PathDriver: A Path-Driven Architectural Synthesis Flow for Continuous-Flow Microfluidic Biochips

Xing Huang[1], Youlin Pan[2], Grace Li Zhang[1], Bing Li[1], Wenzhong Guo[2*], Tsung-Yi Ho[3], and Ulf Schlichtmann[1]
[1]Chair of Electronic Design Automation, Technical University of Munich, Munich, Germany
[2]College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
[3]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
xing.huang1010@gmail.com, panyoulin@163.com, {grace-li.zhang, b.li}@tum.de, guowenzhong@fzu.edu.cn,
tyho@cs.nthu.edu.tw, ulf.schlichtmann@tum.de

ABSTRACT

Continuous-flow microfluidic biochips have attracted high research interest over the past years. Inside such a chip, fluid samples of milliliter volumes are efficiently transported between devices (e.g., mixers, etc.) to automatically perform various laboratory procedures in biology and biochemistry. Each transportation task, however, requires an exclusive flow path composed of multiple contiguous microchannels during its execution period. Excess/waste fluids, in the meantime, should be discarded by independent flow paths connected to waste ports. All these paths are etched in a very tiny chip area using multilayer soft lithography and driven by flow ports connecting with external pressure sources, forming a highly integrated chip architecture that dominates the performance of biochips. In this paper, we propose a practical synthesis flow called *PathDriver* for the design automation of microfluidic biochips, integrating the actual fluid manipulations into both high-level synthesis and physical design, which has never been considered in prior work. Given the protocols of biochemical applications, PathDriver aims to generate highly efficient chip architectures with a flow-path network that enables the manipulation of actual fluid transportation and removal. Additionally, fluid volume management between devices and flow-path minimization are realized for the first time, thus ensuring the correctness of assay outcomes while reducing the complexity of chip architectures. Experimental results on multiple benchmarks demonstrate the effectiveness of the proposed synthesis flow.

1 INTRODUCTION

Advances in continuous-flow microfluidic biochips have led to the emergence of "lab-on-a-chip" devices for automating laboratory procedures in biology and biochemistry. On such a micrometerscale platform, various biochemical applications such as protein crystallization [1], nanoparticle synthesis [2], etc., can be performed automatically with the advantages of high efficiency and high precision [3]. Accordingly, the global microfluidics market is estimated to be valued at USD 12,380 million by 2025 at an annual growth rate of 11.6% [4].

Structurally, a biochip is composed of two elastomer layers as shown in Fig. 1(a) [5], where each layer consists of a set of micro-scale channels. Microchannels in the flow layer, also called flow channels, are connected to external pressure sources through flow ports to transport samples and reagents. Microchannels in the control layer, also called control channels, are connected to control ports to conduct air pressure, so that the membrane in the overlapping area of the two channels can be pushed down into the flow channel to block the transportation of fluids, essentially forming a valve as illustrated in Fig. 1(b).

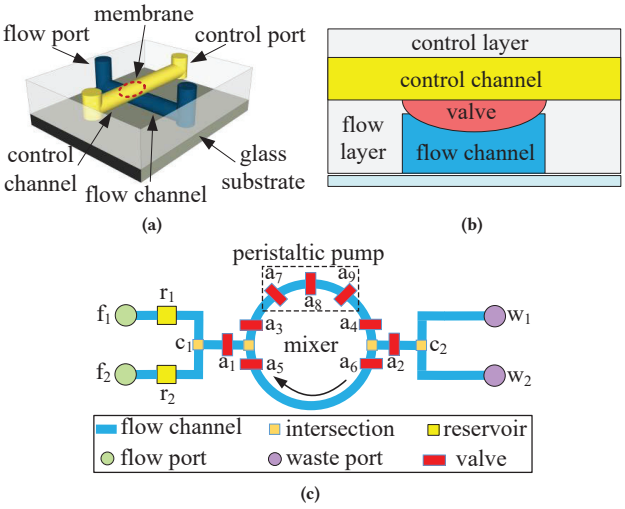

Figure 1: (a) Schematic of a biochip, (b) front view of (a), and (c) a part of the layout of a cell-culturing biochip [9].

With valves as the basic control units, complex microfluidic devices such as switches and mixers can be constructed [6], and biochemical applications can be executed automatically with a pre-customized assay plan [7, 8]. Fig. 1(c) shows a part of the layout of a cell-culturing biochip, where the reaction loop at the center is a mixer constructed with 9 valves [9]. Valves a_1–a_6 are used for directing the loading of fluid samples. For example, by closing a_5–a_6 and opening other valves, a sample can be loaded into the upper half of the mixer. Once the loading of samples is completed, the mixer can be sealed by closing a_1 and a_2 and the peristaltic valves a_7–a_9 are actuated sequentially to force the liquid inside to rotate.

