A Network-Flow Based Valve-Switching Aware Binding Algorithm for Flow-Based Microfluidic Biochips

Kai-Han Tseng[1] Sheng-Chi You[1] Wajid Hassan Minhass[2] Tsung-Yi Ho[1] Paul Pop[2]
[1]National Cheng Kung University, Taiwan
[2]Technical University of Denmark, Denmark

Abstract— Designs of flow-based microfluidic biochips are receiving much attention recently because they replace conventional biological automation paradigm and are able to integrate different biochemical analysis functions on a chip. However, as the design complexity increases, a flow-based microfluidic biochip needs more chip-integrated micro-valves, i.e., the basic unit of fluid-handling functionality, to manipulate the fluid flow for biochemical applications. Moreover, frequent switching of micro-valves results in decreased reliability. To minimize the valve-switching activities, we develop a network-flow based resource binding algorithm based on breadth-first search (BFS) and minimum cost maximum flow (MCMF) in architectural-level synthesis. The experimental results show that our methodology not only makes significant reduction of valve-switching activities but also diminishes the application completion time for both real-life applications and a set of synthetic benchmarks.

I. INTRODUCTION

Recently, microfluidics-based biochips have emerged as a popular alternative for laboratory experiments. Unlike conventional biochemical analyzers, which consist of fluid-handling robots, microfluidic biochips integrate necessary biochemical functionalities (e.g., mixers, filter, dispensers, detectors) on a chip [5], offering a number of advantages such as high portability, high throughput, high sensitivity, less human intervention, and low sample volume consumption [2]. These composite microsystems are also known as lab-on-a-chip, replacing cumbersome equipments to miniaturized and integrated systems [9].

Nowadays, the mainstream for microfluidic system is flow-based microfluidic biochip due to its easy implementation and less sensitive to fouling problems [4]. The fluid flow on flow-based microfluidic biochips were manipulated continuously through the pre-defined micro-channels using micro-valves and micro-pumps [10]. Since the valves are the basic unit of fluid-handling functionalities, by combining several chip-integrated valves, more functional components such as mixer, reactor, separator, detector, filter can be built and the flow of liquid can be successfully controlled to accomplish the biochemical applications [3].

However, as the requirements and the design complexity rapidly increase, the manufacture and the biochemical analysis of flow-based microfluidic biochip become more complicated. According to recent study [8], the biochips can now use more than 25,000 valves and about a million features to run 9,216 parallel polymerase chain reactions. Moreover, the number of mechanical valves per square inch for flow-based microfluidic biochips has grown exponentially and four times faster than the reflection of Moore's Law [1].

In addition, to the best of our knowledge, recent works for flow-based microfluidic biochips ignore the issue of valve-switching. Because the valves play the most fundamental role of flow-based microfluidic biochip, a biochemical application running on an inappropriately synthesized biochip would result in dispensable switching of the valves. It should be averted to have the redundant activities because a valve operates reliably only for a few thousands of actuations [7]. The valve switching also consumes energy, which may be an issue for future portable, battery powered, systems. Therefore, a well-designed resource binding algorithm should be developed while synthesizing the biochip to avoid this problem.

In this paper, we propose a network-flow based valve-switching aware binding algorithm for flow-based microfluidic biochips. Our contribution can be summarized as follows:

- We model the valve-switching procedures for functional components such as mixer, and identify how to reduce the valve-switching amounts by binding the continuous operations to the same component.
- We develop a set-based minimum cost maximum flow (SMCMF) resource binding algorithm by using breadth-first search (BFS) technique to group continuous operations together to diminish dispensable valve-switching and applying minimum cost maximum flow (MCMF) to minimize the application complete time and enhance the component utilization.

Experimental results show that our network-flow based algorithm not only makes significant reduction of valve-switching activities but also diminishes the application completion time for both real-life applications and a set of synthetic benchmarks.

The remainder of this paper is organized as follows: Section II introduces the system model of the flow-based microfluidic biochips and the manipulations of the valves. Section III formulates the biochip synthesis problem. Section IV introduces our synthesis methodology. Section V - VI show our experimental results and conclusions.

