Reliability-aware Synthesis for Flow-based Microfluidic Biochips by Dynamic-device Mapping

Tsun-Ming Tseng[‡], Bing Li[‡], Tsung-Yi Ho[⋆◇], and Ulf Schlichtmann[‡]
{tsun-ming.tseng, b.li, ulf.schlichtmann}@tum.de, tyho@cs.nctu.edu.tw
[‡]Institute for Electronic Design Automation, Technische Universität München, Arcisstrasse 21, D-80333 Munich, Germany
[⋆]Department of Computer Science, National Chiao Tung University, No. 1001, University Rd., 30010 Hsinchu, Taiwan
[◇]Institute for Advanced Study, Technische Universität München, Lichtenbergstrasse 2 a, D-85748 Garching, Germany

ABSTRACT

On flow-based biochips, valves that are used to form peristaltic pumps wear out much earlier than valves for transportation since the former are actuated more often, which leads to a reduced lifetime of the chip. In this paper, we introduce a valve-role-changing concept to avoid always using the same valves for peristalsis. Based on this, we generate dynamic devices from a valve-centered architecture to distribute the valve actuation activities evenly and reduce the largest number of valve actuations with even fewer valves. In addition, we propose *in situ* on-chip storages, which can overlap with other devices, so that less area is needed compared with dedicated storages on traditional chips. Moreover, our method provides good support for assays requiring different volumes and ratios of samples. Experiments show that compared with traditional designs, the largest number of valve actuations can be reduced by 72.97% averagely, while the number of valves is reduced by 10.62%.

1 Introduction

Microfluidic biochips have drawn much attention in recent years. Compared with transferring samples between equipment in a big laboratory, using small biochips that consist of all needed devices can save us much time and effort. For example, it takes 2-4 days traditionally to identify target pathogens even in the best laboratory in the world, but a few minutes are already enough when using microfluidic biochips [1]. Besides, reagents for biochemical experiments are sometimes extremely expensive. For instance, RNase inhibitor, a polyclonal antibody that is commonly used in reverse transcription polymerase chain reaction (RT-PCR) [2], costs 600 euros per milliliter in December 2014 [3]. Therefore, microfluidic biochips also have merits in cost saving, since they require smaller amounts of samples and reagents.

Usually microfluidic biochips can be classified into two types: digital biochip and flow-based biochip. The latter uses micro mechanical valve as its control unit and consists of a control layer and some flow layers. To describe the working principle of flow-based biochip, we show a general structure of valve in Figure 1(a). This valve structure consists of a pressure source, a control channel lying in the control layer and a flow channel lying in the flow layer. When a valve is open, fluids can pass through the valve without obstruction and travel along the paths formed by flow channels. To control the flow direction, we only need to close some valves by pumping air or oil from their pressure sources into their control channels, which can be inflated and thus block the flow channels underneath. Therefore, dedicated devices, such as mixers, can be constructed as shown in Figure 1(b).

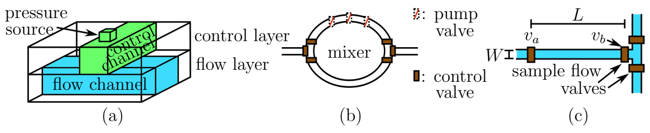


Figure 1: (a) Structure of a valve. (b) A dedicated mixer. (c) Separation of sample flow.

Design flow methodology for flow-based microfluidic biochips has been developed considerably in the last decade. The research works first target specified problems. For example the system-level modelling for a specific flow-based biochip [4]. Then the target problems become more general. As in [5], the whole design flow is considered and the possibility of mapping biochemical assays to flow-based biochip designs automatically is demonstrated. But there is a fact that has usually been neglected in early research: valves can only be actuated reliably for a few thousand times [4], and the whole chip function can be affected even when only a few valves wear out. Recently, some research works have noticed this problem and proposed some methods to reduce the number of valve actuations for guiding fluid transportation [6] [7]. However, during a mixing operation, valves for peristalsis in mixers are actuated many more times compared with valves for transportation. Since the service life of a biochip might be affected by the first worn out valve, it is more important to reduce the number of valve actuations for peristalsis.