When executing bioassays on a biochip, fluid samples and reagents need to be transported between devices to execute various biochemical operations. Each transportation task, however, requires an exclusive flow path during its execution period. Specifically, to drive the movement of fluid flow, a flow port that connects to an external pressure source should be placed at the start of the path to conduct thrust. Furthermore, since the initial status of a channel/device is not a vacuum

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-6654-2324-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415725

but filled with air, to avoid channel blockage, a waste port needs to be connected to the end of the path to release air pressures [10]. As a consequence, the flow port, the source and target devices, the waste port, as well as the flow channels connecting them form a complete flow path of the transportation task. For example, when transporting the reagent stored in reservoir r_1 to the mixer in Fig. 1(c), a valid flow path should be $\lceil f_1 \rightarrow r_1 \rightarrow c_1 \rightarrow \text{mixer} \rightarrow c_2 \rightarrow w_1 \rfloor$.

Besides fluid transportation, fluid samples that are not needed anymore during the execution of bioassays should be removed to waste ports by independent flow paths, including the following two categories: 1) excess fluids left in flow channels after loading samples into devices and 2) waste fluids that occupy devices after completing the execution of biochemical operations. Since fluid transportation and removal need to be performed frequently when executing bioassays [6], this leads to a large number of flow paths that need to be planned and constructed during chip design. These paths intersect with each other within a very small region, forming a highly integrated chip architecture that dominates the performance of the biochip.

In recent years, a number of methods have been proposed to deal with the architecture design of biochips [11–16]. In [11] a top-down design flow is proposed to generate an optimized chip architecture while minimizing the completion time of bioassays. In [12] physical design of biochips is investigated using a sequence-pair-based method, and it is formulated as a SAT problem in [13] to generate a close-to-optimal solution. In [14] a synthesis flow is proposed to generate a chip architecture under strictly constrained control ports. In [15] a co-layout synthesis method is proposed to minimize the total cost of chip architectures. Moreover, in [16] a timing-aware flow-channel routing method is proposed to optimize the timing behaviors of biochips. The aforementioned methods, however, do not take the actual manipulation of fluid transportation/removal into account and most of them assume that fluidic ports, including both flow port and waste port, are implicitly connected to flow channels, the vast flexibility in flow-path planning has so far been ignored during architecture design.

To fundamentally solve the problems above, in this paper, we propose *PathDriver*, a practical path-driven synthesis flow for microfluidic biochips. Our contributions include:

- We propose the first synthesis flow that takes actual fluid manipulations into account for the design automation of microfluidic biochips. With the proposed method, a practical biochip architecture with high efficiency and low cost can be generated automatically.
- We take both fluid volume management and excess/waste fluid removal into consideration during the high-level synthesis of biochips, so that biochemical operations can be executed accurately and efficiently, thus ensuring the correctness of assay outcomes with minimized completion time.
- We integrate flow-path planning into the physical design of biochips for the first time, so that the complete flow paths that enable the manipulation of actual fluid transportation and removal can be constructed automatically.
- We evaluate the proposed method using three real-life bioassays and three synthetic benchmarks. Experimental results demonstrate the effectiveness of the proposed synthesis flow.

The remainder of this paper is organized as follows. In Section 2, we briefly introduce the common synthesis flow of microfluidic biochips and analyze the design challenges in path-driven architectural synthesis. Section 3 discusses the proposed synthesis flow in detail. Experimental results are shown in Section 4 to demonstrate the effectiveness of PathDriver, and conclusions are drawn in Section 5.

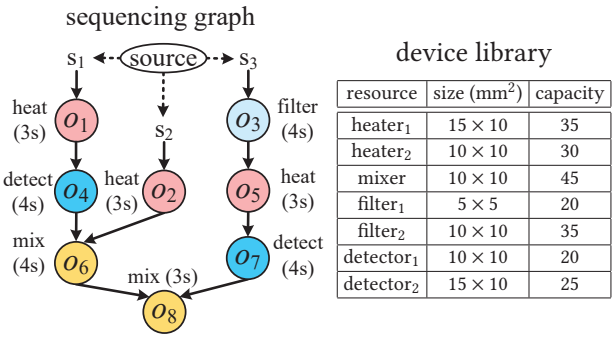

Figure 2: Sequencing graph of a bioassay and the given device library.

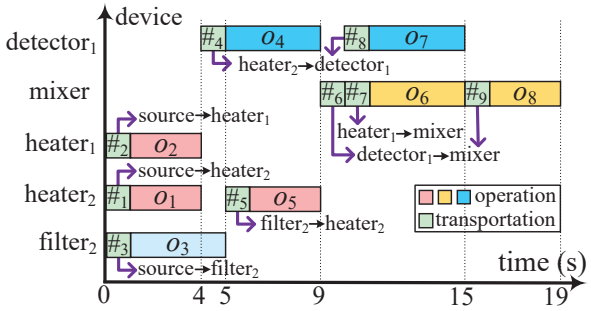

Figure 3: Scheduling and binding scheme corresponding to the bioassay described in Fig. 2.