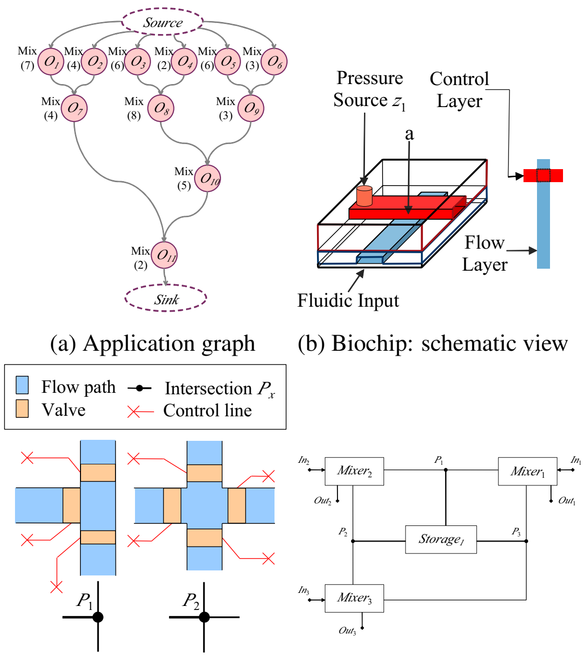
II. PRELIMINARY

A. Biochemical Application Model

We model a biochemical application using a sequencing graph. The graph is directed, acyclic with a source vertex whose in-degree is zero and a sink vertex whose out-degree is zero. Other vertexes represent operations. Each directed edge from vertex O_i to O_j means that the sample will transport from O_i to O_j. Fig. 1(a) shows an example of a biochemical application containing 11 mixing operations, the execution time and the component type of each operation is shown in its left side.

B. Micro-Valve Model

The basic unit of a flow-based microfluidic biochip is a valve. By actuating the valves, we can successfully manipulate the fluid flow on the biochip. In Fig. 1(b), the control layer (red) is connected to an external air pressure source z_1. The flow layer (blue) is connected to a fluid reservoir through a pump that generates the fluid flow. Without the activation of pressure source, the fluid can flow through freely. Otherwise, high pressure through the control layer pinches the underlying flow layer (i.e., a in Fig. 1(b)) and blocks the fluid flow. A flow-based microfluidic biochip might allocate thousands of valves. By combining these valves, more complex units, such as switches, multiplexers, micro-pumps, mixers, can be built. Moreover, the valves can also use to control the fluid flow on the biochip. For example, Fig. 1(c) shows two cases that flow-channels intersect with each other. 3 and 4 valves are required for the intersection P_1 and P_2, respectively. Valves on flow paths can control the movement of fluid by switching these valves open or closed.



(a) Application graph (b) Biochip: schematic view

(c) Flow control with valves (d) Simple architecture

Fig. 1. Elements of microfluidic biochip

C. Biochip Architecture Model

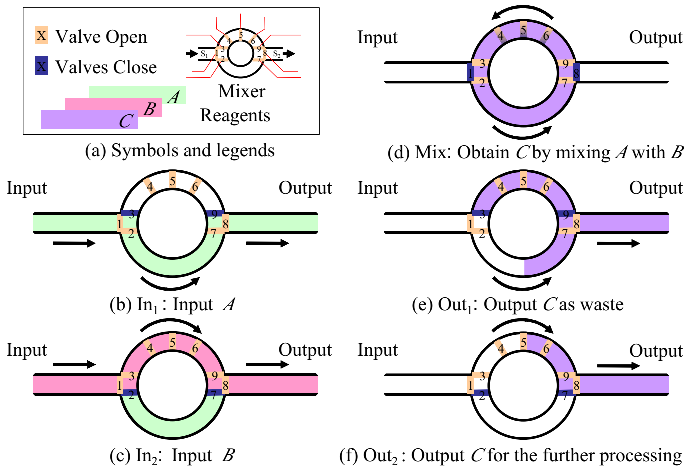
The flow-based microfluidic biochip is manufactured using multilayer soft lithography [5]. Basically, the biochip can contain multiple layers. These layers can be logically divided into two types: the flow layer and the control layer. In this paper, we will focus on the discussion of flow layer.

The biochip architecture can be seen as a set of vertex, edge and flow path. Note that there are tow types of vertex: intersection (e.g., P_1 in Fig. 1(d)), component or an input/output node (e.g., $Mixer_1$ and In_1, respectively, in Fig. 1(d)). The flow path is the set of permissible flow routes on the biochip. A directed edge represents a directed transportation from a vertex to another vertex. The transportation function can be defined as $c(x)$ representing the transportation time for the distance x, which is the length of a flow path. Note that a flow path is either a single directed edge or a subset of two or more directed edges. For example, in Fig. 1(d), a flow path can represent a directed link from vertex $Mixer_2$ to vertex $Storage_1$.