In this work, we solve this reliability problem by mapping assay operations to dynamic devices. Our contributions include:

- We reduce the largest number of valve actuations by building dynamic devices in a valve-centered architecture. In this architecture, we define the valves for guiding fluid transportation as *control valves*, and the valves for peristalsis as *pump valves*. Besides these two types of valves that already exist on traditional flow-based biochips, we define a third type of valves as *wall valves*, which work as device boundaries and flow channel walls. During the

DAC 2015, DOI: 10.1145/2744769.2744899
http://ieeexplore.ieee.org/document/7167327/
http://dl.acm.org/citation.cfm?id=2744899

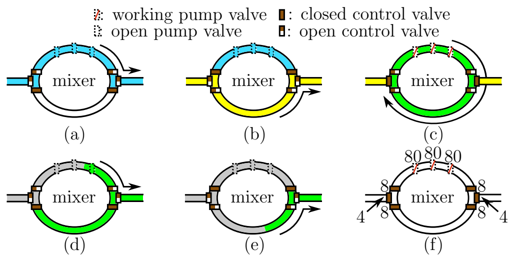


Figure 2: (a) The first input enters the mixer. (b) The second input enters the mixer. (c) Mixing starts. (d) Half of the product leaves the mixer (e) The rest product leaves the mixer. (f) The numbers of valve actuations after two mixing operations.

assay, valves can change their roles on request, so that their actuations are balanced among themselves.

- We reduce the total number of valves in most cases compared with traditional flow-based biochips, where dedicated devices are used. At first we assume a valve-centered architecture formed by virtual valves, which are used to build the dynamic devices and control channels. Then we decide which of these virtual valves will actually be actuated to execute a given assay, while the other virtual valves keep open or closed all the time. Those non-actuated virtual valves are removed, so that the final number of valves totally used in our method can be reduced.
- We introduce *in situ* on-chip storages to store the operation products temporarily. In traditional flow-based biochips, operation products can be either stored in off-chip storages, which leads to extra transport delay as well as control efforts, or stored in dedicated on-chip storages, which requires precious on-chip area. In our method, the *in situ* on-chip storages can share valves with devices as well as routing paths and thus much area can be saved.
- We adjust dynamic devices to different sizes according to the need. In traditional flow-based biochips, to deal with input samples of different volumes, dedicated devices of every particular size must be built. In our method, we use dynamic devices instead of dedicated devices, so that the devices that have completed their tasks can release their space for further use.
- We support assays with input samples in different proportions. In traditional flow-based biochips, for example, a 1:3 mixer has different port locations than a 1:1 mixer, which means that we may have to build two different mixers, even when the total volume of input samples is the same. In our method, since device boundaries can be built by valves, we are free to choose device ports from multiple locations by using the available valves and thus saving the effort to build another mixer.

2 Basic Idea and Problem Formulation

2.1 Working Principle of Valves inside Mixers

In a traditional flow-based biochip, valves can be classified into two types: control valves controlling flow directions, and pump valves forming peristaltic pumps in mixers. Figure 2 shows an example of a traditional mixer. It consists of a circular flow channel and 9 valves, 6 of which are control valves, and the other 3 are pump valves. Generally, when executing a mixing operation, we can fill the mixer with two input samples by changing the status of different control valves as

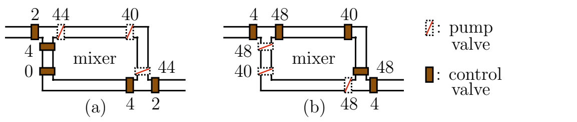


Figure 3: (a) The numbers of valve actuations after the first mixing operation. (b) The number of valve actuations after the second mixing operation.

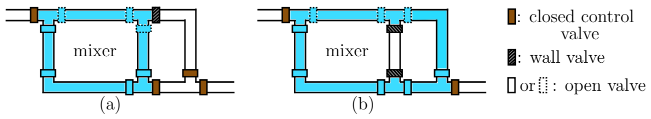


Figure 4: Mixers in different sizes using the same area: (a) A smaller mixer. (b) A larger mixer.

shown in Figure 2(a) and Figure 2(b). Then we close the control valves connected to the outside flow channels and actuate the pump valves inside this mixer repeatedly in a particular order to generate a circulation flow as shown in Figure 2(c) like the flow in a washing machine [8]. After the reaction, we use control valves to guide the mixing product out as shown in Figure 2(d) and Figure 2(e). Assuming a pump valve needs to be actuated 40 times [9] during each mixing operation, Figure 2(f) shows the total number of valve actuations of a mixer after two mixing operations. When more mixing operations are bound to the same mixer, the imbalance of total number of actuations between pump valves and control valves becomes even larger.

