Reliability-aware Synthesis with Dynamic Device Mapping and Fluid Routing for Flow-based Microfluidic Biochips

Tsun-Ming Tseng, *Member, IEEE*, Bing Li, Mengchu Li, *Student Member, IEEE*, Tsung-Yi Ho, *Senior Member, IEEE*, and Ulf Schlichtmann, *Member, IEEE*

***Abstract*—In flow-based biochips, peristaltic pumps consisting of valves are essential to generate circulation flow in a mixer. When a peristaltic pump is activated, the related valves for peristalsis are required to be actuated for many times. However, the roles of valves in traditional chips are fixed, and therefore the valves for peristalsis can wear out much faster than the valves for guiding fluid transportation. This could lead to a reduced lifetime of the chip, because the whole chip function can be affected when just a few or even only a single valve wears out. In this paper, we propose a valve-centered architecture with virtual valves, based on which we introduce a valve-role-changing concept to balance the valve actuations. By switching a valve into different roles, microfluidic components such as mixers, storages and flow channels can be formed dynamically during the assay process, which enables us to balance the utilization of valves, and synthesize designs that support various kinds of operations. Compared with our preliminary work, we further decrease the largest number of valve actuation as well as the number of valves by the revised dynamic device mapping and fluid path routing. For dynamic device mapping, we introduce a virtual-boundary concept to generate devices at better places while connections between devices are still guaranteed. For fluid path routing, we accurately model valve actuation resulting from our valve-actuation-aware routing, and revise the results by rip-up and reroute. In addition to performance, we improve the reliability of our method by assuring fluid paths from devices to chip boundaries. Experiments show that the new method can be 8 times better than the traditional method, and outperforms our preliminary work for large cases even with fewer valves.**

***Index Terms*—Flow-based microfluidic biochip, reliability, dynamic device, pump valve, routing.**

I. Introduction

Microfluidic biochips have drawn much attention in recent years, because they can replace various cumbersome laboratory equipment by integrating all devices into one small chip.

The work of T.-Y. Ho was supported in part by the Ministry of Science and Technology of Taiwan, under Grant MOST 102-2221-E-007-149-MY3 and 104-2220-E-007-021 and in part by the Technische Universität München — Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763. The preliminary version of this paper was published in the Proceedings of the 52nd Annual Design Automation Conference (DAC), 2015.

Tsun-Ming Tseng, Bing Li, Mengchu Li and Ulf Schlichtmann are with the Institute for Electronic Design Automation, Technische Universität München (TUM), Munich 80333, Germany (e-mail: tsun-ming.tseng@tum.de; b.li@tum.de; mengchu.li@campus.lmu.de; ulf.schlichtmann@tum.de).

Tsung-Yi Ho is with National Tsing Hua University, No. 101, Section 2, Kuang-Fu Road, Hsinchu 30013, Taiwan (e-mail: tyho@cs.nthu.edu.tw).

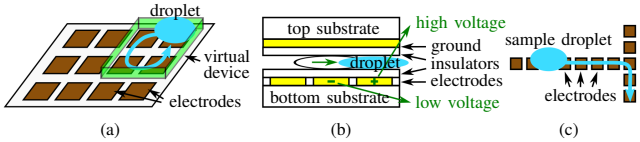


Figure 1: (a) Electrodes in a digital biochip. (b) Sectional graph of droplet movement. (c) A droplet path.

Since it is much faster to move samples between devices in a chip than between equipment in a big laboratory, using a biochip to execute biochemical assays can save 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) to protect RNA from degradation by inhibiting the activity of RNases [2] [3], costs 600 euros per milliliter in June 2015 [4]. Since microfluidic biochips require smaller amounts of samples and reagents, they also have merits in cost saving.

Microfluidic biochips can be classified into two types: digital biochip and flow-based biochip. In digital biochips, electrodes, which are connected to voltage sources, are usually arranged regularly as shown in Figure 1(a). By changing the applied voltages on different electrodes, a droplet can be attracted from an electrode with a lower applied voltage to an electrode with a higher applied voltage as shown in Figure 1(b). Therefore, when we switch the voltages on electrodes along a path alternatively, a moving path for the droplet and thus also a virtual device such as a mixer can be formed as shown in Figure 1(c) and Figure 1(a), respectively.

Based on this working principle, regular-placed electrodes provide a general platform to perform different assays in the same chip. Since the electrodes can easily be integrated and manufactured in large scale, digital biochips also provide good support for complicated assays with numerous operations. However, it is difficult for the electrical field force on the electrodes to deal with large droplets: It may be too strong so that it can tear up the droplets, or it may be too weak to drag them [5]. Therefore, operations taking large droplets as inputs need to be split into several operations with small input volumes, and their results need to be combined back

TCAD Dec. 2016, DOI: 10.1109/TCAD.2016.2547902
http://ieeexplore.ieee.org/document/7442807/

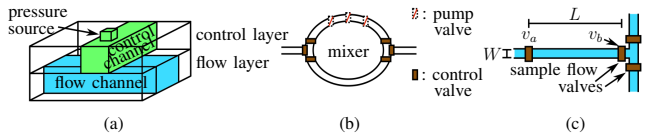


Figure 2: (a) Structure of a valve. (b) A dedicated mixer. (c) Separation of sample flow.

afterwards, which may prolong assay execution time [6]. Furthermore, in digital biochips, the only input ratio that is allowed for mixing and splitting operations in most works is 1:1, which may cause quantization error in many sample preparation applications [7] [8] [9].

Compared with digital biochips, flow-based biochips provide the possibility to perform biochemical operations requiring various input volumes and ratios directly. Flow-based biochips, which is the other important type of microfluidic biochips, consist of some flow layers and a control layer, and uses micro-mechanical valves as its control units. Valve is the most fundamental component to perform any active control in flow-based biochips. Since a valve is typically made from the material Polydimethylsiloxane (PDMS), it is pneumatic and its shape can be changed if inflated.

In Figure 2(a), we show the general structure of a valve. 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 guide the fluid to its destination, we only need to close some valves by pumping air or oil from their pressure sources into their control channels, which will be inflated and thus block the flow channels underneath. Therefore, dedicated devices, such as mixers, can be constructed as shown in Figure 2(b).

In order to separate fluids into operation inputs of peculiar volumes and ratios, which is also called *volume metering*, biologists build fixed-length channels that are enclosed by two valves. An example is shown in Figure 2(c), in which the width of the flow channel is W, and the distance between valves v_a and v_b is L. By closing v_a and v_b, we can separate the required $W \times L$ volume from the main sample flow and transport it to a device easily.

Design methodology for flow-based microfluidic biochips has been developed considerably in the last decade. The research works first targeted specified problems. For example, the system-level modeling for a specific flow-based biochip [10]. Then the target problems became more general. As in [11], the whole design flow was considered and the possibility of mapping biochemical assays to flow-based biochip automatically was demonstrated.

However, all the proposed work concentrated on flow-based biochips integrated with specific-purpose microfluidic devices, which can only support a specialized type of operations. Even a slight change of operation protocols may involve the fabrication of a new device, or the re-design of the whole chip. Therefore, the demand for reconfigurable or programmable microfluidic hardware has arised [12]. [13] proposed and manufactured a general-purpose software-programmable microfluidic chip which is constructed with regularly arranged valve matrix and is capable of performing various biochemical operations without hardware modifications. However, this design requires a much larger number of valves than traditional designs, which results in much more control efforts.

Another issue of flow-based biochips used to be neglected in early research is the reliability of valves. As proposed in [10] [14], the valves they applied are only promised to be actuated reliably for a few thousand times, and the whole chip function can be affected when just a few or even only a single valve wears out. Recently, some research works have noticed this problem and proposed methods to reduce the number of valve actuations for guiding fluid transportation [15] [16]. 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 classify the roles of a valve into three categories: *control valves* for guiding fluid transportation; *pump valves* for peristalsis; and *wall valves* for forming device boundaries and flow channels. We transform the valve matrix proposed in [13] into a valve-centered architecture with virtual valves, based on which we propose a dynamic device mapping method that synthesizes designs which support various kinds of operations with a reduced number of valves, and solves the reliability problem caused by unbalanced valve actuations.