2 DESIGN FLOW AND CHALLENGES

2.1 Design Flow of Microfluidic Biochips

The protocol of a bioassay is usually modelled as a directed acyclic sequencing graph $G(O, E)$. A vertex $o_i \in O$ represents a biochemical operation, e.g., mixing and heating, and is associated with a parameter indicating the corresponding execution time. An edge in G defines the dependency between operations, i.e., $e_{i,j} \in E$ represents that the output fluid of o_i is an input of o_j. For example, Fig. 2 shows the sequencing graph of a bioassay, which takes 3 fluid samples (s_1–s_3) as inputs and processes them with 8 biochemical operations (o_1–o_8). To realize the execution of these operations, a device library D is provided, where each device $d_i \in D$ is represented by a rectilinear box with input/output port on its boundary [12–15]. Moreover, each device is associated with a parameter indicating its capacity in milliliter. In Fig. 2, there are 7 devices available for realizing the bioassay above.

Given the sequencing graph of a bioassay and the corresponding device library, the architectural synthesis of microfluidic biochips is usually divided into two major stages, high-level synthesis and physical design.

In the first stage, the high-level synthesis aims to realize the following goals: 1) find a binding function $\phi : O \rightarrow D$ such that each operation is bound to a specific device to execute and 2) generate a scheduling scheme such that all the operations are executed satisfying the given dependencies while minimizing the completion time of the

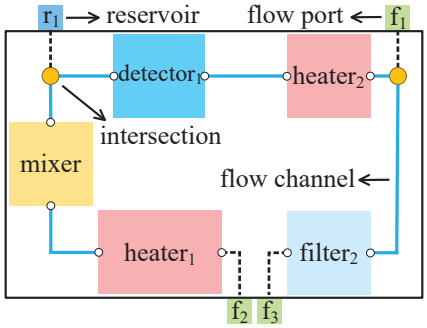

Figure 4: Chip layout constructed for the scheduling described in Fig. 3 using the existing design framework.

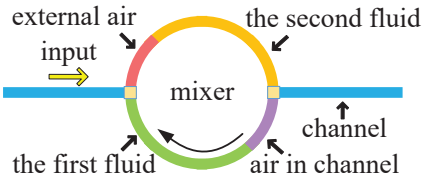

Figure 5: Failure of fluid volume control when transporting fluids to a mixer.

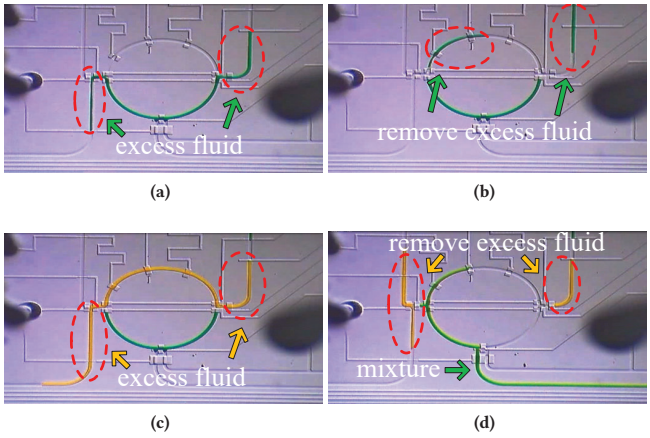

Figure 6: Snapshots of fluid manipulation when performing the mixing operation on a biochip [17]. (a) Step 1: loading the first fluid flow into the lower half of the mixer, (b) step2: removing the excess fluids left behind by the first fluid, (c) step 3: loading the second fluid flow into the upper half of the mixer and starting the mixing operation, (d) step 4: recovering the resulting mixture and removing the corresponding excess fluids.

bioassay. Fig. 3 shows a high-level synthesis result of the bioassay described in Fig. 2, where 5 devices are allocated to execute operations o_1–o_8 and the assay is completed in 19 seconds with 9 transportation tasks (#$_1$–#$_9$).

Afterwards, physical design is performed to assign the allocated devices to exact locations on the chip while establishing efficient connections among them. Correspondingly, Fig. 4 shows a chip layout generated by the existing design framework [12–14], where samples s_1–s_3 are injected into the chip through three flow ports f_1–f_3 and the outcome of the assay is collected by a reservoir r_1.

2.2 Design Challenges

During the architectural synthesis above, several key parameters such as completion time of the assay, total length of channels, and the number of intersections, etc., have been optimized systematically. Moreover, flow channels for all the transportation tasks (i.e., #$_1$–#$_9$) specified in Fig. 3 have been constructed during physical design. The chip layout shown in Fig. 4, however, is still not practical since the following key problems are ignored in the existing design framework: 1) fluid volume management between devices, 2) removal of excess/waste fluids, and 3) flow-path planning that enables the actual manipulation of fluid transportation/removal.