2.2 Our Idea: Valve-role-changing Concept

Since pump valves are actuated much more often than control valves, if we make a valve play different roles at different time, namely changing its role between control valve and pump valve, the actuations among different valves can be balanced significantly. Figure 3 shows an example implementing this valve-role-changing concept. In this example, two mixing operations are mapped to the same rectangular mixer. The mixer contains 8 valves, 2 of which only work as control valves, and the other 6 change their roles between control valves and pump valves in different operations. As shown in Figure 3(b), the largest number of valve actuations is reduced from 80 to 48 compared with Figure 2(f), which means that the service life of this mixer is nearly doubled. In addition, we only use 8 valves to form the mixer in Figure 3, instead of using 9 valves as shown in Figure 2.

Besides control valves and pump valves, we can also make some valves work as wall valves as shown in Figure 4. These valves are used as the boundary walls of a certain device and thus providing the possibility to change the size and function of devices. Therefore, a valve is no longer dedicated to a single device. After a device finishes its work, it can be treated as a free space composed by valves for further device construction. This flexibility offers us more options for placement of devices, so that we can take advantage of valves which used to be rarely actuated and therefore spread valve actuation activities more evenly.

2.3 Problem Formulation

Input:

1. A bioassay sequencing graph, which specifies operation relations, durations, volumes and input proportions.
2. A bioassay scheduling result, which specifies the start time of each operation.

Objective:

Reduce the largest number of valve actuations.

Output:

The bioassay synthesis result, which specifies the device locations, shapes and orientations.

3 Reliability-aware Synthesis

3.1 Valve-centered Architecture

In the following section, we explain how to generate a design for a certain scheduled biochemical assay from a valve-centered architecture. The idea of the valve-centered architecture is from a valve matrix proposed and manufactured by [9], in which valves are arranged regularly and every component including flow channels on the chip is completely constructed by valves. Therefore, this valve matrix is programmable just like the electrode matrix in digital biochips. However, the number of valves implemented on the chip can be very large, which leads to much control effort. In this paper, we transform that valve matrix into a valve-centered architecture with virtual valves.

In the valve-centered architecture, virtual valves are arranged regularly. A 4×4 example in a coordinate system is shown in Figure 5(a). These valves are *virtual* because some of them may not be manufactured as real valves, but removed after synthesis. The virtual valves can be used as wall valves to construct the boundary walls of devices, so that the devices can be formed and split up on request dynamically during the biochemical assay. We call such devices *dynamic devices*.

In the valve-centered architecture, different dynamic devices can share the same area without making any valve play the role as pump valve twice. For example, two 2×4 mixers with different orientations as shown in Figure 5(b)(c) can be generated in the same region at different time as shown in Figure 5(d): though the two mixers overlap with each other, their pump valves are completely different.

3.2 Dynamic-device Mapping

In our method, instead of modeling all actuation activities, we only model the actuation activities for peristaltic use, since pump valves dominate the valve actuation problem. To model the location, shape, and orientation of each dynamic device, we introduce a binary variable $s_{x,y,k,i}$ as *selection variable*. (x,y) is the left-bottom corner coordinate of a device to represent its location, for example, (0,0), (2,0), (0,2), (2,2) as shown in Figure 6(b)(c)(d)(e); k represents the index of a device type, which includes device shape and orientation, such as 1 for 3×3, 2 for 2×4, and 3 for 4×2; i is the index for the i_{th} operation. When a selection variable $s_{x,y,k,i}$ is set to 1, it means that the i_{th} operation is mapped to a device of type k at the location (x,y). Since each operation can be only mapped to a single device, we introduce the following constraint

$$\sum_{x,y,k} s_{x,y,k,i} = 1, \quad \forall i \le |O| \tag{1}$$

where O is the set of all operations in the assay.

Each time an operation is mapped to a dynamic mixer, some virtual valves related to this mixer work as pump valves. With location, shape, and orientation information of a device, the coordinates of these temporary pump valves

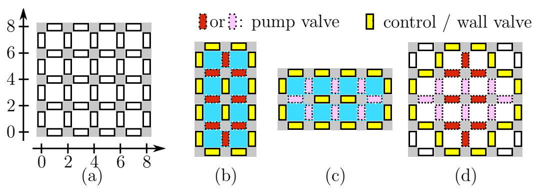


Figure 5: (a) A 4×4 valve-centered architecture (b) A 2×4 dynamic mixer. (c) A 4×2 dynamic mixer. (d) Dynamic mixers of different orientations sharing the same area.