Compared with the traditional method, our contributions include:

- We propose the first synthesis method for a valve-centered general-purpose biochip architecture, which generates dynamic chip layouts from the sequencing graph and scheduling results of a bioassay.
- We balance the valve actuations and thus enhance the service life of the whole chip by introducing a valve-role-changing concept and applying dynamic device mapping. Experimental results show that compared with the traditional method, we reduce the largest number of valve actuations by 45%–80%.
- We reduce the number of valves by proposing *virtual valves* in the synthesis process. The valves are called "virtual", because instead of fabricating every valve in the valve-centered architecture, some of the valves will be removed from the chip or replaced by functionless wall at the end of the synthesis process. Experimental results show that compared with the traditional method, besides the reduction of valve actuations, we also reduce the number of valves by 5%–25% in most cases
- Other benefits of our method include:
  + We introduce *in situ* on-chip storages to store operation products temporarily. Compared with off-chip or dedicated on-chip storages, our method saves the transportation delay and additional control efforts for off-chip communication, without aggravating the chip area cost.
  + We generate *dynamic devices* according to the need of each operation to fulfil its specific requirement on input

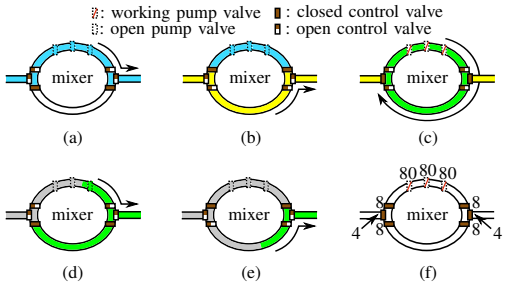


Figure 3: (a) The first input enters the mixer. (b) The second input enters the mixer. (c) Mixing starts. (d) Half of the product leaves the mixer (e) The rest product leaves the mixer. (f) The numbers of valve actuations after two mixing operations.

volume and ratio. Compared with dedicated specialized devices, dynamic devices do not occupy chip areas permanently, and the space they once occupied can be released for further use.

- Compared with our preliminary work [17], major technical improvements include:
  + We propose *virtual boundaries* for dynamic devices, which provide more flexibility to the dynamic device mapping, and result in a better solution with fewer valve actuations.
  + We improve the reliability of our method by assuring fluid paths from devices to chip ports.
  + We improve our routing algorithm by taking valve actuations caused by path routing into consideration, and revise our routing results by rip-and-reroute to approximate the optimal solution.
  + Experimental results show that besides the reliability improvements, in most cases, we further reduce the largest number of valve actuations by up to 35%, and the number of valves by up to 12%.

The rest of this paper is organized as follows: In Section II, we introduce the reliability issue and our basic idea. In Section III, we propose an ILP model to perform dynamic device mapping and then route the fluid paths. We show experimental results of the numbers of valve actuations and the number of valves in Section IV, and conclude our work in Section V.

II. Basic Idea and Problem Formulation

In this section, we first introduce the reliability issue in valve actuation. Then we show how to distribute valve actuations evenly by assigning different roles to valves on request. At the end of this section, we define the problem formulation of our work.

A. Working Principle of Valves inside Mixers

In a traditional flow-based biochip, valves can be classified into two types: control valves that are used to control flow directions, and pump valves that are used to form peristaltic pumps in mixers. Figure 3 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

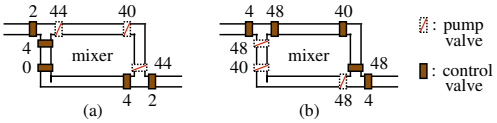


Figure 4: (a) The numbers of valve actuations after the first mixing operation. (b) The number of valve actuations after the second mixing operation.

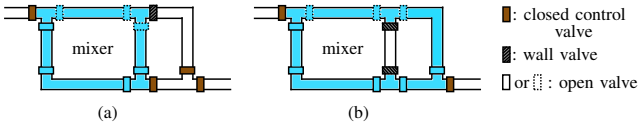


Figure 5: Mixers in different sizes using the same area: (a) A smaller mixer. (b) A larger mixer.

the mixer with an input sample and a reagent by changing the status of different control valves as shown in Figure 3(a) and Figure 3(b). Then we close the control valves connected to the outer flow channels and actuate the pump valves inside this mixer repeatedly in a particular order to generate a circulation flow as shown in Figure 3(c) like the flow in a washing machine [18]. After the reaction, we use control valves to guide the mixing product out as shown in Figure 3(d) and Figure 3(e). Assuming a pump valve needs to be actuated 40 times [13] during each mixing operation, Figure 3(f) shows the total number of valve actuations of a mixer after two mixing operations. We can see that some valves are actuated 80 times while the others are only actuated 8 or 4 times. When more mixing operations are bound to the same mixer, the imbalance of sum of actuations between pump valves and control valves becomes even larger.

B. 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 times, namely changing its role between control valve and pump valve, the actuations can be balanced significantly among different valves. Figure 4 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 4(b), the largest number of valve actuations is reduced from 80 to 48 compared with Figure 3(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 4, instead of using 9 valves as shown in Figure 3.

Besides control valves and pump valves, we can also make some valves work as wall valves as shown in Figure 5. These valves are used as the boundary walls of a certain device and thus provide 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 generating devices, so that we can take advantage of valves which used to be

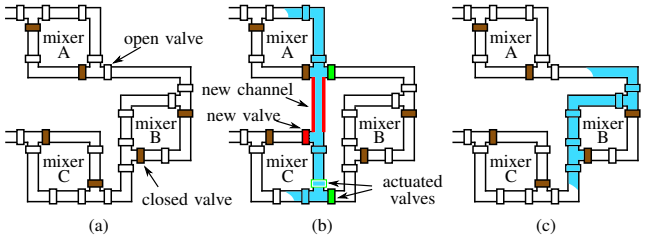


Figure 6: Flow channel sharing: (a) Original design. (b) A dedicated channel from mixer A to mixer C. (c) Part of mixer B used as a channel for fluid transportation.

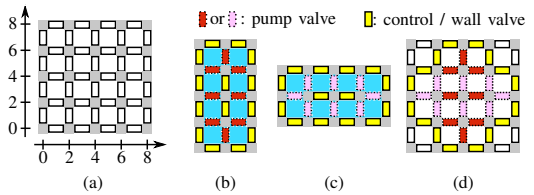


Figure 7: (a) A 4×4 valve-centered architecture (b) A 2×4 dynamic mixer. (c) A 4×2 dynamic mixer. (d) Dynamic mixers of different orientations sharing the same area.

rarely actuated and therefore spread valve actuation activities more evenly.

C. Flow Channel Sharing

By changing the roles of valves on request, valves that were part of previously-generated devices or previously-routed fluid paths can be reused to form a new flow channel as well. An example is shown in Figure 6, which shows a chip with three mixers. In this example, the product of an operation performed in mixer A is the input of another operation performed in mixer B. Similarly, the product of an operation performed in mixer B is the input of another operation performed in mixer C. To carry out the transportations, we build direct flow channels from mixer A to mixer B and from mixer B to mixer C respectively as shown in Figure 6(a). The status of valves as open or closed in each mixer are similar to the valves of a mixer just finishing a mixing operation as shown in Figure 3(e).

When there is one more pair of sequential operations that are bound to mixer A and mixer C respectively, a traditional approach is to build one more dedicated channel between mixer A and mixer C, by which a new flow channel, one more new valve, and some more valve actuations on existing valves are introduced as shown in Figure 6(b). However, if flow channels inside devices are no longer dedicated to these devices, we can use part of other devices that are not working for the time being, e.g. mixer B in Figure 6(c) to transport fluids. Compared with the cost caused by the new channel in Figure 6(b), as shown in Figure 6(c) we do not even need an extra valve actuation.

D. Problem Formulation

We use the valve-role-changing concept to solve the reliability problem of valves. The problem formulation of this work is defined as:

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:

1. Reduce the largest number of valve actuations by distributing them evenly.
2. Reduce the number of valves.

Output:

1. The result of dynamic device mapping, which specifies the device locations, shapes and orientations.
2. The routing result of fluid paths.

III. Reliability-aware Synthesis

In this section, we first show the working principles of a valve-centered architecture, and how to utilize this architecture to support our valve-role-changing concept. Then we introduce an ILP model for a routing-convenient dynamic device mapping as well as the idea of *in situ* storages. We show our methods of improving the routing reliability and describe our routing algorithm, which takes the impact on valve actuations and the sum of valves into consideration.

A. Valve-centered Architecture

The idea of the valve-centered architecture is from a valve matrix proposed and manufactured by [13], in which valves are arranged regularly and every component including flow channels in the chip is completely constructed by valves, and the basic unit of a flow channel is a chamber encircled by four valves. Therefore, this valve matrix is programmable just like the electrode matrix in digital biochips. However, the number of valves implemented in 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 7(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 7(b)(c) can be generated in the same region at different time as shown in Figure 7(d): though the two mixers overlap with each other, their pump valves are completely different.

