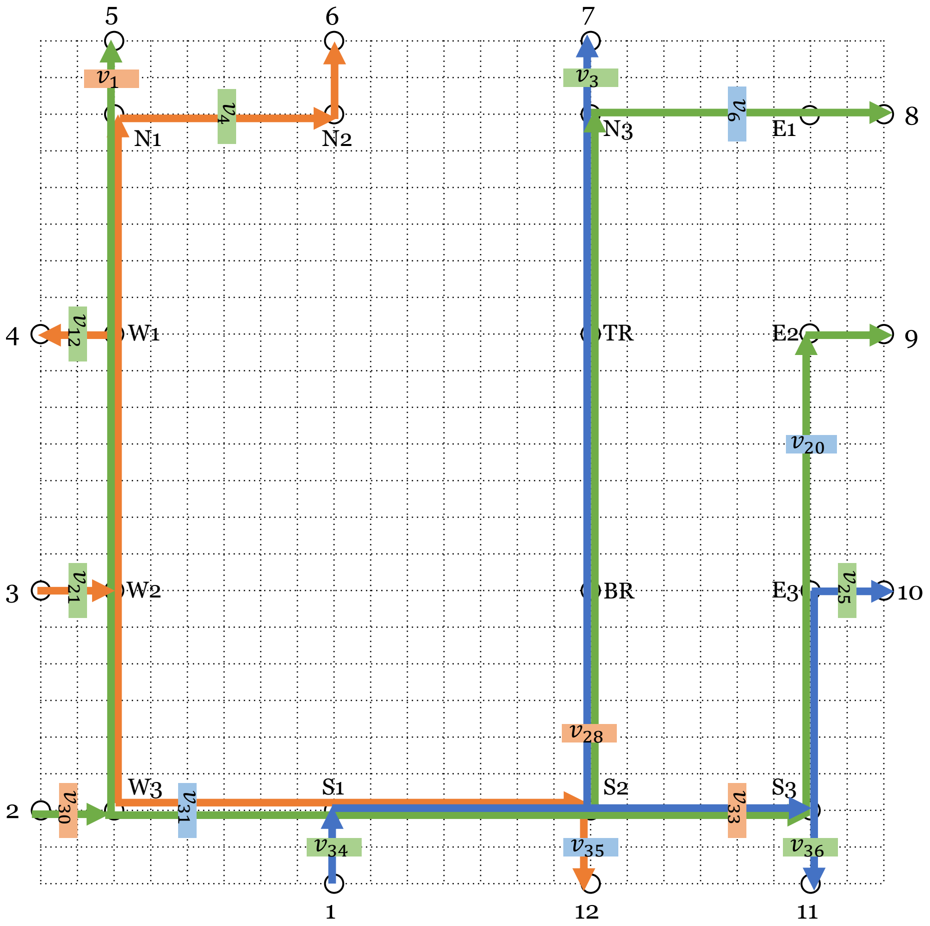


Figure 4.4: Switch structure and flow paths of the example case described in Table 4.2. (presented in the flow layer)

5 Discussion

Prior studies either do not consider the fluid contamination problem [8], or do not provide enough routing space to avoid fluid contamination or collision [14]. Since the switch is among the most polluted components in a microfluidic biochip, this thesis sets out to design a microfluidic switch that is able to avoid contaminations.

The results of this thesis indicate that the synthesized switch is always able to route the conflicting flows apart and thus ensuring the avoidance of contamination. Moreover, the implementation of flow scheduling is able to arrange the flows to be executed so that collision can be avoided. The objective to minimize the number of flow set aims to minimize the control effort as well as the flow transmission time. A smaller number of flow set indicates less changing of valve status, and thus decreased controlling effort for the control inlets. Further, the results of different binding policies indicate that the clockwise binding policy performs in general the best if the locations of the connected modules are relatively fixed around the switch. The synthesis applying the clockwise binding policy takes relatively short runtime, as well as small flow channel lengths. The fixed policy may be more desirable when the locations of the connected modules are fixed, or the synthesis expects a minimum program runtime. On the other hand, an unfixed policy is preferred when the other two methods cannot solve the model due to the constraints of conflicting flows.