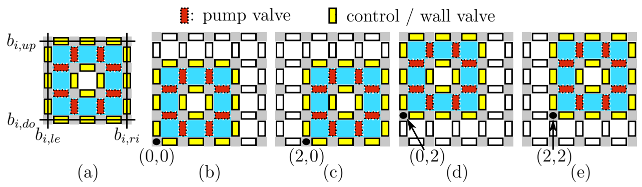


Figure 6: (a) A 3×3 dynamic mixer with 8-units volume. (b)(c)(d)(e) Four possible locations to place a 3×3 mixer.

are determined. We represent the number of valve actuations for peristaltic use of each virtual valve by an integer variable $v_{x,y}$ and calculate it as

$$v_{x,y} = \sum_{x_p,y_p,k,i} p_i s_{x_p,y_p,k,i}, \quad \forall (x,y) \in C, \quad \forall s_{x_p,y_p,t,i} \in S \tag{2}$$

where p_i is a constant representing the number of actuations of a pump valve belonging to the mixer that is mapped by the i_{th} operation, C is the set of all coordinates, and S is a set containing all selection variables $s_{x_p,y_p,k,i}$ that satisfy the following condition: when $s_{x_p,y_p,k,i}$ is set to 1, the virtual valve at (x,y) will work as pump valve.

To avoid generating different devices in the same area at the same time which may lead to pollution, we introduce four more integer variables as $b_{i,le}$, $b_{i,ri}$, $b_{i,up}$, and $b_{i,do}$. As shown in Figure 6(a), $b_{i,le}$, $b_{i,ri}$, $b_{i,up}$, $b_{i,do}$ represent the coordinates of all wall valves, which build the boundaries of the dynamic device that the i_{th} operation is mapped to. By using these variables, the non-overlapping constraints for two devices mapped by operations i_1 and i_2 can be modeled as

$$(b_{i_1,ri} \le b_{i_2,le}) \vee (b_{i_1,le} \ge b_{i_2,ri}) \vee (b_{i_1,up} \le b_{i_2,do}) \vee (b_{i_1,do} \ge b_{i_2,up}) \tag{3}$$

which can be transformed into linear form as

$$b_{i1,ri} \le b_{i2,le} + c_1 M, \tag{4}$$

$$b_{i1,le} \ge b_{i2,ri} - c_2 M, \tag{5}$$

$$b_{i1,up} \le b_{i2,lo} + c_3 M, \tag{6}$$

$$b_{i1,lo} \ge b_{i2,up} - c_4 M, \tag{7}$$

$$c_1 + c_2 + c_3 + c_4 = 3 \tag{8}$$

in which c_1, c_2, c_3, c_4 are auxiliary binary variables, and M is a very large constant. From constraint (4) to (7), when one of c_k, $k \in \{1,2,3,4\}$ is set to 1, the corresponding inequation becomes trivial. However, with constraint (8), one of the elements in the set $\{c_1,c_2,c_3,c_4\}$ must be set to 0, so that at least one of the four non-overlapping conditions can be successfully fulfilled.

With the constraints mentioned above, we build an ILP model to minimize the highest $v_{x,y}$, which is the largest number of actuations of those valves for peristaltic use. We bound this number by an integer variable w with the following constraint

$$v_{x,y} \leq w, \quad \forall (x,y) \in C \tag{9}$$

and the whole model can be described as

$$\text{Minimize: } w \tag{10}$$

$$\text{Subject to: constraints } (1)-(2),(4)-(9) \tag{11}$$

3.3 In Situ On-chip Storages

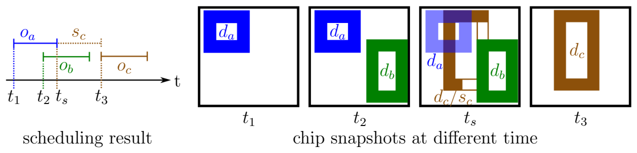


Figure 7: An example of an *in situ* on-chip storage s_c.