D. Component Model

The general purpose microfluidic processor contains primitive elements, including storage chambers, input and output ports (I/O), a mixer, and an interconnection network that enables to build a path for samples [12]. In this section, we will model the switching step of a mixer.

In Fig. 2, considering the pneumatic mixer, it contains nine valves. Each valve has its control line connecting with the control layer (the red line in Fig. 2(a)). The valve set v_1, v_2, v_3 is the input port S_1 and the valve set v_7, v_8, v_9 is the output port S_2. The valve set v_4, v_5, v_6 acts as an on-chip pump. The two ports facilitate the inputs and outputs. The mixing output can either be sent to the waste reservoir or to the other components for further processing.



(a) Symbols and legends (d) Mix: Obtain C by mixing A with B

(b) In_1 : Input A (e) Out_1: Output C as waste

(c) In_2: Input B (f) Out_2 : Output C for the further processing

Fig. 2. Illustration for a mixer operation

E. Valve-Switching Minimization

According to recent studies, a valve can only be actuated in thousands of times [7]. Because the flow-based microfluidic biochip are manipulated by the valves, more valve-switching means reduced reliability.

TABLE I: Valve-Switching Phase of a Mixer

Phase	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
In_1	0	0	1	0	0	0	0	0	1
In_2	0	1	0	0	0	0	1	0	0
Mix	1	0	0	Mix	Mix	Mix	0	1	0
Out_1	0	0	1	0	0	0	0	0	1
Out_2	0	1	0	0	0	0	1	0	0

In order to minimize the valve-switching amounts, in this paper, we take mixer as an example to explain how to reduce the valve-switching activities by grouping the continuous operations together. In Fig. 2(a), there are three reagents A, B, and C. For reagent C, it is the mixing result of reagent A and B. As shown in Table I, a mixer will go through 5 phases to complete a mixing operation. The valve activation for each phase is also shown in Table I. Here, "0" represents an open and "1" represents a close of the valve. The status "Mix" shown for the valve set v_4, v_5, v_6 represents the mixing step in which these valves are opened and closed in a specific frequency to achieve mixing. The first two phases, which called In_1 and In_2, store reagent A and reagent B into the mixer by closing (v_3, v_9) and (v_2, v_7) respectively (Fig. 2(b) and (c)). The third phase is to mix two stored reagents into reagent C by closing (v_1, v_8) and switching in the order of (v_4, v_5, v_6) repeatedly that acts as an on-chip pump (Fig. 2(d)). The last two phases, Out_1 and Out_2, represent to output the mixed reagent C. Out_1 phase output the half of reagent C as the waste by closing (v_3, v_9), and Out_2 phases output another half of reagent C as the mixing result by closing (v_2, v_7) (Fig. 2(e) and (f))[6]. Finally, transport reagent C for the further processing.

By Table I, 20 valve-switching activities ($0 \rightarrow 1$ or $1 \rightarrow 0$) are required for one mixing operation (v_4, v_5, v_6 activation is not considered due to the different requirements of switching times in mixing phase).

F. Continuous Grouping

By reducing the Out_2 phase for two continuous mixing operations in the same component, we can diminish the valve-switching by 6 times. Considering Fig. 2(f) again, if sample C could stay in component for second mixing, the Out_2 phase will not be necessary at all. That is, we can transform Out_1 phase as In_1 phase for second mixing. Therefore, in Fig. 3, we only need 14 valve-switching activities for these two mixing operations (6 valve-switching are reduced in Out_2 phase). If we can bind more mixing operations on the same component, only 8 valve-switching activities are needed.

G. Transportation Cost

Binding more operations on the same components not only decreases total valve-switching amounts but also removes dispensable transportations. The transportations would lead to unnecessary valve-switching since that the fluid flow transport from one component to another component is manipulated by the valves. This problem might become more critical as the increasing valve numbers of biochip architecture.