2.2.1 Volume Management Between Devices. In microfluidic biochips, as mentioned before, the initial status of a flow channel/device is not a vacuum but filled with air [10]. When executing biochemical operations, the air already present should be completely pushed out of devices, so that the operations can be executed correctly while ensuring the purity of samples/reagents. On the other hand, since the movement of fluids in flow channels is driven by the air pressure injected from external pressure sources [17], the air should avoid entering into the target device during fluid transportation.

Consider the mixing operation shown in Fig. 5 as an example, where two fluid flows have been loaded into the mixer, respectively. However, due to insufficient fluid volumes, external air enters into the upper half of the mixer and a small amount of air still remains in the lower half of the mixer. As a consequence, the two fluids that need to be mixed are separated during the mixing process, leading to execution failure of the operation. In contrast, Fig. 6 shows the snapshots of fluid manipulation when the mixing operation is performed in a wet laboratory. In Fig. 6(a), the lower half of the mixer is filled with fluid flow when loading the first sample. After removing the excess fluids at the two ends of the mixer in Fig. 6(b), the second fluid is loaded into the mixer in a similar manner, as shown in Fig. 6(c). Because no extra air is kept inside the mixer, the mixing of two fluids is completed successfully, as shown in Fig. 6(d). Note that since an extra output port is added to the bottom of the mixer, excess fluids of the second fluid flow and the resulting mixture can removed simultaneously in Fig. 6(d).

To avoid execution failure of operations, volumes of input fluids should be controlled strictly according to the capacities of devices. Specifically, each input of a device should be bound by a minimum volume, so that air can be pushed out of the device while avoiding the entering of external air. For example, the minimum input volume of a detector corresponds directly to its capacity since a detecting operation has only one input. Moreover, for a rotary mixer with two inputs, the volume of each input should be at least half of its capacity. Accordingly, when binding operations of a bioassay to specific devices in the high-level synthesis stage, volume constraints between devices should be considered carefully based on the dependencies specified in the sequencing graph.

Based on the analysis above, let us revisit the scheduling described in Fig. 3, where o_4 and o_7 are bound to detector$_1$ and the resulting fluids of both operations need to be transported to mixer. Since the maximum output volume of detector$_1$ is only 20 ml, which is smaller than the minimum input volume of mixer (i.e., 22.5 ml), this results in an invalid binding scheme. In contrast, if o_4 and o_7 are bound to detector$_2$, the volume constraint between the two devices can be satisfied.

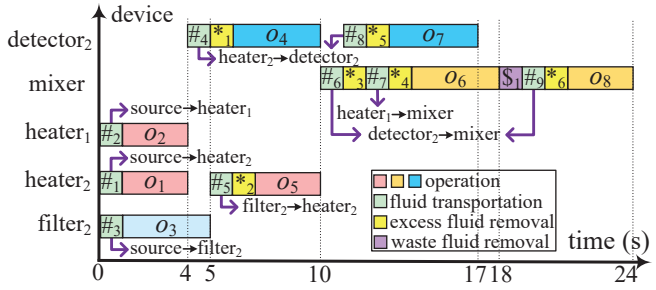

Figure 7: The updated scheduling corresponding to Fig. 3 after considering fluid volume management.

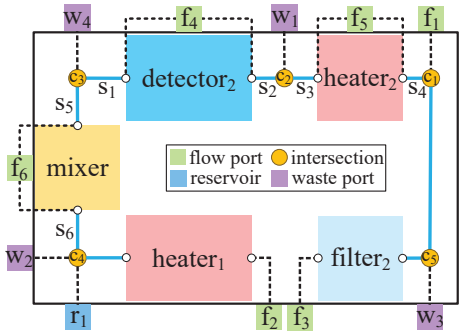

Figure 8: The updated chip layout corresponding to Fig. 4 based on the scheduling described in Fig. 7.

2.2.2 Removal of Excess/Waste Fluids. Besides fluid volume management between devices, another key issue that has been ignored in previous work is the removal of excess/waste fluids. After completing a fluid transportation task during the execution of a bioassay, as illustrated in Fig. 6, excess fluids cached at the two ends of a device should be removed from the chip. Additionally, fluids that are not needed anymore should also be discarded. For example, since operations o_6 and o_8 are bound to the same mixer in Fig. 3, before starting the execution of o_8, one half of the resulting fluid of o_6 should be removed so that the output fluid of o_7 can be loaded into the mixer.

Correspondingly, Fig. 7 shows a modified scheduling corresponding to Fig. 2, where both volume management and fluid removal discussed above have been considered. Note that since the volumes of source samples (i.e., s_1–s_3 in Fig. 2) can be controlled precisely to be equal to the minimum input volumes of the corresponding devices, there is no excess fluid after completing transportation tasks $\#_1$–$\#_3$.