In a biochemical assay, the product of a preceding operation is usually the input of a later operation. Because an operation can only start after all its inputs are ready, the early coming products of preceding operations should be stored. A traditional practice is to build some dedicated storages, which needs extra chip area and can cause transport delay. In our method, with the valve-centered architecture, we generate dynamic devices ahead of schedule as *in situ* on-chip storages to store coming products, so that chip area and transportation time can be saved.

An example is shown in Figure 7, in which the scheduling result is drawn as a Gantt Chart, and the dynamic mixers are simplified and drawn as the circulation flows that they contain. o_a, o_b and o_c are mixing-operations, in which o_c takes the products of o_a and o_b as its inputs. d_a, d_b and d_c are dynamic devices for o_a, o_b and o_c. s_c is an *in situ* on-chip storage that will be transformed into d_c directly after collecting all needed inputs, and thus saving the transportation effort. We call o_a and o_b the *parent operations* of o_c. Correspondingly, d_a and d_b are called the *parent devices* of d_c and s_c.

At time t_s, o_a is completed and thus the valves which have constructed d_a can be treated as free valves, so that we can build s_c by using some of these valves to store the product of o_a immediately. Since s_c only contains the product of o_a at time t_s, there is still some free space inside it. In our method, we take advantage of those free space by allowing them to overlap with their parent devices. In this example, o_b is in process at time t_s. Therefore, s_c only occupies part of the later d_c until o_b is completed at time t_3. Then s_c is turned to d_c by using the free valves of the former d_b, and the product of o_b can be also conveniently led to d_c for the coming operation.

To implement this special overlapping permission to our ILP model, we only need to add an auxiliary binary variable c5 to constraint (8)

$$c_1 + c_2 + c_3 + c_4 = 3 + c_5. \tag{12}$$

If c_5 is set to 1, c_1, c_2, c_3 and c_4 must all be 1, which permits the overlapping between two devices. But if we do not want this overlapping to happen, we can set c_5 to 0, so that the meaning of this constraint will be the same as constraint (8).

3.4 Routing-convenient Mapping

To save the transportation time, we force two sequential operations to be mapped to two close devices, so that transportation time can be saved. A constant d, which is the minimum dimension of all devices, is set to the maximum distance between the dynamic devices for two sequential operations, so that no other device can be inserted between them, and the routing path can be directly constructed.

To introduce the distance limit d to our model so that the dynamic-device mapping can be routing-convenient, we add four more constraints by using the device boundary coordinates as shown in Figure 6(a) to the ILP model as

$$b_{i1,ri} > b_{i2,le} - d, \tag{13}$$

$$b_{i1,le} < b_{i2,ri} + d, \tag{14}$$

$$b_{i1,up} > b_{i2,lo} - d, \tag{15}$$

$$b_{i1,lo} < b_{i2,up} + d \tag{16}$$

where i_1 is the parent operation of i_2.

3.5 Routing between Devices and Chip Ports

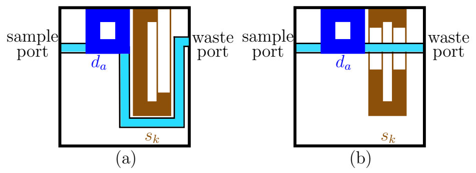


Figure 8: (a) The storage s_k is an obstacle for routing paths. (b) The storage s_k can be passed through by routing paths.

Besides our routing-convenient dynamic-device mapping process,we also propose a method to determine the routing paths between devices and chip ports that are connected to off-chip sample pumps or waste sinks.

Our valve-role-changing concept brings us more options for routing. When an *in situ* on-chip storage still has enough free space, we allow routing paths to pass through this storage as shown in Figure 8(b), thus saving the efforts for a long detour as shown in Figure 8(a). We apply Dijkstra's shortest path algorithm to decide the routing paths. And we restrict the crossings of routing paths by assigning higher costs to the area occupied by previously-routed paths, so that we can transport samples in parallel.