Nevertheless, A number of limitations need to be noted regarding the present thesis. First, the proposed switch models cannot generate solutions for some cases owing to the strict constraints of contamination prevention when the clockwise or fixed binding policy is applied. Second, this thesis fails to solve complex cases on the 16-pin switch. The program runtime exceeds 5 hours for the 13-module input case in mRNA [4]. Besides, the current switch occupies more chip area than a traditional spine-like structure. Furthermore, although the tested cases can always be tackled using the unfixed binding policy, it takes much program runtime and may require relocation afterwards. Last, the synthesis process does not include control channel routing. Although each valve can be accessed by at least two control channels, and the unused valves are recognized during the synthesis, control channel routing should be considered for pressure sharing.

6 Conclusion

The microfluidic switch is among the most likely modules to be polluted in a continuous-flow microfluidic biochip. The purpose of this thesis is to synthesize switch designs that avoid fluid contamination with efficient flow scheduling, multiple module-pin binding policies and pressure sharing among valves. This thesis has designed reconfigurable general structures of the 8-pin, 12-pin and 16-pin switch models. And then it has built an IQP model to solve the synthesis problem. The results show that the switch designs synthesized by the proposed method are always contamination-free. This is the first attempt to investigate methods of avoiding the fluid contamination in automated synthesis tools for mLSI.

For the future work, new switch structures can be developed to be more flexible and efficient. In addition, I recommend to enhance the efficiency of the synthesis tool.

List of Figures

List of Tables

Glossary

clique is a subset of vertices of an undirected graph such that its induced subgraph is complete; that is, every two distinct vertices in the clique are adjacent. 18, 19, 31

continuous-flow microfluidics is the technology of microfluidic chips containing a large number of micro-mechanical valves and control components, enabling hundreds of assays to be automatically performed on a single miniaturized chip. 2, 4

microfluidics is the technology that precisely manipulates reagent fluids at a nanoliter-scale or below. 1

mLSI microfluidic Large-Scale Integration. 1, 2, 30

Acronyms

ChIP chromatin immunoprecipitation. 1

PCR polymerase chain reaction. 1

Bibliography

[1] M. A. Unger, H. P. Chou, T. Thorsen, A. Scherer, and S. R. Quake. “Monolithic microfabricated valves and pumps by multilayer soft lithography”. In: *Science* 288.5463 (Apr. 2000), pp. 113–116. DOI: 10.1126/science.288.5463.113.

[2] T. Thorsen, S. Maerkl, and S. Quake. “Microfluidic Large-Scale Integration”. In: *Science (New York, N.Y.)* 298 (Nov. 2002), pp. 580–4. DOI: 10.1126/science.1076996.

[3] J. Liu, M. Enzelberger, and S. Quake. “A nanoliter rotary device for polymerase chain reaction”. In: *Electrophoresis* 23 (May 2002), pp. 1531–6. DOI: 10.1002/1522-2683(200205)23:10<1531::AID-ELPS1531>3.0.CO;2-D.

[4] J. Marcus, W. Anderson, and S. Quake. “Microfluidic Single-Cell mRNA Isolation and Analysis”. In: *Analytical chemistry* 78 (June 2006), pp. 3084–9. DOI: 10.1021/ac0519460.

[5] A. Wu, J. Hiatt, R. Lu, J. Attema, N. Lobo, I. Weissman, M. Clarke, and S. Quake. “Automated microfluidic chromatin immunoprecipitation from 2,000 cells”. In: *Lab on a chip* 9 (June 2009), pp. 1365–70. DOI: 10.1039/b819648f.

[6] D.-W. Cho, V. Studer, G. Hang, W. Anderson, and S. Quake. “A nanoliter-scale nucleic acid processor with parallel architecture”. In: *Nature Biotechnology* 22 (Apr. 2004). DOI: 10.1038/nbt951.