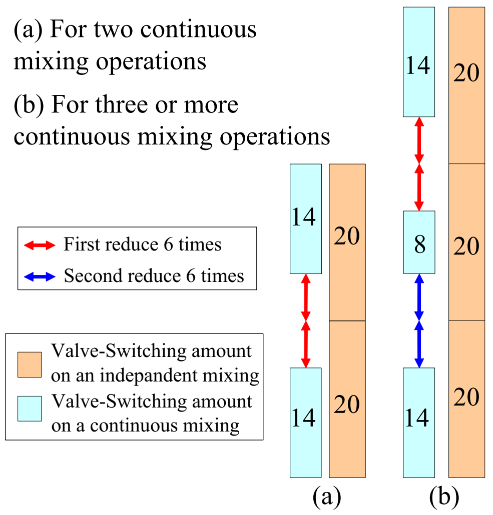

(a) For two continuous mixing operations
(b) For three or more continuous mixing operations
First reduce 6 times
Second reduce 6 times
Valve-Switching amount on an indepandent mixing
Valve-Switching amount on a continuous mixing
14
20
14
20
14
20
8
20
14
20
(a)
(b)

Fig. 3. Continuous valve-switching on a mixer

III. PROBLEM FORMULATION

Based on previous discussions, the problem addressed in this paper can be formulated as follows:

Input: *A biochemical application modelled as a sequencing graph and a component library includes different types of components.*

Component Constraint: *Component Constraint: The components used for the architecture cannot exceed the maximum allowable number in the component library.*

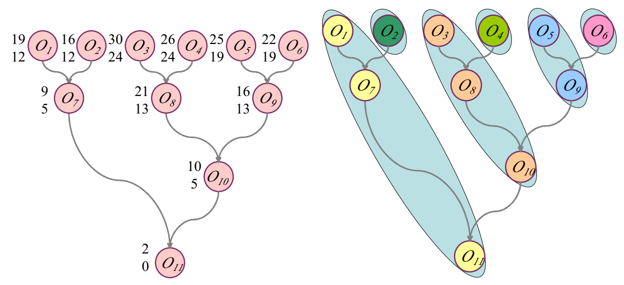
Objective: *Obtain a resource binding result under the above constraints such that the total valve-switching amounts and the application complete time for the biochemical application is minimized.*

IV. BIOCHIP SYNTHESIS METHODOLOGY

To accomplish a biochemical application and evaluate the total valve-switching amounts and application complete time, we should also generate the biochip architecture in the end of our synthesis methodology. We will first introduce our set-based minimum cost maximum flow algorithm (SMCMF) and then using a relation-based method to generate the biochip architecture in physical synthesis and adopt a list scheduling-based approach to evaluate our performance.

In this section, we will use the application graph shown in Fig. 1(a) under the component constraint of 3 mixers as an example to present our SMCMF algorithm. The basic idea of our SMCMF algorithm is to find the set in the application graph and then use these sets to build the minimum cost maximum flow network. In the flow network, each node represents a set which is also referred as some continuous operations. By using this methodology, we can not only minimize the valve-switching activities but also maximize the component utilization.

The SMCMF algorithm can be hierarchically divided into three steps. In the first step, we apply a breadth-first search (BFS) technique to count the topological order and group the continuous operations in a set, and in the second step, we will construct the sets into a flow network and apply minimum cost maximum flow algorithm to obtain the concatenated-sets. In the last step, we will make the components allocate to the high-priority concatenated-sets and split the the low-priority concatenated-sets into several operations to rearrange them into other components. Our SMCMF algorithm will be described in the following subsections.



19 12 O_1 16 12 O_2 30 24 O_3 26 24 O_4 25 19 O_5 22 19 O_6
9 5 O_7 21 13 O_8 16 13 O_9
10 5 O_{10}
2 0 O_{11}
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9 O_{10} O_{11}

(a) Topological sort (b) Depth first search

Fig. 4. Separate operation strategy

A. Separating Continuous Operation by BFS

Our SMCMF algorithm can be started by calculating the topological order of the biochemical application. As shown in Fig. 4(a), the topological sort begins with the last operation O_{11}. Because the biochip architecture is not realized yet, The transportation cost $c(x)$ should be estimated as a constant time 3. After doing topological sort, each operation receives two numbers represented the starting time and ending time. The starting time is the bigger one which means their *urgency criteria* and the other one implies the estimated finish time of the operations. This step is used to calculate the estimated time period for each operation.