3.6 Overall Algorithm

Algorithm 1 gives an overall view of our methods. After reading the program input in L1 and building the data structure in L2, we perform our dynamic-device mapping by using an ILP model in L3-L9 and then decide the routing paths in L10-L19. In the proposed method, we allow overlapping in two particular situations: overlapping between storages and parent devices, and overlapping between storages and routing paths. After overlapping, the remaining area inside the storages may not be big enough to store the required products. In our model, the constraints that can be used to prevent this possibility are not included to save program runtime. Instead, when the overlapping area between a storage s and a device d exceeds the remaining free space in s, we set c_5 in constraint (12) to 0 to prevent s and d from overlapping again, and re-run the dynamic-device mapping. Analogously, when the overlapping area between a storage s and a path p exceeds the remaining free space in s, we treat s as an obstacle, rip p, and re-route a new path.

After dynamic devices are generated and the routing paths are decided, in L20, we transform the virtual valve-centered architecture into a real design by removing the virtual valves that are never actuated and implementing the remaining

Algorithm 1: Reliability-aware Synthesis

```
Read sequencing graph and scheduling result.
Build virtual valves in valve-centered architecture.
# Dynamic−device Mapping
repeat
    Build and solve ILP model for dynamic-device mapping.
    if overlapping area of (storage s, device d) > free space of s then
        Forbid (s,d) from overlapping with each other.
    end
until feasible dynamic-device mapping;
# Routing
for time t=1 to maxT do
    forall the connections do
        Route a path using Dijkstra's algorithm.
        if overlapping area of (storage s, path p) > free space of s then
            Forbid (s,p) from overlapping with each other.
            Rip p and re-route.
        end
    end
end
Remove non-actuated valves.
```

valves, so that the valve actuation activities are balanced even with fewer valves.

4 Experimental Results

We implemented the reliability-aware synthesis in C++ on a computer with a 2.67 GHz CPU. The ILP model for dynamic-device mapping was solved by the ILP solver Gurobi [10]. The four test cases are from widely used laboratory protocols [11] [12]. For each test case we set up three different policies. As the policy index increases, we increase the number of mixers used in a traditional design, in which dedicated mixers, storages, and detectors are used. Correspondingly, we can obtain different scheduling results as the inputs for experiments. We compare the experimental results of our method in two different settings with the results of the optimal binding for the traditional designs in Table 1, in which the meaning of each column is:

$\#op$ **:** the number of operations and mixing operations thereof.
$Po.$ **:** the policy index.
$\#d$**:** the number of devices, including mixers and detectors.
$\#m_{4-6-8-10}$**:** the numbers of operations bound to the same mixers, while the hyphens separate mixers of different sizes.
vs_t_{max}**:** the largest number of valve actuations applying the optimal binding for the traditional designs.
vs_1_{max}, vs_2_{max}**:** the largest number of valve actuations and actuations for peristalsis thereof applying our methods in different settings.
$\#v$**:** the number of used valves.
imp_1_{vs}, imp_2_{vs}**:** the improvements in the largest number of valve actuations in different settings.
imp_v**:** the improvement in the number of valves.
T**:** the program runtime.

In the traditional designs, we assume there are 4 different sizes of mixers: 4, 6, 8, and 10. Each design contains a storage to store products temporarily, and the number of cells in the storage is determined by the largest number of simultaneous accesses to the storage. Each assay operation, according to the volume of its inputs, is assigned to a mixer with the required size. If there are multiple mixers with the same size, we apply an optimal binding regarding valve actution by distributing operations to mixers as evenly as possible. Because the loadings on mixers with different sizes may vary considerably, we add one more mixer for each mixer type that is under the heaviest loading as the policy index increases to alleviate the heavy burden.

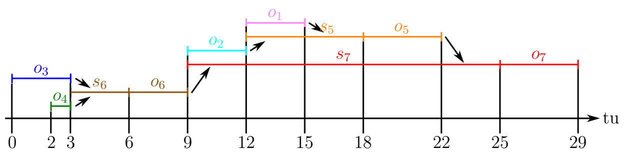


Figure 9: The scheduling result of case PCR in p1.

When we introduce more and more mixers as the policy index increases, the valve actuation activities for peristalsis are distributed much more evenly among different dedicated mixers. However, because the roles of valves cannot be changed in the traditional method, introducing more mixers leads to more rarely used valves, thus enlarging the total number of valves.

In our method, we first built a square matrix containing virtual valves based on the valve-centered architecture. Then we built and solved the model for dynamic-device mapping and routed the sample paths. We calculated the largest number of valve actuations in vs_1_{max} and vs_2_{max} in Table 1, which are close to the numbers of actuations for peristalsis thereof. This fact validates our method in Section 3.2, where we only model actuation activities for peristalsis.