2.2.3 Flow-Path Planning. With a practical scheduling scheme, flow-path planning should be integrated into the physical design of biochips, so that the actual manipulation of fluid transportation/removal can be realized. Note that a valid flow path is not a simple connection of several channel segments. As discussed previously, flow ports that connect with external pressure sources should be deployed to provide impetus for fluid movement. Moreover, waste ports should be connected to the end of flow paths to release the air in flow channels while recovering the excess/waste fluids. As a result, a complete flow path should be composed of fluidic ports, device(s) as well as flow channels. Accordingly, the construction of flow paths during architectural synthesis can be divided into three types:

Table 1: Flow paths of the fluid manipulations in the chip layout shown in Fig. 8

flow paths of fluid transportation
$\#_1$: $f_1 \rightarrow c_1 \rightarrow \text{heater}_2 \rightarrow c_2 \rightarrow w_1$
$\#_2$: $f_2 \rightarrow \text{heater}_1 \rightarrow c_4 \rightarrow w_2$
$\#_3$: $f_3 \rightarrow \text{filter}_2 \rightarrow c_5 \rightarrow w_3$
$\#_4, \#_8$: $f_1 \rightarrow c_1 \rightarrow \text{heater}_2 \rightarrow c_2 \rightarrow \text{detector}_2 \rightarrow c_3 \rightarrow w_4$
$\#_5$: $f_3 \rightarrow \text{filter}_2 \rightarrow c_5 \rightarrow c_1 \rightarrow \text{heater}_2 \rightarrow c_2 \rightarrow w_1$
$\#_6, \#_9$: $f_5 \rightarrow c_2 \rightarrow \text{detector}_2 \rightarrow c_3 \rightarrow \text{mixer} \rightarrow c_4 \rightarrow w_2$
$\#_7$: $f_2 \rightarrow \text{heater}_1 \rightarrow c_4 \rightarrow \text{mixer} \rightarrow c_3 \rightarrow w_4$
flow paths of excess fluid removal
$*_1, *_5$: $f_4 \rightarrow s_1 \rightarrow c_3 \rightarrow w_4$; $f_4 \rightarrow s_2 \rightarrow c_2 \rightarrow w_1$
$*_2$: $f_5 \rightarrow s_3 \rightarrow c_2 \rightarrow w_1$; $f_5 \rightarrow s_4 \rightarrow c_1 \rightarrow c_5 \rightarrow w_3$
$*_3, *_4, *_6$: $f_6 \rightarrow s_5 \rightarrow c_3 \rightarrow w_4$; $f_6 \rightarrow s_6 \rightarrow c_4 \rightarrow w_2$
flow path of waste fluid removal
$\$_1$: $f_4 \rightarrow c_3 \rightarrow \text{mixer} \rightarrow c_4 \rightarrow w_2$

- *Type 1* (fluid transportation): When performing a fluid transportation task between devices, the corresponding flow path should be ⌈flow port → source device → target device → waste port⌋.
- *Type 2* (excess fluid removal): When removing excess fluids cached at the two ends of a device, the corresponding flow paths should be ⌈flow port → caching position → waste port⌋.
- *Type 3* (waste fluid removal/external inputting): When removing waste fluids from a device or inputting a sample/reagent from an external source, the corresponding flow path should be ⌈flow port → target device → waste port⌋.

The flow paths above should be planned and constructed systematically during the architectural synthesis of biochips, so that all the specified fluid transportation/removal tasks can be completed efficiently. Furthermore, since flow-path planning directly determines the final chip architecture, chip costs such as channel length, should also be minimized in this stage. Accordingly, Fig. 8 shows an updated chip layout based on the scheduling described in Fig. 7, where flow paths of all the specified fluid manipulations have been constructed as shown in Table 1. Note that since extra space has already been reserved inside the bounding boxes of devices [14], flow channels can be line touched with their boundaries.

3 DETAILS OF THE PROPOSED SYNTHESIS FLOW

In this section, we discuss the proposed PathDriver in detail, which integrates the challenges discussed in Section 2.2 into both high-level synthesis and physical design, thus generating an efficient chip architecture supporting the actual manipulation of fluid transportation and removal. The proposed synthesis flow is formulated into integer linear programming problems in the following and solved by a solver accordingly.

3.1 High-Level Synthesis with Volume Management

Given the sequencing graph $G(O, E)$ of a bioassay and a device library D, the target of high-level synthesis in PathDriver is to generate a practical and optimized binding and scheduling scheme such that: 1) the completion time of the assay is minimized, 2) volume constraints

between devices are satisfied, 3) excess/waste fluids are removed efficiently, and 4) the number of required flow paths is minimized.