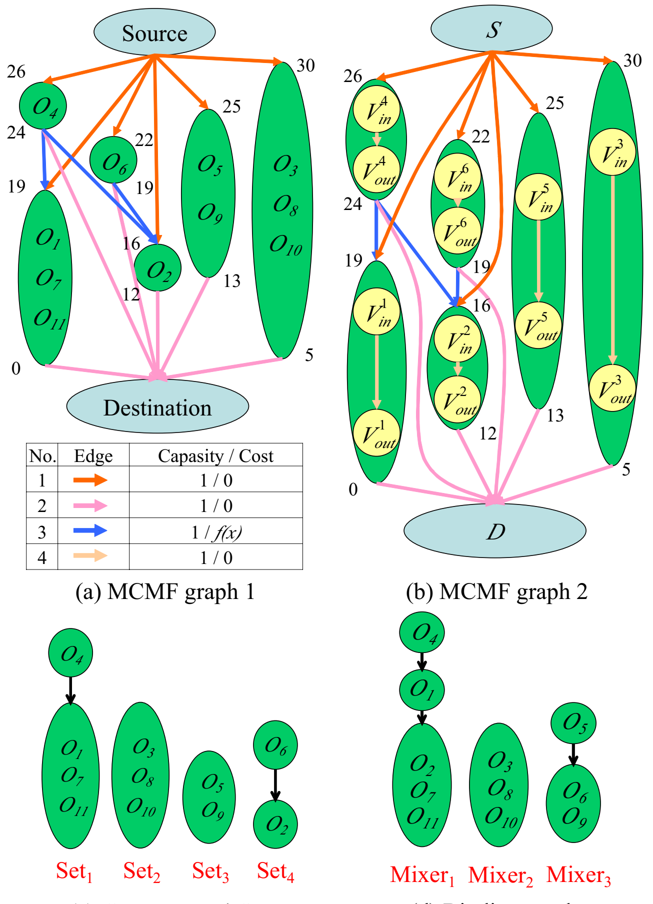
As describes in Section II, using the same component for the continuous operations will reduce valve-switching activities. In order to minimize the total valve-switching amounts, we should group the continuous operations together. Therefore, it is very obvious that we can use breadth-first search technique to achieve our goal. In Fig. 4(a), O_{11} is regarded as a source node when applying BFS. The purpose for BFS is to make the continuous operations into a set and make each set as big as possible. We can use different colors to distinguish the operations into different sets. The rules for assigning the colors for the nodes while doing BFS traversal can be listed as follows:

1. For the source node, assign an unique color for it and regard it as a new set.
2. For each non-source node, if its predecessor doesn't group with other operations, assign a color to the node the same as its predecessor and group them into the same set. Otherwise, assign a new unique color for the node and regard it as a new set.
3. While all the nodes are assigned to a color, the BFS traversal is finished.

The results for the BFS is shown in Fig. 4(b). The operations are separated into 6 sets including $s_1\{O_1, O_7, O_{11}\}$, $s_2\{O_2\}$, $s_3\{O_3, O_8, O_{10}\}$, $s_4\{O_4\}$, $s_5\{O_5, O_9\}$, $s_6\{O_6\}$. For a set s_i, it has two attributes ($_S$, $_E$), where $s_i{\cdot}_S$ is inherited from the starting time of the first operation in s_i and $s_i{\cdot}_E$ is inherited from the ending time of the last operation in s_i. Those operations grouping into a set are continuous operations.

B. Minimum Cost Maximum Flow Formulation

Though we already grouped operations into a set, the given number of mixers may still be insufficient. Nevertheless, the executing time for some sets may be disjointed. So, we need to concatenate those sets together to maximize the component utilization. To achieve our goal, we construct a minimum cost maximum flow graph $G_{mcmf} = (V_{mcmf}; E_{mcmf})$ for these sets and propose two formulation rules. The first one describes the establishment for the node V_{mcmf}, and the second one describes how to build the edge E_{mcmf}.



Source
Destination
26 24 19 0 22 19 16 12 25 13 30 5
O_4 O_6 O_5 O_9 O_3 O_8 O_{10} O_1 O_7 O_{11} O_2
S
D
V^4_{in} V^4_{out} V^6_{in} V^6_{out} V^5_{in} V^5_{out} V^3_{in} V^3_{out} V^1_{in} V^1_{out} V^2_{in} V^2_{out}
Set₁ Set₂ Set₃ Set₄
Mixer₁ Mixer₂ Mixer₃

No.	Edge	Capasity / Cost
1	→	1 / 0
2	→	1 / 0
3	→	1 / *f(x)*
4	→	1 / 0

(a) MCMF graph 1 (b) MCMF graph 2

(c) Concatenated-Set (d) Binding result

Fig. 5. Minimum cost maximum flow methodology

MCMF-Rule #1: Formulation of V_{mcmf}

1. Create a source node S, and a destination node D.
2. For each set s_i separating by BFS, create a node V^i, and then split V^i into an in-node V^i_{in} and an out-node V^i_{out}. Both V^i_{in} and V^i_{out} have two attributes ($_S$, $_E$). For $V^i_{in}{\cdot}_S$ and $V^i_{in}{\cdot}_E$, they have the same value inherited from $s_i{\cdot}_S$. For $V^i_{out}{\cdot}_S$ and $V^i_{out}{\cdot}_E$, they have the same value inherited form $s_i{\cdot}_E$