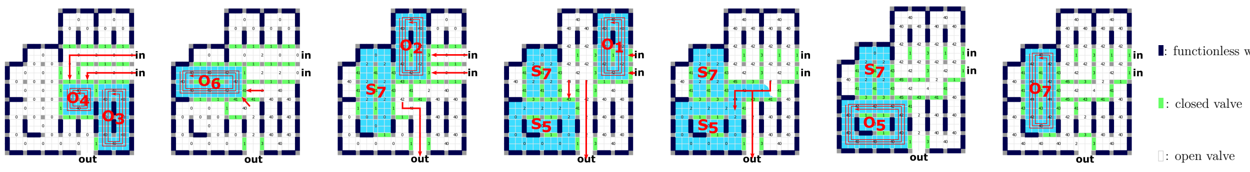
In our model, as mentioned in Section 3.1, all valves passed by the circulation flow inside a dynamic mixer are regarded as pump valves. For example, the 2×4 dynamic mixer as shown in Figure 5(b) uses 8 pump valves, while the dedicated mixer as shown in Figure 2(f) only uses 3 pump valves. Though in our method we use more pump valves, so that theoretically the loading on each valve should be alleviated under the same efficiency, it is difficult to tell how many actuation times are sufficient for a single mixing operation.

Therefore, in our first setting we still assume each pump valve is actuated 40 times for a single mixing operation, which is exactly the same as the pump valve works in a dedicated mixer in the traditional method as a conservative comparison. imp_1_{max} shows that even with this conservative setting, we still reduce the largest number of actuations by more than half of that in the traditional method. In our second setting, we directly use the synthesis result obtained in the first setting, but change the number of actuations of each valve for a single mixing operation, so that the total number of actuations is the same as that in the traditional method. For example, the total number of valve actuations of a dedicated mixer for a single mixing operation is 3×40=120, so we change the number of actuations of each valve in the mixer using 8 pump valves to 15. As shown in imp_2_{max}, the results are much better, even with a small number of valves shown in $\#v$.

To show the working principles of our method intuitively, we take the synthesis result of case PCR with 7 mixing operations in p1 as an example. The input of our method is the scheduling result of this case with 3 time-units (tu) as the transport delay. As shown in Figure 9, o_1 and o_2 are the parent operations of o_5, while o_3 and o_4 are the parent operations of o_6, and o_5 and o_6 are the parent operations of o_7. At t = 3tu, the storage s_6 appears immediately after o_3 and o_4 are completed, and begins to store the products of o_3 and o_4. Likewise, s_5 appears immediately after o_2 is completed at t = 12tu, since o_2 is completed earlier than o_1. Finally, o_7 takes the products of o_5 and o_6, while s_7 has been

Table 1: Comparison of the highest valve actuation times and the number of valves.

			Optimal Binding for Traditional Designs				Our Method						
	$\#op$	$Po.$	$\#d$	$\#m_{4-6-8-10}$	vs_t_{max}	$\#v$	vs_1_{max}	imp_1_{vs}	vs_2_{max}	imp_2_{vs}	$\#v$	imp_v	T
PCR	15(7)	p1	3	1-0-4-2	160	83	45(40)	71.88%	35(30)	78.13%	71	14.46%	0.8
		p2	4	1-0-(2,2)-2	80	99	45(40)	43.75%	34(30)	57.50%	76	23.23%	0.8
		p3	6	1-0-(2,1,1)-(1,1)	80	131	43(40)	46.25%	31(30)	61.25%	82	37.40%	0.9
Mixing Tree	37(18)	p1	4	2-4-5-7	280	108	93(80)	66.79%	46(42)	83.57%	105	2.78%	2.9
		p2	5	2-4-5-(4,3)	200	124	93(80)	53.50%	46(42)	77.00%	105	15.32%	2.9
		p3	6	2-4-(3,2)-(4,3)	160	140	90(80)	43.75%	60(50)	62.50%	124	11.43%	3.3
Interpolating Dilution	71(35)	p1	7	5-9-9-(6,6)	360	178	145(120)	59.72%	72(65)	80.00%	176	1.12%	357.1
		p2	9	5-(5,4)-(5,4)-(6,6)	240	207	94(80)	60.83%	56(42)	76.67%	207	0.00%	87.8
		p3	10	5-(5,4)-(5,4)-(4,4,4)	200	225	92(80)	54.00%	56(50)	72.00%	208	7.56%	101.2
Exponential Dilution	103(47)	p1	10	6-(8,8)-(7,6)-(6,6)	320	241	135(120)	57.81%	75(75)	76.56%	214	11.20%	485.3
		p2	11	6-(6,5,5)-(7,6)-(6,6)	280	254	134(120)	52.14%	71(65)	74.64%	255	-0.39%	488.9
		p3	12	6-(6,5,5)-(5,4,4)-(6,6)	240	268	99(80)	58.75%	58(40)	75.83%	259	3.36%	314.3
average								55.76%		72.97%		10.62%	