B. Dynamic Device Mapping

To generate dynamic devices at the best locations in the valve-centered architecture and thus achieve the most reliable designs, we propose an integer linear programming (ILP) model to accurately model valve actuation brought by con-

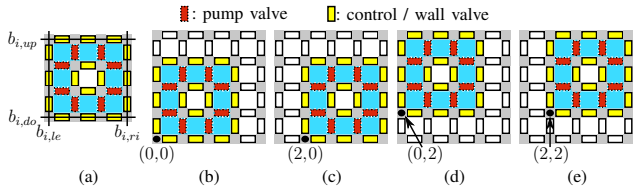


Figure 8: (a) A 3×3 dynamic mixer with 8-unit volume. (b)(c)(d)(e) Four possible locations to place a 3×3 mixer.

structing dynamic devicces. In this model, instead of modeling all actuation activities, we only model the actuation activities for peristalsis, since pump valves dominate the valve actuation problem. To determine 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 8(b)(c)(d)(e); k represents the index of a device type, including shape and orientation of the device, 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

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

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

Each time when an operation is mapped to a dynamic mixer, some virtual valves related to this mixer will work as pump valves. With location, shape, and orientation information of a device, the coordinates of these temporary pump valves are ascertained. We represent the number of valve actuations for peristalsis of each virtual valve by an integer variable $v_{x,y}$ and calculate it as

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

where p_i is a constant representing the number of actuations for a pump valve to perform the i_{th} mixing 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, a k-type mixer will be generated with left-bottom corner at (x_p,y_p) to perform the i_{th} mixing opration, and the virtual valve at (x,y) will work as one of its pump valves.

To avoid generating different devices in the same area at the same time, we introduce four more integer variables as $b_{i,le}$, $b_{i,ri}$, $b_{i,up}$, and $b_{i,do}$. As shown in Figure 8(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

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

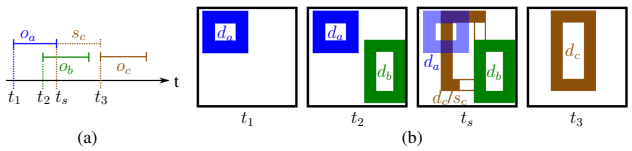


Figure 9: An example of an *in situ* on-chip storage s_c: (a) Scheduling result. (b) Chip snapshots at different time.

which can be transformed into linear form as

$$b_{i1,ri} \leq b_{i2,le} + c_1 M, \tag{4}$$
$$b_{i1,le} \geq b_{i2,ri} - c_2 M, \tag{5}$$
$$b_{i1,up} \leq b_{i2,lo} + c_3 M, \tag{6}$$
$$b_{i1,lo} \geq b_{i2,up} - c_4 M, \tag{7}$$
$$c_1 + c_2 + c_3 + c_4 = 3 \tag{8}$$

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 peristalsis. We bound this number by an integer variable w with the following constraint

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

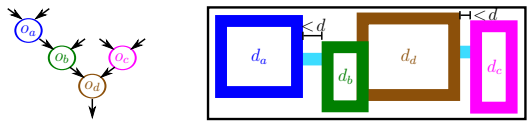
and the whole model can be described as

$$\text{Minimize: } w \tag{10}$$
$$\text{Subject to: constraints } (1)-(2),(4)-(9) \tag{11}$$

C. In Situ On-chip Storages

In a biochemical assay, the product of a preceding operation is usually the input of a later operation. We call the preceding operation *parent operatio* of the later operation, and the later operation *child operatio* of the preceding operation. Correspondingly, the device performing the parent operation is called the *parent device* of the device performing the child operation, and the device performing the child operation is called the *child device* of the device performing the parent operation. Because an operation can only start after all its inputs are ready, the early coming products of preceding operations need to be stored. A traditional practice is to build some dedicated storages, which need extra chip area and can cause transport delay. In our method, with the valve-centered architecture, we generate dynamic devices 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 9, 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 and therefore o_c is the child operation of o_a and o_b. d_a, d_b, and d_c are dynamic devices



(a) (b)

Figure 10: An example of routing-convenient dynamic device mapping: (a) Sequencing graph. (b) Device locations.

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 inputs, and thus save the transportation effort.

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 spaces 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 also conveniently be 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).

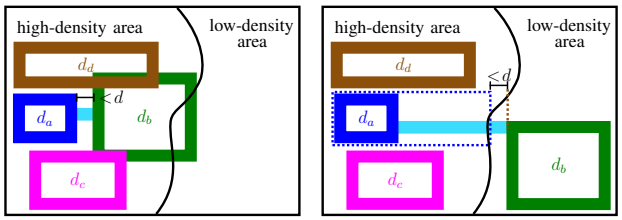
D. Routing-convenient Mapping

Our dynamic device mapping also guarantees the transportation paths between parent and child devices, which brings convenience to routing. When we map two sequential operations to two different devices, we prefer to build a direct connection between these devices to save the transportation effort. In order to do that, we introduce a constant d, which is the minimum dimension of all devices, as the maximum distance between a parent device and its child device. This distance limit can be introduced to our model by adding four more constraints:

$$b_{i1,ri} > b_{i2,le} - d, \tag{13}$$
$$b_{i1,le} < b_{i2,ri} + d, \tag{14}$$
$$b_{i1,up} > b_{i2,lo} - d, \tag{15}$$
$$b_{i1,lo} < b_{i2,up} + d \tag{16}$$

where i_1 is the parent operation of i_2.

These constraints ensure that the distance between a parent device and its child device is short enough to prevent other devices from being inserted between them and thus obstructing their connection path. For example, as shown in Figure 10(a), suppose o_a is the parent operation of o_b, and o_d is the child operation of o_b and o_c. If these operations are mapped to different devices, by controlling their device locations, direct connections can be easily built between parent and child devices as shown in Figure 10(b).



(a) (b)

Figure 11: Device location of d_b: (a) Without virtual boundaries. (b) With virtual boundaries.

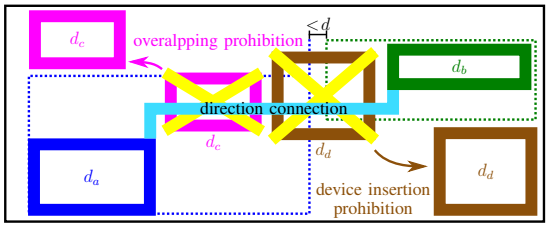


Figure 12: Guarantee of direct connection from d_a to d_b.

However, with a strict distance control, the number of potential device locations will be remarkably reduced, since a child device has to be put next to its parent device. An example is shown in Figure 11(a), the left part of the chip is crammed with devices while the other part is left unused. This limitation may keep us from finding an optimal solution and lengthen the optimization process.

Since our target is to build direct connections between parent and child devices, but not necessarily to put them close to each other, we introduce *virtual boundaries* to refine our distance controlling method. The virtual boundaries of a device circle a *virtual area* which is larger than or equal to the real size of this device, which can be introduced to our model by adding following constraints:

$$b'_{i,ri} \geq b_{i,ri}, \tag{17}$$
$$b'_{i,le} \leq b_{i,le}, \tag{18}$$
$$b'_{i,up} \geq b_{i,up}, \tag{19}$$
$$b'_{i,lo} \leq b_{i,lo} \tag{20}$$

where $b'_{i,ri}$, $b'_{i,le}$, $b'_{i,up}$, and $b'_{i,lo}$ are the virtual boundaries of the dynamic device that the i_{th} operation is mapped to.

Instead of controlling the distance between exact device locations, we control the distance between virtual areas, which can be easily introduced to our model by replacing the boundary variables in constraints (13)-(16) with virtual boundary variables. As shown in Figure 11(b), d_a and d_b are no longer forced to be put together. Therefore, d_a can be located in the low-density area of the chip.

We have also modified the overlapping rule of devices with our virtual-area concept by replacing the boundary variables in constraints (4)-(7) with virtual boundary variables. As shown in Figure 12, by prohibiting the overlapping among virtual areas instead of the exact device locations, we prevent the direct connection between parent and child devices from being obstructed by other devices.

Since the occupancy rate of chip area varies during an assay process according to the assay sequencing graph and the scheduling result, virtual boundaries can be applied especially when the occupancy rate of chip area is low so that the virtual area is not an issue for area competition among devices. As a conclusion, the introduction of virtual boundaries provides us more flexibility of locating devices and thus allows us to maximize the utilization of chip area and balance the valve actuations even further.

E. Assurance of Fluid Paths to Chip Boundaries

In order to transport waste, samples, reagents, and final products, devices need to be connected with chip ports. Since our valve-centered architecture is implemented in a matrix-shaped biochip [13], in which chip ports are all located at the chip boundaries, we propose a method to assure the fluid paths from a dynamic device to chip boundaries.

When a device lies in the inner part of a chip and is completely encircled by other devices as shown in Figure 13(a) after the dynamic device mapping, the path between a device and chip boundaries can be blocked, Therefore, we break the encirclement by adding a new objective in our ILP model to draw this certain device closer to the corner of the chip, and perform the dynamic device mapping again.