MCMF-Rule #2: Formulation of E_{mcmf}

1. For each in-node V^i_{in}, create a directed edge $S \rightarrow V^i_{in}$ with *one* unit capacity and *zero* cost per unit flow.
2. For each out-node V^i_{out}, create a directed edge $V^i_{out} \rightarrow D$ with *one* unit capacity and *zero* cost per unit flow.
3. For each node pair V^i_{in} and V^i_{out}, create a directed edge $V^i_{in} \rightarrow V^i_{out}$ with *one* unit capacity and *zero* cost per unit flow.

4. For an out-node V^i_{out} and an in-node V^j_{in} if $V^i_{out}{\cdot}E \geq V^j_{in}{\cdot}S + c(x)$, create a directed edge $V^i_{out} \rightarrow V^j_{in}$ with *one* unit capacity and *f(x)* cost per unit flow, where $c(x)$ is the transportation time and $f(x)$ is defined as the difference between $V^i_{out}{\cdot}E$ and $V^j_{in}{\cdot}S$.

Fig. 5(a) and Fig. 5(b) show the example for constructing the MCMF graph. In Fig. 5(a), the nodes are represented by the sets s_i containing operations, while in Fig. 5(b), each node in Fig. 5(a) is split into one in-node V^i_{in} and one out-node V^i_{out}. For example, the node pair (V^1_{in}, V^1_{out}) in Fig. 5(b) can be mapping to the set $s_1\{O_1, O_7, O_{11}\}$ in Fig. 5(a). The following paragraph will explicitly explain the procedures for constructing the MCMF graph.

To construct the nodes for the MCMF graph, we first create a source node S and a destination node D by MCMF-Rule 1.1, and then using MCMF-Rule 1.2 to create the other nodes for each set s_i separating by BFS. For example, there is a set $s_5\{O_5, O_9\}$ as shown in Fig. 5(a), so we initially construct a node V^5 and then split V^5 into a node pair (V^5_{in}, V^5_{out}) as shown in Fig. 5(b).

After creating the nodes, we need to construct the edges for the MCMF graph, so we apply MCMF-Rule 2. The first two rules mean that we need to create the edges from the source to all the in-nodes V^i_{in} and from all the out-nodes V^i_{out} to D. The third rule represents that there is an internal edge between each node pair V^i_{in} and V^i_{out}, as shown in Fig. 5(b). The last rule indicates that we have to build the edge between two nodes if the estimated time difference is more than the transportation function $c(x)$. For example, $V^4_{out}{\cdot}E$ is 24, $V^1_{in}{\cdot}S$ is 19 and the transportation function $c(x)$ is estimated to be 3. Because $V^4_{out}{\cdot}E \geq V^1_{in}{\cdot}S + c(x)$, we should build the edge from V^4_{out} to V^1_{in}., as shown in Fig. 5(b), there are 3 edges $V^4_{out} \rightarrow V^1_{in}$, $V^4_{out} \rightarrow V^2_{in}$ and $V^6_{out} \rightarrow V^2_{in}$ should be built by this rule.

In the end, we construct the flow network shown in Fig. 5(b) and apply MCMF algorithm to obtain flow paths. Each flow path contains node pair (V^i_{in}, V^i_{out}), by mapping the node pair back to its corresponding set, we can obtain the new concatenated-set as shown in Fig. 5(c).

C. Component Allocation

Nevertheless, the given number of mixers is still insufficient, since we have only three mixers but four concatenated-sets. The goal in this step is to allocate the finite components to the high-priority concatenated-sets and split the low-priority concatenated-sets into a single operations to rearrange it into the component before its successor. The priority for the concatenated set is defined by the continuous operation number as well as the total operation number. We will first compare the continuous operation number and adopt the total operation numbers as the tight breaker. In Fig. 5(c), Set_1 is the concatenated sets with the highest priority because it has three continuous operations and totally four operations. Though Set_2 also has the same number of continuous operations, it has only three operations totally which indicates lower priority than Set_1. After determining their priorities, we can allocate the 3 mixers to the top-3 priority sets. For the lowest priority set, which is Set_4, we split it into O_2 and O_6 and rearrange them to $Mixer_1$ and $Mixer_3$ because the next operations for O_2 is O_7 and O_6 is O_9 which are in $Mixer_1$ and $Mixer_3$, respectively.