To complete the execution of the assay, each operation in O should be bound to a specific device in D, which can be formulated as

$$\sum_{d_k \in D} b_{i,k} = 1, \forall o_i \in O \tag{1}$$

where $b_{i,k}$ is a 0-1 variable representing whether operation o_i is bound to device d_k. For the sake of simplicity, hereinafter we denote the output fluid and the allocated device of an operation o_i by $out(o_i)$ and $d(o_i)$, respectively.

Before starting the execution of o_i, for each dependency $e_{j,i} \in E$, $out(o_j)$ should be transported from device $d(o_j)$ to $d(o_i)$. We use a 0-1 variable $c_{i,j}$ representing whether o_j is a child operation of o_i in the sequencing graph and assume that the resulting fluid of an operation is distributed equally to its child operations, the volume constraint between $d(o_j)$ and $d(o_i)$ can be formulated as

$$\frac{\sum_{d_k \in D} b_{j,k} \times v_k}{\sum_{o_k \in O} c_{j,k}} \geq \frac{\sum_{d_k \in D} b_{i,k} \times v_k}{\sum_{o_k \in O} c_{k,i}}, \ \forall e_{j,i} \in E \tag{2}$$

where v_k is the capacity of device d_k.

When transporting $out(o_j)$ to $d(o_i)$, if $d(o_i)$ is still occupied by another operation, $out(o_j)$ should be cached temporarily in a storage, leading to two sub-transportation tasks: 1) $p_{j,i,1}$: from $d(o_j)$ to the storage and 2) $p_{j,i,2}$: from the storage to $d(o_i)$. Note that o_i and o_j may be bound to the same device. In this case, after completing o_j, the device is allocated to execute another operation before o_i. We use variables t_i^s and t_i^e to represent the start time and end time of o_i, the durations of the two transportation tasks above can be constrained as

$$t^e_{p_{j,i,l}} - t^s_{p_{j,i,l}} = \begin{cases} T_p, & \exists o_h \in O, \sum_{d_k \in D} z_{i,h,k} > 0 \\ & \quad \forall e_{j,i} \in E, \\ & \wedge\, t_h^e < t_i^s \wedge t_h^e > t_j^e \quad , \ l = 1, 2. \\ 0, & otherwise \end{cases} \tag{3}$$

where $t^s_{p_{j,i,l}}$ and $t^e_{p_{j,i,l}}$ represent the start time and end time of the tasks $p_{j,i,l}$ ($l = 1, 2$), respectively. $z_{i,h,k}$ is a 0-1 variable representing whether o_i and o_h are bound to device d_k. Since the layout of the chip is remain undermined during high-level synthesis, we use a constant T_p to represent the duration of a transportation task. Note that $p_{j,i,2}$ can only be started after the completion of $p_{j,i,1}$ and thus can be constrained as

$$t^e_{p_{j,i,1}} \leq t^s_{p_{j,i,2}}, \ \forall e_{j,i} \in E \tag{4}$$

In contrary, if $d(o_i)$ is not occupied when o_j is completed, there is only one transportation task, denoted by $p_{j,i,3}$, directly from $d(o_j)$ to $d(o_i)$. Note that if the two operations are bound to the same device, the transportation above can be avoided accordingly. Thus, the duration of $p_{j,i,3}$ can be constrained as

$$t^e_{p_{j,i,3}} - t^s_{p_{j,i,3}} = \begin{cases} T_p, & \sum_{d_k \in D} z_{i,j,k} \leq 0 \\ 0, & \sum_{d_k \in D} z_{i,j,k} > 0 \end{cases}, \ \forall e_{j,i} \in E \tag{5}$$

Moreover, as discussed in Section 2.2.2 and Fig. 7, when transporting $out(o_j)$ to $d(o_i)$, if o_i is a multi-input operation, e.g., a mixing operation, and $d(o_i)$ is filled with the resulting fluid of another father operation of o_i, a part of the fluid should first be removed from $d(o_i)$, leading to a waste-fluid removal task, denoted by $w_{j,i}$, whose duration can be constrained as

$$t^e_{w_{j,i}} - t^s_{w_{j,i}} = \begin{cases} T_p, & \exists e_{h,i} \in E, \sum_{d_k \in D} z_{i,h,k} > 0 \\ & \quad \wedge\, t^e_{p_{h,i,1}} - t^s_{p_{h,i,1}} = 0, \ \forall e_{j,i} \in E \\ 0, & otherwise \end{cases} \tag{6}$$

$$t^e_{w_{j,i}} \leq t^s_{p_{j,i,k}}, \ \forall e_{j,i} \in E, k = 2, 3. \tag{7}$$

where $t^s_{w_{j,i}}$ and $t^e_{w_{j,i}}$ represent the start time and end time of the task $w_{j,i}$.