Before introducing the objective, we first define the distance between a device and chip corners. As shown in Figure 13(a), the distance between d_a and chip corners is decided by the horizontal and vertical distance between d_a and the chip boundaries. This can be introduced to our model by adding following constraints:

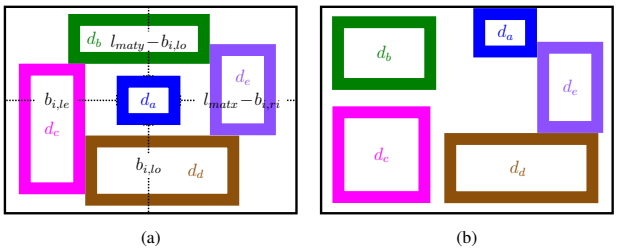
$$l_{i,1} \geq b_{i,le} - q_1 M, \tag{21}$$
$$l_{i,1} \geq l_{matx} - b_{i,ri} - q_2 M, \tag{22}$$
$$l_{i,2} \geq b_{i,lo} - q_3 M, \tag{23}$$
$$l_{i,2} \geq l_{maty} - b_{i,up} - q_4 M, \tag{24}$$
$$q_1 + q_2 = 1, \tag{25}$$
$$q_3 + q_4 = 1 \tag{26}$$

in which $l_{i,1}$, $l_{i,2}$ are the horizontal and vertical distance between a device and its nearest chip corner, l_{matx}, l_{maty} are the horizontal and vertical dimensions of the chip, q_1, q_2, q_3, q_4 are auxiliary binary variables and M is a very large constant.

When one of q_k, $k \in \{1,2,3,4\}$ is set to 1, the corresponding inequation becomes trivial. Taking Figure 13(a) as an example, the set of horizontal distances between d_a and chip boundaries is $\{b_{i,le}, l_{matx} - b_{i,ri}\}$, and the set of vertical distances between d_a and chip boundaries is $\{b_{i,lo}, l_{maty} - b_{i,up}\}$. Constraints (21)(22) ensure that we will choose exactly one value from each set to control the distance between d_a and its nearest chip corner. For the sake of model reduction, we represent this distance by $l_{i,1} + l_{i,2}$, instead of $\sqrt{l_{i,1}^2 + l_{i,2}^2}$ applying the Pythagorean theorem. Accordingly, we modify our optimization objective:

$$\text{Minimize: } w + \alpha f_i \times (l_{i,1} + l_{i,2}), \forall l_i \in S_d \tag{27}$$

where α is a constant coefficient, f_i is a weight factor which increases each time the connection path problem occurs to the i_{th} operation, and S_d is the set of operations whose device connection to chip boundaries are blocked.



(a) (b)

Figure 13: An example of fluid path assurance for d_a to chip boundaries: (a) d_a is freely placed. (b) d_a is closer to chip boundaries.

With this modification, as shown in Figure 13(b), we can obtain a new mapping result. Based on this different request, the shape and location of our dynamic devices are adjusted so that d_a is drawn near the upper right chip corner and no longer encircled by other devices, which assures the fluid path between d_a and chip boundaries and thus chip ports.

F. Valve-actuation-aware Routing

After the dynamic device mapping process, we route the fluid paths in the chip. Our routing method takes valve actuations caused by path routing into consideration and thus further reduces the largest number of valve actuations and the number of valves.

We apply Dijkstra's shortest path algorithm and construct the cost function according to valve actuations. In our valve-centered architecture, fluid paths can be divided to chambers formed by valves. As shown in Figure 14(a), A, B, C, and D are such chambers. By controlling the valves connected with these chambers, namely v_1, v_2, v_3, and v_4, we can control the direction of fluids and thus build different fluid paths.

Before we route a new fluid path, we record the number of current valve actuations of each valve and set it as the cost of this valve, and we set the initial cost of each chamber as infinity, as shown in Figure 14(b).

When the actuation of valve v is involved in forming a fluid path to a chamber C_H, the cost of v will be added with the cost of the last chamber that this path is constructed with. We define the sum of the costs as s and compare it with the cost of chamber C_H. If s is smaller, we update the cost of chamber C_H with s. Our target is to find the lowest cost of each chamber that we want to reach to from a starting point, and thus deciding the routing path with back-tracing. As shown in Figure 14(b), the cost of the starting sample port is set to 0. In order to reach chamber A, we add the cost of valve v_1 with 0 and get a new value 3, which is smaller than infinity, and thus replace infinity as the new cost of chamber A as shown in Figure 14(c).

In our example, v_4 has been used as pump valve for multiple times and thus actuated for 120 times. We suppose that 120 is exactly the largest number of valve actuations in the chip. On the other hand, v_2 has not been actuated yet and may be removed at the end of the entire synthesis. Therefore, when the actuation of v_4 and v_2 is involved in forming a routing

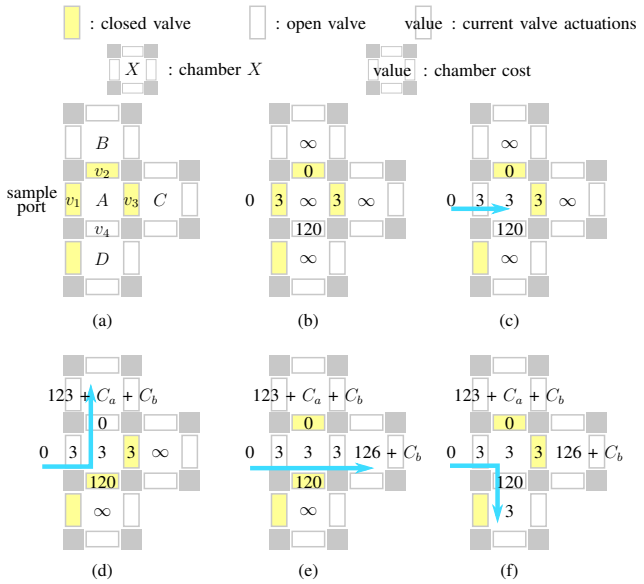


Figure 14: Update of chamber costs: (a) Target chambers. (b) Initial costs. (c) For chamber A. (d) For chamber B. (e) For chamber C. (f) For chamber D.

path, we will either worsen our former optimization result, or we will need to manufacture an extra new valve. In order to reduce these actuations, we set extra cost for actuating these valves. As shown in Figure 14(d), we set extra cost C_a to v_2 which is never used and extra cost C_b to v_4 which has the highest number of actuations. When we want to reach chamber B from chamber A, we need to actuate v_2 and v_4. Therefore, the cost of chamber B will be updated with the sum of costs of v_2, v_4, and A, which is $123+C_a+C_b$, since $123+C_a+C_b$ is less than infinity. Correspondingly, the new cost of chamber C is $126+C_b$ and the new cost of chamber D is 3, as shown in Figure 14(e)(f).

This cost function can be represented as:

$$\begin{aligned} \text{if} \quad & c_N < c_C + s_{v_i} a_{v_i} \\ & + (1-u_{v_i}) C_a + m_{v_i} C_b, \forall v_i \in S_C, \\ \text{then} \quad & c_N = c_C + s_{v_i} a_{v_i} \\ & + (1-u_{v_i}) C_a + m_{v_i} C_b, \forall v_i \in S_C \end{aligned} \tag{28}$$

where c_N is the cost of the to-be-reached chamber N, c_C is the cost of the last passed-by chamber C, s_{v_i} is a binary variable indicating whether the actuation of valve v_i is involved in forming a fluid path from C to N, a_{v_i} is the number of actuation of v_i, u_{v_i} is a binary variable indicating whether v_i has ever been actuated, m_{v_i} is a binary variable indicating whether v_i is the valve with the highest number of actuations, and S_C is a set containing the valves encircling chamber C.

According to the assay schedule, each time when we need a new routing path, we accurately record the current valve status to decide which valve actuations should be involved in forming this new routing path. Based on these information, we apply our above mentioned method to get a routing solution. But since the assay is in progress, a valve with currently fewer actuations may also serve as a frequently actuated pump valve later. Therefore, we apply a rip-up and reroute method for several iterations based on the former routing results to revise our routing solution.

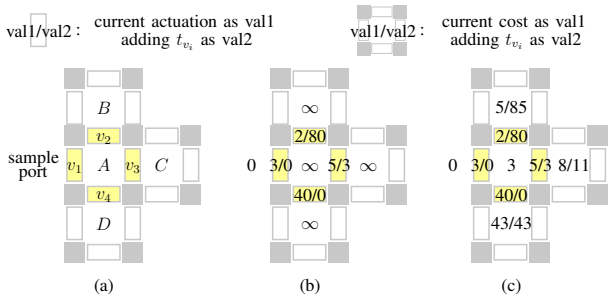


Figure 15: Chamber costs applying rip-up and reroute: (a) Target chambers. (b) Initial costs. (c) Updated chamber costs.