[7] Q. Wang, H. Zou, H. Yao, T.-Y. Ho, R. Wille, and Y. Cai. “Physical Co-Design of Flow and Control Layers for Flow-Based Microfluidic Biochips”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* PP (Aug. 2017), pp. 1–1. DOI: 10.1109/TCAD.2017.2748003.

[8] T.-M. Tseng, M. Li, B. Li, T.-Y. Ho, and U. Schlichtmann. “Columba: co-layout synthesis for continuous-flow microfluidic biochips”. In: June 2016, pp. 1–6. DOI: 10.1145/2897937.2897997.

[9] K. Hu, T.-Y. Ho, and K. Chakrabarty. “Wash optimization for cross-contamination removal in flow-based microfluidic biochips”. In: *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC* (Jan. 2014), pp. 244–249. DOI: 10.1109/ASPDAC.2014.6742897.

[10] K. Dorfman, M. Chabert, J.-H. Codarbox, G. Rousseau, P. Cremoux, and J.-L. Viovy. “Contamination-Free Continuous Flow Microfluidic Polymerase Chain Reaction for Quantitative and Clinical Applications”. In: *Analytical chemistry* 77 (July 2005), pp. 3700–4. DOI: 10.1021/ac050031i.

[11] F. Su, K. Chakrabarty, and R. Fair. “Microfluidics-Based Biochips: Technology Issues, Implementation Platforms, and Design-Automation Challenges”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.2 (2006), pp. 211–223. DOI: `10.1109/TCAD.2005.855956`.

[12] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann. “Columba 2.0: A Co-Layout Synthesis Tool for Continuous-Flow Microfluidic Biochips”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.8 (2018), pp. 1588–1601. DOI: `10.1109/TCAD.2017.2760628`.

[13] 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”. In: *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. 2018, pp. 1–6. DOI: `10.1109/DAC.2018.8465905`.

[14] Y. Ma. *Switch Design for Microfluidic Large-Scale Integration.* `https://www.ei.tum.de/en/eda/theses-jobs/finished-theses/,https://edawww.regent.e-technik.tu-muenchen.de/public/upload/202010021545_MasterThesis_YanluMa.pdf`. Accessed on 2021-05-31.

[15] T. Yan and M. D. F. Wong. “A Correct Network Flow Model for Escape Routing”. In: *Proceedings of the 46th Annual Design Automation Conference* (July 2009), pp. 332–335. DOI: `https://doi.org/10.1145/1629911.1630001`.

[16] A. Truppel, T.-M. Tseng, D. Bertozzi, J. Alves, and U. Schlichtmann. “PSION: Combining Logical Topology and Physical Layout Optimization for Wavelength-Routed ONoCs”. In: *Proceedings of the 2019 International Symposium on Physical Design* (2019).

[17] Stanford Foundry. *Basic Design Rules*. `http://web.stanford.edu/group/foundry`. Accessed on 2021-05-31.

[18] H. Yao, T.-Y. Ho, and Y. Cai. “PACOR: Practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips”. In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2015, pp. 1–6. DOI: `10.1145/2744769.2744887`.

[19] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*. `http://www.gurobi.com`. Accessed on 2021-05-31.

[20] C. Fang, Y. Wang, N. Vu, W.-Y. Lin, Y.-T. Hsieh, L. Rubbi, M. Phelps, M. Müschen, Y.-M. Kim, A. Chatziioannou, H.-R. Tseng, and T. Graeber. “Integrated Microfluidic and Imaging Platform for a Kinase Activity Radioassay to Analyze Minute Patient Cancer Samples”. In: *Cancer research* 70 (Nov. 2010), pp. 8299–308. DOI: `10.1158/0008-5472.CAN-10-0851`.

[21] T.-M. Tseng, M. Li, Y. Zhang, T.-Y. Ho, and U. Schlichtmann. “Cloud Columba: Accessible Design Automation Platform for Production and Inspiration: Invited Paper”. In: *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2019, pp. 1–6. DOI: `10.1109/ICCAD45719.2019.8942104`.

[22] Cloud Columba Team. *Cloud Columba*. `https://cc1.cloud-columba.org/`. Accessed on 2021-05-31.