After applying these steps, the resource binding and scheduling results can be obtained as shown in Fig. 5(d), and we can also use the information to construct a relational graph for physical synthesis. A relational graph is a undirected graph that represents the transportation for the components. In the relational graph, each node is regarded as a component, and an edge means that there are transportations between two components and the edge cost is the transportation time. For example, according to the binding result and the application graph as shown in Fig. 5(d) and Fig. 1(a), we know that O_{10} is bound to $Mixer_2$ and O_{11} is bound to $Mixer_1$. Because the successor for O_{10} is O_{11}, it means that there is a transportation from $Mixer_2$ to $Mixer_1$. By applying this rule for each edge on the application graph, we can construct a relational graph for the components.

D. Physical Synthesis

The biochip architecture can be generated in physical synthesis. It is similar to traditional VLSI design automation, the physical synthesis can be addressed as the issue of placement and routing. In this paper, we use a relation-based method to synthesize the biochip architecture. As describes in the previous subsection, we will obtain a relational graph after applying SMCMF to get the binding information. In the relational graph, the edges means the number of transportations between two components. The concept of the relation-based method is to place the components much closer if the transportation number between them are bigger. It can also be referred as placing the more related components together. Using this concept we can diminish the transportation time of the components after the routing step because the components are close to each other, which reduces the total length of the flow-channels.

E. Explicit Scheduling

After the binding information and the architecture are completed, the next step is to dispatch every operation and transportation to their corresponding components. This problem is related to resource constrained scheduling problem with non-uniform weights, which is NP-complete [11]. We adopt list scheduling-based approach to solve the problem in a computationally efficient manner [6]. It takes the biochemical application model, the flow layer models of the biochip architecture, the biochip components and binding information as input. As output, it generates a schedule for all components and transportations. This information contains the details for each component on every moment, such as flow transportation or operation. So, it can be used to generate the control sequence for executing the operation on the specified biochip. Also, we can calculate the valve-switching amounts after the list scheduling-based approach.

Besides, this approach should also consider the issue of routing latency. While doing scheduling, the transportations can't pass through one flow-channel at the same time. Once the collision of the transportations occur, they should be deferred until the flow-channel is idle. Moreover, due to the routing latency, it might need to wait for the inputs since there are two inputs

TABLE II: PCR Real-Life Applications with SMCMF Algorithm

Arc. No.	Allocated Units	Total Valves	Valve-Switching On Flow / Op.	Total Idle / Time
1	(3,3,3,0,0,1)	54	40 / 104	5.4 / 20.1
2	(4,4,4,0,0,1)	72	14 / 125	3.1 / 17.0

TABLE III: PCR Real-Life Applications with Baseline Method

Arc. No.	Allocated Units	Total Valves	Valve-Switching On Flow / Op.	Total Idle / Time
1	(3,3,3,0,0,1)	54	40 / 116	15.6 / 20.9
2	(4,4,4,0,0,1)	72	18 / 137	9.0 / 18.3

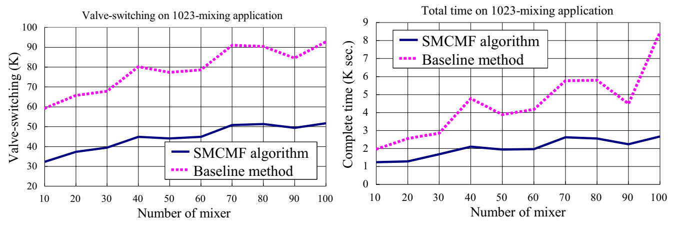
for some kinds of components. The components might receive only one input and waiting for the other one as a result of routing latency. We define this time period as component idle time which indicates the waiting time between the first input and the second input.

V. EXPERIMENTAL RESULTS

We evaluate both real-life case and a set of synthetic benchmarks by using our SMCMF algorithm for binding to determine the application complete time, component idle time and valve-switching amounts for this application and the corresponding architecture. These algorithms are implemented in C++, running on a PC with Core2 Quad processors at 2.66GHz and 3.25GB of RAM.