Suppose that we want to revise the costs of chambers A, B, C and D as shown in Figure 15(a). Valve v_1, v_2, v_3 and v_4 are currently actuated for 3, 2, 5 and 40 times respectively. As shown in Figure 15(b), we know from the former routing results that in the last iteration, from the current time till the end of the assay, v_2 will be actuated 80 more times. Therefore, we revise the cost of v_2 by adding 80 to it. Similarly, we also add the cost of v_3 with 3. We then get the new costs of chambers A, B, C, D as shown in Figure 15(c), which provides us a more comprehensive solution.

The revised cost function can be formulated as follows:

$$\begin{aligned} \text{if} \quad & c_n < c_c + s_{v_i} \times a_{v_i} \\ & + (1-u_{v_i}) C_a + m_{v_i} C_b \\ & + t_{v_i}, \forall v_i \in S_c, \\ \text{then} \quad & c_n = c_c + s_{v_i} \times a_{v_i} \\ & + (1-u_{v_i}) C_a + m_{v_i} C_b \\ & + t_{v_i}, \forall v_i \in S_c \end{aligned} \tag{29}$$

in which t_{v_i} indicates the extra actuations of v_i from the current time till the end of the assay in the last iteration. In this way, we maximize the utilization of existing valves with fewer actuations and thus also existing flow channels, which enables us to reduce the largest number of valve actuations and the sum of valves even further.

G. Overall Algorithm

Algorithm 1 gives an overall view of our methods. We index the lines as Li, $i \in \mathbb{N}$, at the beginning of each line. After reading the program input as shown in L1 and building the data structure as shown in L2, we perform our dynamic device mapping by using an ILP model as shown in L3-L12 and then decide the routing paths as shown in L13-L26.

After we get our first dynamic device mapping results, we perform an area check in L6-L8 and a fluid-path-assurance check in L9-L11 to support the reliability of our method:

Our valve-role-changing concept brings us more options for overlapping. Besides the overlapping permission for *in situ* on-chip storages and parent devices as mentioned in Section III-C, when a storage has enough free space, we also allow routing paths to pass through this storage as shown in Figure 16(b), thus saving the efforts for a long detour

Algorithm 1: Reliability-aware Synthesis

```
L1  Read sequencing graph and scheduling result.
L2  Build virtual valves in valve-centered architecture.
L3  # DynamicDeviceMapping
L4  repeat
L5      Build and solve ILP model for dynamic device mapping.
L6      if overlapping area of (storage s, device d) > free space of s then
L7          Forbid (s,d) from overlapping with each other.
L8      end
L9      if fluid paths from or to device d can not reach chip boundaries
        then
L10         Draw d closer to chip boundaries.
L11     end
L12 until feasible dynamic device mapping;
L13 # Routing
L14 for iteration ite = 1 to maxIte do
L15     Rip up all routed paths.
L16     for time t = 1 to maxT do
L17         forall the connections do
L18             Route a path with minimum cost.
L19             if overlapping area of (storage s, path p) > free space
                of s then
L20                 Forbid (s,p) from overlapping with each other.
L21                 Rip up p and reroute.
L22             end
L23         end
L24         Record the numbers of valve actuations.
L25     end
L26 end
L27 Remove non-actuated valves.
```

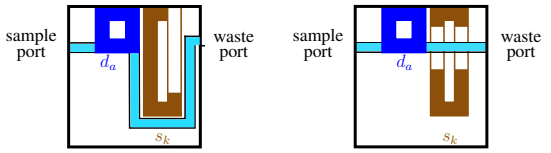


Figure 16: (a) The storage s_k is an obstacle for routing paths. (b) The storage s_k can be passed through by routing paths.

as shown in Figure 16(a). To make sure that overlapping only happens on the premise of enough free storage-space, we perform an area check as shown in L6-L8 and L19-L22. We then perform a fluid-path-assurance check as mentioned in Section III-E as shown in L9-L11.

We route the fluid paths after dynamic device mapping. The valve-actuation-aware routing results will be revised by a rip-up and reroute method for several iterations as shown in L14-L26.

IV. 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 [19]. The device library applied in this work is shown in Table I, where *volume* indicates the number of chambers occupied by a device, *dimension* indicates the number of chambers in horizontal and vertical directions of this device, and *ratio* indicates the input ratios that are supported by the device.

In our method, we assume that mixing operations with the same input volume and ratios have the same duration regardless of the mixer dimensions, and this duration indicates the maximum duration in mixers of all dimensions. For example, suppose that a mixing operation o_a can be executed in either mixer m_1 or mixer m_2 (m_1 and m_2 only differ in dimensions), the execution time of o_a in m_1 is t_1 and the execution time of o_a in m_2 is t_2. If $t_1 > t_2$, we will specify t_1 as the duration of o_a in our method, regardless of whether

Table I: Library of devices used in this work.

Volume	4	6	8	8	10	10
Dimension	2×2	2×3	2×4	3×3	2×5	3×4
Ratio	1 : 1	1 : 2	1 : 1, 1 : 3	1 : 1, 1 : 3	1 : 4, 2 : 3	1 : 4, 2 : 3

m_1 or m_2 is finally in use. The proposed method provides a conservative execution of operations in different mixers, and it can be extended easily to handle different execution durations by describing the execution time of an operation in different mixers with a lookup table.

We take four test cases from widely used laboratory protocols [20] [21]. 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 new method under two different settings with the results of the optimal binding for the traditional designs and with the results from our preliminary work [17] in Table II, 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, with hyphens separating mixers of different sizes.
vs_t_{max}**:** the largest number of valve actuations applying the optimal binding for the traditional designs.
vs_{max}**:** the largest number of valve actuations and actuations for peristalsis thereof applying our methods.
$\#v$**:** the sum of used valves.
T**:** the program runtime.

In Table II, column 8-10 show the results of applying the method in our preliminary work under conservative setting, column 11-13 show the results of applying the new method under conservative setting, column 14-15 show the results of applying the method in our preliminary work under aggressive setting, and column 16-18 show the results of applying the new method under aggressive setting.

In the traditional designs, we assume there are 4 different sizes of mixers: 4, 6, 8, and 10. The 4-unit mixers with two ports can support 1:1 mixing operations, the 6-unit mixers with two ports can support 1:2 mixing operations, the 8-unit mixers with three ports can support 1:1 as well as 1:3 mixing operations, and the 10-unit mixers with three ports can support 2:3 as well as 1:4 mixing operations. 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 actuation 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. For example, as shown in Table II, in test case PCR policy 1, there are 3 mixers with different sizes. 1 mixing operation

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

			Optimal Binding for Traditional Designs				[17] Cons.			New Method Cons.			[17] Aggr.		New Method Aggr.		
Column Index	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	$\#op$	$Po.$	$\#d$	$\#m_{4-6-8-10}$	vs_t_{max}	$\#v$	vs_{max}	$\#v$	T	vs_{max}	$\#v$	T	vs_{max}	$\#v$	vs_{max}	$\#v$	T
PCR	15(7)	p1	3	1-0-4-2	160	83	45(40)	71	0.8	42(40)	77	0.8	35(30)	71	31(30)	62	0.9
		p2	4	1-0-(2,2)-2	80	99	45(40)	76	0.8	44(40)	71	0.7	34(30)	76	36(30)	62	0.8
		p3	6	1-0-(2,1,1)-(1,1)	80	131	43(40)	82	0.9	44(40)	82	0.9	31(30)	82	32(30)	76	1.1
Mixing Tree	37(18)	p1	4	2-4-5-7	280	108	93(80)	105	2.9	87(80)	109	2.4	46(42)	105	35(30)	105	94
		p2	5	2-4-5-(4,3)	200	124	93(80)	105	2.9	87(80)	109	2.3	46(42)	105	35(30)	105	94
		p3	6	2-4-(3,2)-(4,3)	160	140	90(80)	124	3.3	90(80)	122	3.3	60(50)	124	38(30)	120	21.5
Interpolating Dilution	71(35)	p1	7	5-9-9-(6,6)	360	178	145(120)	176	357.1	132(120)	163	30.1	72(65)	176	62(42)	173	0.5h
		p2	9	5-(5,4)-(5,4)-(6,6)	240	207	94(80)	207	87.8	94(80)	207	20.7	56(42)	207	38(32)	206	1h
		p3	10	5-(5,4)-(5,4)-(4,4,4)	200	225	92(80)	208	101.2	90(80)	206	108	56(50)	208	47(35)	209	1.5h
Exponential Dilution	103(47)	p1	10	6-(8,8)-(7,6)-(6,6)	320	241	135(120)	214	485.3	101(80)	224	774.6	75(75)	214	55(50)	216	0.5h
		p2	11	6-(6,5,5)-(7,6)-(6,6)	280	254	134(120)	255	488.9	103(80)	254	858	71(65)	255	48(40)	261	1h
		p3	12	6-(6,5,5)-(5,4,4)-(6,6)	240	268	99(80)	259	314.3	93(80)	259	957.6	58(40)	259	47(40)	253	1.5h

is bound to the 4-unit mixer, 4 mixing operations are bound to the 8-unit mixer, and 2 mixing operations are bound to the 10-unit mixer. Hence we add one more 8-unit mixer in policy 2, so that in the result of the optimal binding the 4 mixing operations can be evenly assigned to the two 8-unit mixers as 2 operations per mixer.