After completing the transportation of $out(o_j)$, if its volume is larger than the minimum input volume of $d(o_i)$, excess fluids cached at the two ends of $d(o_i)$ should be removed, leading to an excess fluid removal task, denoted by $r_{j,i}$, whose duration can be constrained as

$$t^e_{r_{j,i}} - t^s_{r_{j,i}} = \begin{cases} T_p, & \frac{\sum_{d_k \in D} b_{j,k} \times v_k}{\sum_{o_k \in O} c_{j,k}} > \frac{\sum_{d_k \in D} b_{i,k} \times v_k}{\sum_{o_k \in O} c_{k,i}} \\ & \quad \wedge\, 1 - \sum_{d_k \in D} z_{i,j,k} + \rho_{j,i} > 0, \ \ \forall e_{j,i} \in E \\ 0, & otherwise \end{cases} \tag{8}$$

$$t^s_{r_{j,i}} \geq t^e_{p_{j,i,k}}, \ \forall e_{j,i} \in E, k = 2, 3. \tag{9}$$

where $t^s_{r_{j,i}}$ and $t^e_{r_{j,i}}$ represent the start time and end time of the task $r_{j,i}$ and $\rho_{j,i}$ is a 0-1 variable representing whether a storage is introduced in the transportation from $d(o_j)$ to $d(o_i)$. Moreover, if o_i is a multi-input operation, the next input should be started after $r_{j,i}$ is completed. Note that the input fluids of o_i should be loaded separately. Thus, we have the following constraints:

$$\begin{aligned} (1 - \lambda_{j,h,i})M + t^s_{p_{h,i,k}} &\geq t^e_{p_{j,i,k}} \\ \lambda_{j,h,i} M + t^s_{p_{j,i,k}} &\geq t^e_{p_{h,i,k}} \end{aligned}, \ \forall e_{j,i}, e_{h,i} \in E, k = 2, 3. \tag{10}$$

$$t^s_{p_{h,i,k}} \geq t^e_{r_{j,i}}, \ \exists e_{h,i} \in E \wedge t^s_{p_{h,i,k}} \geq t^e_{p_{j,i,k}}, k = 2, 3. \tag{11}$$

where $\lambda_{j,h,i}$ is a 0-1 variable representing the input order of $out(o_j)$ and $out(o_h)$ with respect to $d(o_i)$ and M is a very large constant to transform the two situations indicated by $\lambda_{j,h,i}$ into linear constraints.

After completing the transportation of all the input fluids of o_i, the execution of o_i can be started and should last long enough, constrained as

$$t_i^s \geq t^e_{p_{j,i,k}}, \ \forall e_{j,i} \in E, k = 2, 3. \tag{12}$$

$$t_i^s + t(o_i) \leq t_i^e \tag{13}$$

where $t(o_i)$ is the execution time of o_i specified in the sequencing graph.

Only when both the execution of o_i and the removal of corresponding excess fluids are completed, the resulting fluid can be transported to the next operation and thus we have

$$t^s_{p_{i,h,k}} \geq t_i^e, \ t^s_{p_{i,h,k}} \geq t^e_{r_{j,i}}, \ \forall e_{j,i}, e_{i,h} \in E, k = 1, 3. \tag{14}$$

Similar to the situation when loading fluids into a device, if o_i is a multi-output operation, e.g., a splitting operation, the corresponding output fluids should be removed separately and thus can be constrained as

$$\begin{aligned} (1 - \mu_{i,h,q})M + t^s_{p_{i,h,k}} &\geq t^e_{p_{i,q,k}} \\ \mu_{i,h,q} M + t^s_{p_{i,q,k}} &\geq t^e_{p_{i,h,k}} \end{aligned}, \ \forall e_{i,h}, e_{i,q} \in E, k = 1, 3. \tag{15}$$

where $\mu_{i,h,q}$ is a 0-1 variable representing the output order of the resulting fluids of o_i.

Additionally, after completing the execution of o_i, if there exists another operation $o_q \in O$ that needs to be executed using the same

device, it can only be started after removing the resulting fluid of o_i. Thus, we have the following cosntraint

$$t^s_{p_{u,q,k_1}} \geq t^e_{p_{i,h,k_2}}, \quad \begin{array}{c} \exists o_q \in O \wedge t^s_q > t^e_i \wedge \sum_{d_k \in D} z_{i,q,k} > 0, \\ \forall e_{u,q}, e_{i,h} \in E, k_1 = 2,3, k_2 = 1,3. \end{array} \tag{16}$$

After completing the execution of all the operations, the complete assay is finished. We use a variable T_{assay} to represent the completion time of the assay, which can be constrained as

$$t^e_i \leq T_{assay}, \ \forall o_i \in O \tag{17}$$

Finally, an optimized binding and scheduling scheme minimizing both completion time of the assay and the number of flow paths can be determined by solving the following optimization problem

$$\begin{aligned} \text{minimize } \alpha \times T_{assay} + \beta \times \sum_{e_{j,i} \in E} \frac{1}{T_p} (\sum_{k=1}^{3} (t^e_{p_{j,i,k}} - t^s_{p_{j,i,k}}) \\ + 2 \times (t^e_{r_{j,i}} - t^s_{r_{j,i}}) + t^e_{w_{j,i}} - t^s_{w_{j,i}}) \qquad (18) \\ \text{subject to } (1) - (17) \qquad (19) \end{aligned}$$

where α and β are two weighting factors.