Table II and III show our experimental results on PCR real-life case. We implement both SMCMF algorithm and baseline method to construct the binding information for different architectures. The baseline method here simply binds according to the operations' topological orders, which are referred to the urgency criteria as describes in Section A. The bigger of the urgency criteria means the operation has the higher priority to bind the given components. Column 1 presents the number of each architecture and column 2 shows the list of allocated components, in the following format: (Input ports, Output ports, Mixers, Heaters, Filters, Storage). Columns 3 shows the total number of valves on the chip. Columns 4 presents the valve-switching amounts on flow paths and components respectively. Columns 5 shows the total component idle time and the application complete time respectively. The real-life case, polymerase chain reaction (PCR), has 7 mixing operations and is used in DNA amplification. We synthesize the assay on two different biochip architectures varying the number of I/O ports and mixer units (3 and 4) with one storage. These two table shows that our SMCMF algorithm makes better binding information and obtains less valve-switching amounts and shorter total component idle time/application complete time than the baseline method does.

The following two line charts are experimental results on synthetic benchmarks under different considerations. Fig. 6 shows the distribution of different number of mixers and one storage among valve-switching amounts and application complete time. We use a synthetic application containing 1023 mixing operations as a benchmark and two binding information made by SMCMF algorithm and baseline method. By tuning the number of mixer, we can obtain the trend for each algorithm. It shows 43% and 54% improvements on valve-switching amounts and application complete time on average. When an architecture contains more mixers, the complete time and valve-switching amounts will increase because only one storage in this architecture. However, the increasing rate of the blue line using SMCMF algorithm is smaller and more stable than the pink one using baseline method.



(a) Valve-switching (b) Application complete time

Fig. 6. The results on different number of mixer

VI. CONCLUSIONS

In this paper, we proposed a network-flow based valve-switching aware binding algorithm for flow-based microfluidic biochips. Given a biochip application graph, we use SMCMF to get the resource binding information and using a relation-based method to get the biochip architecture. And using list-scheduling based approach to evaluate the results, which shows that our synthesis algorithm not only minimizes the total valve-switching amounts but also diminishes the application complete when comparing with a baseline method for both real-life case PCR and a set of synthetic benchmarks. It means that our SMCMF algorithm is good enough to get a proper resource binding information while synthesizing a biochip.

REFERENCES

[1] J. W. Hong and S. R. Quake, "Integrated nanoliter systems," *Nature Biotechnology*, 21:1179–1183, 2003.

[2] T.-W. Huang, T.-Y. Ho, and K. Chakrabarty, "Reliability-oriented broadcast electrode-addressing for pin-constrained digital microfluidic biochips," *IEEE/ACM ICCAD*, pp. 448–455, 2011.

[3] Y. C. Lim, A. Z. Kouzani, and W. Duan, "Lab-on-a-chip: a component view," *Journal of microsystems technology*, 16(12), December 2010.

[4] K.-K. Liu, R.-G. Wu, Y.-J. Chuang, H. S. Khoo, S.-H. Huang and F.-G. Tseng, "Microfluidic Systems for Biosensing," *Sensors*, 10 (7): pp. 6623–6661, 2010;.

[5] J. Melin and S. Quake, "Microfluidic large-scale integration: The evolution of design rules for biological automation," *Annual Reviews in Biophysics and Biomolecular Structure*, 36:213–231, 2007.

[6] W. H. Minhass, P. Pop, and J. Madsen, "System-level modeling and synthesis of flow-based microfluidic biochips," in *IEEE/ACM CASES*, 2011.

[7] W. H. Minhass, P. Pop, J. Madsen, M. Hemmingsen and M. Dufva, "System-Level Modeling and Simulation of the Cell Culture Microfluidic Biochip Pro-Cell," *IEEE DTIP*, pp 91–98, 2010.

[8] J. M. Perkel, "Microfluidics - bringing new things to life science," *Science*, 322, 975–977, 2008.

[9] F. Su, K. Chakrabarty, and R. B. Fair, "Microfluidics based biochips: Technology issues, implementation platforms, and design-automation challenges," *IEEE TCAD*, pp. 211–223, 2006.

[10] T. Thorsen, S. Maerki, and S. Quake, "Microfluidic largescale integration," *Science*, 298, 580–584, 2002.

[11] D. Ullman, "NP-complete scheduling problems," *Journal of Computing System Science*, no. 10, pp. 384–393, 1975.

[12] J. P. Urbanski, W. Thies, C. Rhodes, S. Amarasinghe and T. Thorsen, "Digital microfluidics using soft lithography," *Lab on Chip*, 96–104, 2006.