In our methods, we first built a square matrix containing virtual valves based on the valve-centered architecture. In this matrix, the number of virtual valves is larger than 1.5 times the number of valves used in the traditional method, and the total fluid volume of the matrix is larger than 2 times the highest total fluid volume of the operations that simultaneously work in the chip. This setting is arbitrary, but not harmful to the number of valves implemented at the end, because the non-actuated virtual valves are removed after the synthesis.

After constructing the matrix, we built and solved the model for dynamic device mapping and routed the sample paths. We calculated the largest numbers of valve actuations in vs_{max} in Table II, which are close to the numbers of actuations for peristalsis thereof. This fact validates our method in Section III-B, where we only model actuation activities for peristalsis.

In our model, 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 7(b) uses 8 pump valves, while the dedicated mixer as shown in Figure 3(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 actuations are sufficient for a single mixing operation. Therefore, we provide both a conservative setting and an aggressive setting for comparison with the traditional method.

Under our first setting we still assume that each pump valve is actuated 40 times for a single mixing operation, which is exactly the same as the setting for pump valve working in a dedicated mixer in the traditional method as a conservative comparison. vs_{max} in column 8 and column 11 show that even under this conservative setting, we still reduce the largest numbers of actuations by more than 50% compared with traditional method. By contrast, under our aggressive setting, we assume that the sum of actuations for peristalsis of a mixer

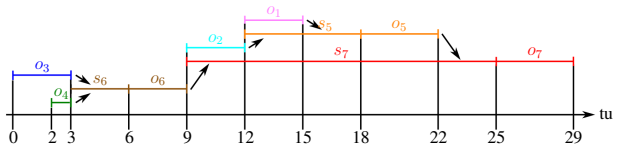


Figure 17: The scheduling result of case PCR in p1.

is the same as that in the traditional method. For example, the sum of valve actuations for peristalsis of a dedicated mixer to perform a single mixing operation in a traditional design is 3×40=120, so we change the number of actuations of each valve in our dynamic mixer using 8 pump valves to 15 since 8×15=120. As shown in vs_{max} in column 14 and column 16, the results are much better, even with a small number of valves shown in $\#v$.

Compared with our preliminary work, we propose three major improvements in this work:

1. Routing-convenient mapping with virtual boundary mentioned in Section III-D.
2. Assurance of fluid paths to chip boundaries mentioned in Section III-E.
3. Valve-actuation-aware routing applying rip-up and reroute method mentioned in Section III-F.

which bring about better solutions in valve actuation as well as the sum of valves, and enhance the reliability of our method.

The routing-convenient mapping provides our model more flexibility in generating devices, while their connections are guaranteed with virtual boundaries and virtual areas of devices. For large designs that provide more options to the locations of devices, devices can be generated in best places when their locations are not strongly limited by the locations of their parent and child devices. Since the best locations for devices performing sequential operations may be far apart from each other, to route the possible long connections between them, our new valve-actuation-aware routing method shows its benefit. In our new routing method, we model corresponding valve actuations accurately for accessing every chamber in the chip, and we further revise the results by rip-up and reroute. Compared with the routing in our preliminary work where we route fluid paths in the shortest length, in the new method a longer path may be preferred if valve actuations led by routing this path can be reduced. As shown in Table II, we achieve

results with noticeable improvements for vs_{max} in column 8 and column 11 as well as for $\#v$ in column 9 and column 12. For example, compared with our preliminary work, for test case Exponential Dilution in policy 1, we further reduce the largest number of valve actuations under both conservative and aggressive settings by more than 25%.

In our preliminary work [17], we only performed optimization under conservative setting, and the results under aggressive setting were directly derived from the results under conservative setting, where the program runtimes under aggressive setting approximate 0 and we thus omitted them in the table. In this work, we perform optimization for each test case both under conservative and aggressive setting.

In general, the program runtime denoted as T in Table II is not very stable, which strongly depends on the heuristics the optimization solver selects. However, a trend can still be observed that the program runtime typically exponentially increases with the problem size, which is mainly caused by the ILP modeling method that we apply. Though the scalibility is usually considered as an issue for ILP modeling method, it is not a serious problem for this work, since the matrix size of the general purpose architecture cannot be unlimitedly enlarged.

To show the working principles of our method intuitively, we take the synthesis result of case PCR with 7 mixing operations in policy 1 as an example. The input of our method is the scheduling result of this case with 3 time-units (tu) as the transport delay. We show the scheduling result in Figure 17 and the synthesis result in Figure 18, in which o_1 and o_2 are the parent operations of o_5, o_3 and o_4 are the parent operations of o_6, and o_5, o_6 are the parent operations of o_7.

In Figure 18, closed valves are drawn in light color. The valves that are never actuated are removed and the area is left empty, just like the two valves at the top-right corner of the chip. In addition, if valves are only actuated once and no fluid flows through them while they are open, they are removed as well and we build functionless walls drawn in dark color at the areas those valves once occupied. The numbers of valve actuations at every time moment are directly labeled on the corresponding valves. Fluid paths are represented by lines and their directions are indicated by the arrows on the lines. If two or more paths come from or go to the same region, the paths routed earlier are drawn in dashed lines, and the paths routed later are drawn in solid lines. Since not all of the products would go to next devices but some of them also would go to waste sink, for each dynamic device finishes its job we route a fluid path for it to the waste port. Unlike the fluid paths from input ports, fluid paths to waste port are drawn in dark color.

At t = 0tu as shown in Figure 18(a), o_3 starts and takes sample and reagent as inputs from port 1 and 2. The input from port 2 comes first, and is followed by the input from port 1.

Since o_3 is a mixing operation with a volume as 8 units, it occupies a 2×4 area in the chip, in which 2 internal valves are closed as internal boundaries of the mixer, and the other 8 internal valves are actuated for 40 times as pump valves to produce a circulation flow.

At t = 2tu as shown in Figure 18(b), o_4 starts and the dynamic device mapped by it is located adjacent to the device mapped by o_3. These two devices share the same closed valves as their outer boundaries. The input of o_4 from port 1 comes first this time, and the valve connected to port 1 at the chip boundary is closed when routing the fluid path from port 2.

At t = 3tu, storage s_6 is constructed immediately after o_3 and o_4 are finished as shown in Figure 18(c), which stores the products of o_3 and o_4. Some products of o_3 and o_4 go to their next device s_6, but some of them also go to waste sink. Note that the storage mapped by s_6 is placed in a distance from o_3 and o_4. In the method in our preliminary work, this situation can not happen since the child devices are forced to be placed adjacent to their parent devices to prevent other devices from being inserted between them. With the concept of virtual boundaries, the virtual area of s_6 can be larger than the area it really occupies, and is adjacent to the mixers mapped by o_3 and o_4. Since the overlapping between virtual areas is prohibited, no device can be inserted between the devices mapped by o_3 and s_6 as well as o_4 and s_6, so that the connections between them can be guaranteed and directly constructed.

At t = 6tu s_6 changes into a device mapped by o_6 and starts to work as shown in Figure 18(d), which ends at t = 9tu as shown in Figure 18(e). Some of product of o_6 goes to s_7, and some other goes to the waste port. This fluid path to the waste port is not the shortest one in distance from the device mapped by o_6, but has the lowest cost according to our cost function (29). At the same time, o_2 starts after receiving inputs from port 1 and port 2.

At t = 12tu o_2 ends and sends its product to s_5 as well as to the waste port as shown in Figure 18(f). Also, o_1 starts while the device performing o_1 occupies a chamber of the storage for s_5 with the prerequisite that the remaining free area inside the storage is still enough to store its input from o_2. In our test case, according to the information from the sequencing graph of the bioassay, only 2 volume-unit product of o_5 comes from o_2 and 8 volume-unit product comes from o_1, thus the overlapping of 1 volume-unit chamber between the device mapped by o_1 and s_5 is allowed, since the storage mapped by s_5 only contains 2 volume-unit product from o_2 for the time being, and the feasibility check mentioned in Section III-G can pass.