(a) t = 2tu. (b) t = 6tu. (c) t = 9tu. (d) t = 12tu. (e) t = 15tu. (f) t = 18tu. (g) t = 25tu.

Figure 10: Snapshots of the synthesis result of case PCR in p1 in the first setting.

formed since t = 9tu, immediately after o_5 is completed.

Figure 10 shows some snapshots of the synthesis result. We assume that there are two input ports for samples and reagents, and one output port for waste and final product. The *in situ* on-chip storages s_7 and s_5 are generated respectively at 9tu and 12tu in Figure 10(c)(d), and become mixers for o_7 and o_5 directly later at 25tu and 18tu in Figure 10(g)(f). Allowing overlap between storage s_7 and its parent device s_5 is demonstrated at 12tu in Figure 10(d). Thanks to our routing-convenient model, we only need trivial routings between devices, and the longest one in this example is from the dynamic mixer for o_1 to s_5 at 15tu as shown in Figure 10(e). The routings between devices and chip ports is shown by the flow directions denoted by arrows.

5 Conclusion

In this paper we have addressed a reliability problem of flow-based biochips due to unbalanced valve actuation. The problem is solved by the proposed reliability-aware synthesis based on a virtual valve-centered architecture with valve-role-changing concept. Indeed, the architecture may also bring benefits to some aspects other than reliability, such as to speed up the bioassay execution, which will be considered in the future. Besides, in this work, we assume that we can freely manipulate sample flows, which needs to be restricted and will be considered in the future as well.

6 References

[1] P. Yager, T. Edwards, E. Fu, K. Helton, K. Nelson, M. R. Tam, and B. H. Weigl. Microfluidic diagnostic technologies for global public health. *Nature*, 442:412–418, 2006.

[2] R. S. Tuan and C. W. Lo. *Developmental Biology Protocols: Volume III.* Humana Press, 2000.

[3] Qiagen, Inc. Qiagen rnase inhibitor, 2014.

[4] W. H. Minhass, P. Pop, J. Madsen, M. Hemmingsen, and M. Dufva. System-level modeling and simulation of the cell culture microfluidic biochip procell. In *IEEE DTIP*, pages 91–98, 2010.

[5] W. H. Minhass, P. Pop, and J. Madsen. System-level modeling and synthesis of flow-based microfluidic biochips. In *Proc. Int. Conf. Compil., Arch. and Syn. Embed. Sys.*, pages 225–234, 2011.

[6] K.-H. Tseng, S.-C. You, W. H. Minhass, T.-Y. Ho, and P. Pop. A network-flow based valve-switching aware binding algorithm for flow-based microfluidic biochips. In *Proc. Asia and South Pacific Des. Autom. Conf.*, pages 213–218, 2013.

[7] K.-H. Tseng, S.-C. You, J.-Y. Liou, and T.-Y. Ho. A top-down synthesis methodology for flow-based microfluidic biochips considering valve-switching minimization. In *Proc. Int. Symp. Phy. Des.*, pages 123–129, 2013.

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

[9] L. M. Fidalgo and S. J. Maerkl. A software-programmable microfluidic device for automated biology. *Lab on a Chip*, 11:1612–1619, 2011.

[10] Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2013.

[11] H. Ren, V. Srinivasan, and R. Fair. Design and testing of an interpolating mixing architecture for electrowetting-based droplet-on-chip chemical dilution. In *Int. Conf. on Solid-State Sensors, Actuators and Microsystems*, pages 619–622, 2003.

[12] K. Chakrabarty and F. Su. *Digital Microfluidic Biochips: Synthesis, Testing, and Reconfiguration Techniques.* Boca Raton, FL: CRC Press, 2006.