In Figure 18(f), though storage s_5 overlaps with one of its parent device mapped by o_1, it is placed far away from its another parent device mapped by o_2, and their direct connection is blocked by the storage for s_7. In this new work, this situation can happen since the device mapped by o_7 is the child device of the device mapped by o_5, thus the overlapping of their virtual areas is allowed when performing dynamic device mapping. However, when performing feasibility check for overlapping area among devices, we use the real boundaries of devices so that the check passes since the actual area of the device mapped by o_7 do not overlap with the actual area of the device mapped by o_5. Consequently, storage s_5 can be placed far from the device mapped by o_2, and storage s_7 can be placed in between them.

Though s_7 seems to obstruct the direct connection from the device mapped by o_2 to the device mapped by o_5, device like s_7 must be a storage and is not a blockage in most cases

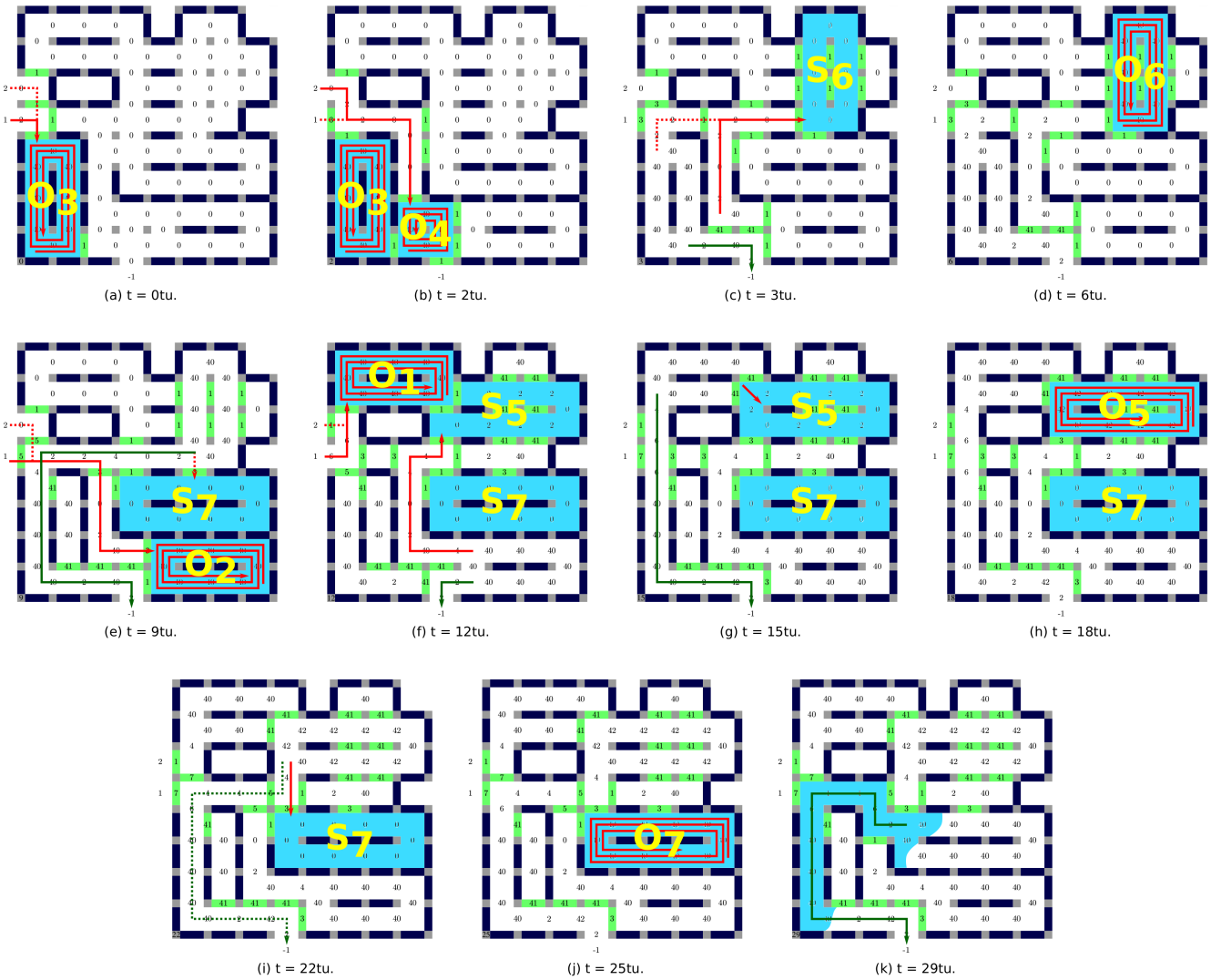


Figure 18: Snapshots of the synthesis result of test case PCR in policy 1 under conservative setting.

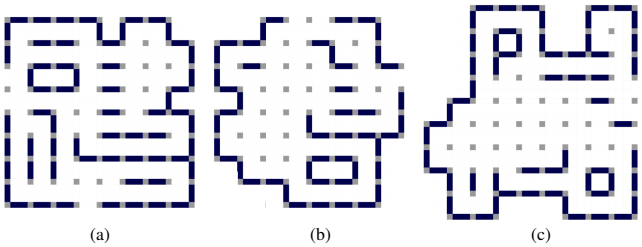


Figure 19: Different designs of PCR in different policies under conservative setting: (a) Policy 1. (b) Policy 2. (3) Policy 3.

since one of its parent devices, e.g. the device mapped by o_5 in this case, just starts to work after receiving inputs from this direction connection. In this case, though the valve-actuation-aware routing method eventually routes this connection as a detour since this routing leads to the minimum valve actuation according to our cost function, a straight connection is still available, as s_7 only contains 2 volume-unit input from o_6 according to our test case and has 8 free volume units left that can be used for overlapping.

Finally, at t = 25tu as shown in Figure 18(j), At t = 15tu as shown in Figure 18(g), o_1 ends and sends its product to s_5 and the waste port. Then o_5 starts to work at t = 18tu as shown in Figure 18(h). After o_5 ends at t = 22tu as shown in Figure 18(i), it sends its product to s_7 as well as to the waste port. Note that the fluid path from the device mapped by o_5 to the waste port has a similar shape with the paths from o_3 to s_6 in Figure 18(c), from o_3 as well as o_4 to the waste port in Figure 18(c), from o_6 to the waste port in Figure 18(e), and from o_1 to the waste port in Figure 18(g), because our rip-up and reroute method tend to minimize valve actuation regarding the current status of valves. Since these valves are used to form fluid paths or flow channels in devices at early time moments, they tend to be chosen to form flow channels for fluid paths as only few valve actuations are needed.

the last operation o_7 starts, and it ends at t = 29tu as shown in Figure 18(k). We assume that for all test cases, their final products leave the chip from the only output port, which is also regarded as the waste port. Note that the fluid path from the device mapped by o_7 to the waste port is routed in the same manner again as for other devices.

Figure 19 shows different designs of PCR in different

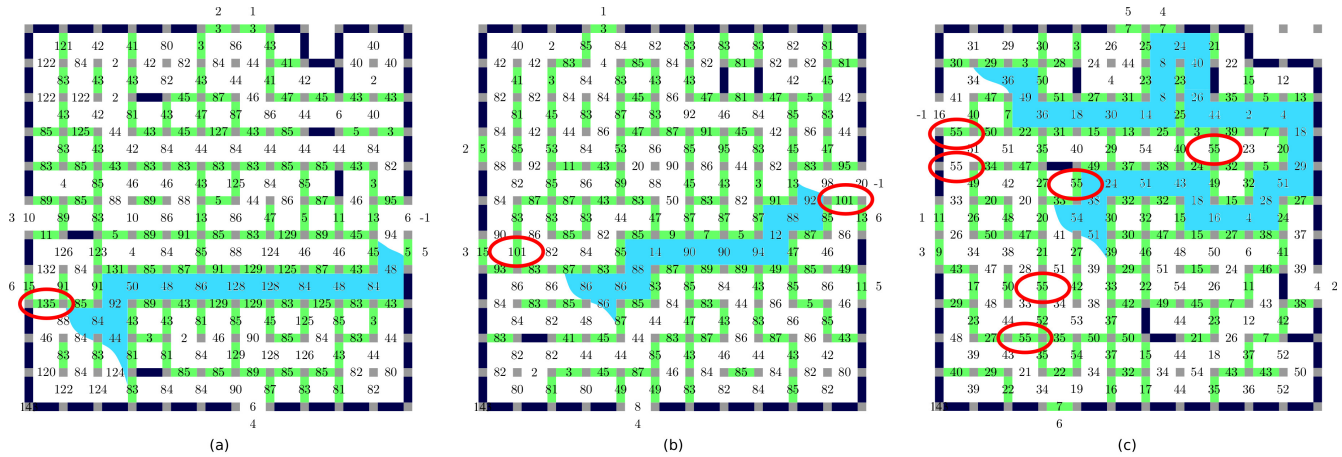


Figure 20: Different synthesis results of test case Exponential Dilution at t = 143tu by different methods: (a) With the method in [17]. (b) With the new method under conservative setting. (c) With the new method under aggressive setting.

policies under conservative setting, where designs in Figure 19(a) and Figure 19(b) are derived from a 8×8 valve matrix, and the design in Figure 19(c) is derived from a 9×9 valve matrix. These designs look very different from each other since compared with the sum of valves actually needed to implement the designs, the sum of virtual valves in these matrices are more than sufficient, and the synthesis results have shown some characteristics about the bioassays.

Figure 20 shows the performance of our method on relatively small virtual-valve matrices for test case Exponential Dilution by using different methods or under different settings, in which the status of valves and the sum of valve actuations show the final moment of the chip that it should be after finishing the whole bioassay at t = 143tu. Figure 20(a) shows the result of applying the method in our preliminary work. Since each time when a valve plays the role as pump valve, it needs to be actuated for 40 times under our setting. In Figure 20(a) there are several valves playing the role as pump valve for 3 times, while there are also several valves playing the role as pump valve for 0 times. For these valves, the difference in valve actuation can be larger than 120, and thus leads to a remarkable imbalance. This imbalance is significantly alleviated by our new comprehensive method. In Figure 20(b), the result is greatly improved as most valves play pump valve for 2 times, and the largest number of actuations decrease from 135 to 101.

Figure 20(c) shows the result of applying the new method under aggressive setting, by which valve actuations are further decreased. Note that in Figure 20(a), there is only one valve under the heaviest loading and is needed to be actuated for 135 times. We circle out this certain valve in Figure 20(a). But in Figure 20(b) there are 2 valves actuated most for 101 times, and in Figure 20(c) there are 6 valves actuated for 55 times under the heaviest loading. This means that compared with our preliminary work, valve actuations in our new work are not only decreased but also distributed evenly.

Figure 20(c) also demonstrates that a fluid path with the minimum valve actuations may not be the shortest one. In Figure 20(c), a long detour is chosen for forming the fluid path to transport the final product of a detector to the output port. Though this path is long, it takes advantage of existing flow channels and thus contributes to a better solution.

V. Conclusion

In this paper we have addressed a reliability problem of flow-based biochips due to unbalanced valve actuations. The problem is solved by the proposed reliability-aware synthesis including two steps as dynamic device mapping and fluid path routing based on a virtual valve-centered architecture with valve-role-changing concept. Compared with our preliminary work, we have revised the routing-convenient device mapping, assured fluid paths to chip boundaries, and proposed a valve-actuation-aware routing for fluid paths. Experimental results show that this new work outperform our preliminary work especially when the designs are large.

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*, vol. 442, pp. 412–418, 2006.

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

[3] Life Technologies (Thermo Fisher Scientific), "Ribolock rnase inhibitor," 2015. [Online]. Available: https://www.lifetechnologies.com

[4] Qiagen, Inc., "QIAGEN RNase inhibitor," 2015. [Online]. Available: http://www.qiagen.com

[5] D. Langemann, "The free transmission problem for a water droplet," *Large-Scale Scientific Computing*, vol. 2907, pp. 387–395, 2004.

[6] S. Roy, S. Kumar, P. P. Chakrabarti, B. B. Bhattacharya, and K. Chakrabarty, "Demand-driven mixture preparation and droplet streaming using digital microfluidic biochips," in *Proc. Design Autom. Conf.*, 2014, pp. 1–6.

[7] W. Thies, J. P. Urbanski, T. Thorsen, and S. Amarasinghe, "Abstraction layers for scalable microfluidic biocomputing," *Natural Comput.*, vol. 7, no. 2, pp. 255–275, 2008.

[8] S. Roy, B. B. Bhattacharya, S. Ghoshal, and K. Chakrabarty, "Low-cost dilution engine for sample preparation in digital microfluidic biochips," in *Proc. Int. Symp. Electron. Syst. Des.*, 2012, pp. 203–207.

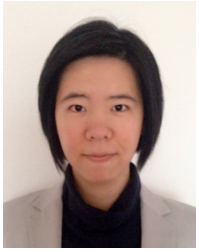
[9] S. Bhattacharjee, A. Banerjee, and B. B. Bhattacharya, "Sample preparation with multiple dilutions on digital microfluidic biochips," *IET Compute. & Digital Tech*, vol. 8, no. 1, pp. 49–58, 2014.

[10] 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*, 2010, pp. 91–98.

[11] 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.*, 2011, pp. 225–234.
[12] J. S. McCaskill and P. Wagler, "From reconfigurability to evolution in construction systems: Spanning the electronic, microfluidic and biomolecular domains," in *Proc. Int. Conf. Field-Programmable Logic and Applications*, 2000, pp. 286–299.
[13] L. M. Fidalgo and S. J. Maerkl, "A software-programmable microfluidic device for automated biology," *Lab on a Chip*, vol. 11, pp. 1612–1619, 2011.
[14] A. M. Amin, M. Thottethodi, T. Vijaykumar, S. Werely, and S. C. Jacobson, "Aquacore: a programmable architecture for microfluidics," 2007, pp. 254–265.
[15] 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.*, 2013, pp. 213–218.
[16] 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.*, 2013, pp. 123–129.
[17] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proc. Design Autom. Conf.*, 2015, pp. 141:1–141:6.
[18] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake, "Monolithic microfabricated valves and pumps by multilayer soft lithography," *Science*, vol. 288, no. 5463, pp. 113–116, 2000.
[19] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*. http://www.gurobi.com.
[20] 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*, 2003, pp. 619–622.
[21] K. Chakrabarty and F. Su, *Digital Microfluidic Biochips: Synthesis, Testing, and Reconfiguration Techniques*. Boca Raton, FL: CRC Press, 2006.

Tsun-Ming Tseng received the bachelor degree in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2010, and the master degree in communications engineering from Technische Universität München (TUM), Munich, Germany, in 2013. He is currently pursuing a Dr.-Ing. degree at the Institute for Electronic Design Automation, TUM. His research interests focus on mathematical methods for computer-aided design.

Bing Li received the bachelor's and master's degrees in communication and information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2000 and 2003, respectively, and the Dr.-Ing. degree in electrical engineering from Technische Universität München (TUM), Munich, Germany, in 2010. He is currently a researcher with the Institute for Electronic Design Automation, TUM. His research interests include timing and power analysis and emerging systems.



Mengchu Li received the bachelor degree in Germanistik from Tongji University in China. She then comes to Germany and is currently pursuing another Bachelor degree in Computer Science at Ludwig-Maximilians-Universität München (LMU) in Germany. She works as working student for the Institute for Electronic Design Automation in Technische Universität München (TUM) in Germany. Her current research interests focus on design automation for continuous-flow microfluidics.

Tsung-Yi Ho (SM'12) received his Ph.D. in Electrical Engineering from National Taiwan University in 2005. He is a Professor with the Department of Computer Science of National Tsing Hua University, Hsinchu, Taiwan. His research interests include design automation and test for microfluidic biochips and nanometer integrated circuits. He has presented 9 tutorials and contributed 9 special sessions in ACM/IEEE conferences, all in design automation for microfluidic biochips. He has been the recipient of the Invitational Fellowship of the Japan Society for the Promotion of Science (JSPS), the Humboldt Research Fellowship by the Alexander von Humboldt Foundation, and the Hans Fischer Fellow by the Institute of Advanced Study of Technische Universität München (TUM). He was a recipient of the Best Paper Awards at the VLSI Test Symposium (VTS) in 2013 and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems in 2015. He served as a Distinguished Visitor of the IEEE Computer Society for 2013-2015, the Chair of the IEEE Computer Society Tainan Chapter for 2013-2015, and the Chair of the ACM SIGDA Taiwan Chapter for 2014-2015. Currently he serves as an ACM Distinguished Speaker, a Distinguished Lecturer of the IEEE CAS Society, and Associate Editor of the ACM Journal on Emerging Technologies in Computing Systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, and IEEE Transactions on Very Large Scale Integration Systems, Guest Editor of IEEE Design & Test of Computers, and the Technical Program Committees of major conferences, including DAC, ICCAD, DATE, ASP-DAC, ISPD, ICCD, etc.

Ulf Schlichtmann (S'88–M'90) received the Dipl.Ing. and Dr.Ing. degrees in electrical engineering and information technology from Technische Universität München (TUM), Munich, Germany, in 1990 and 1995, respectively. He was with Siemens AG, Munich, and Infineon Technologies AG, Munich, from 1994 to 2003, where he held various technical and management positions in design automation, design libraries, IP reuse, and product development. He has been with TUM as a Professor and the Head of the Institute for Electronic Design Automation, since 2003. He served as the Dean of the Department of Electrical Engineering and Information Technology, TUM, from 2008 to 2011. His current research interests include computer-aided design of electronic circuits and systems, with an emphasis on designing reliable and robust systems.