3.2 Physical Design with Flow-Path Planning

After completing the high-level synthesis, a set $D_a \subset D$ of allocated devices and a set L of flow paths that need to be constructed are obtained accordingly. In this subsection, physical design is performed to geneate a chip architecture such that the previously generated scheduling can be realized while minimizing chip costs such as the total length of flow channels, the number of channel intersections, etc.

To generate an optimized chip architecture, PathDriver adopts a virtual grid R of size $W \times H$ to realize the placement of devices and the construction of flow paths as illustrated in Fig. 9. Each cell on the grid can be represented by the coordinates of its lower-left corner. Moreover, the complete grid is divided into the following two regions:

- *Port layer*: The cells on the boundary of the grid are used to deploy fluidic ports and can be mathematically represented as $R_p = \{(x,y)|0 \leq x < W, y = 0, H-1\} \cup \{(x,y)|x = 0, W-1, 0 \leq y < H\}$.
- *Device layer*: The cells surrounded by port layer are used to deploy devices and can be mathematically represented as $R_d = \{(x,y)|0 < x < W-1, 0 < y < H-1\}$.

For example, in Fig. 9, a flow port and a waste port are placed at the lower-right and upper-left corners of the grid, respectively. The two ports are connected with a mixer placed at the center of the device layer, forming a complete flow path.

To ensure that a device is placed inside the device layer, we have the following constraints

$$x_i > 0, y_i > 0, x_i + w_i < W, y_i + h_i < H, \ \forall d_i \in D_a \tag{20}$$

where (x_i, y_i) are the coordinates of the lower-left corner of d_i. w_i and h_i are the width and height of d_i on the grid, respectively.

Moreover, we use a 0-1 variable $u^{d_i}_{x,y}$ to represent whether a cell (x,y) in the device layer is occupied by device d_i. Since any two devices cannot overlap with each other, we have

$$\sum_{d_i \in D_a} u^{d_i}_{x,y} \leq 1, \ \forall (x,y) \in R_d \tag{21}$$

When performing the flow-path planning of each $l_i \in L$, a flow port and a waste port should be placed at the two ends of l_i and thus can be

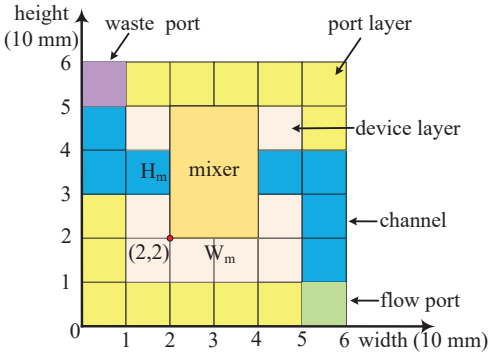

Figure 9: Illustration of the virtual grid used in PathDriver.

constrained as

$$\sum_{(x,y) \in R_p} u^{f_i}_{x,y} = 1, \ \sum_{(x,y) \in R_p} u^{w_i}_{x,y} = 1, \ \forall l_i \in L \tag{22}$$

where $u^{f_i}_{x,y}$ and $u^{w_i}_{x,y}$ are 0-1 variables indicating whether a cell (x,y) is occupied by the flow port and the waste port of flow path l_i, respectively.

Similarly, a flow port and a waste port cannot be placed at the same cell on the grid and thus can be constrained as

$$u^{f_i}_{x,y} + u^{w_i}_{x,y} \leq 1, u^{f_i}_{x,y} + u^{w_j}_{x,y} \leq 1, \forall l_i, l_j \in L, \forall (x,y) \in R_p \tag{23}$$

Flow paths can be divided into three types as mentioned in Section 2.2.3. To construct the corresponding flow channels of l_i, it can be further split into a set of end-to-end routing nets, denoted by N_{l_i}. For example, a *Type 1* path consists of three nets, i.e., (flow port, source device), (source device, target device), and (target device, waste port).

Afterwards, for each net $n_j(v^1_j, v^2_j) \in N_{l_i}$, where v^1_j and v^2_j are the two ends of n_j, which can be a fluidic port, a device, or a grid cell occupied by excess fluid, our task is to construct a routing path of n_j such that v^1_j and v^2_j are connected by channels. Correspondingly, one of the adjacent cells of v^1_j and v^2_j on the grid should be the start point and end point of the routing path, respectively. Thus, we have the following constraints

$$\sum_{(x,y) \in AC_{v^1_j}} u^{n_j}_{x,y} = 1, \ \sum_{(x,y) \in AC_{v^2_j}} u^{n_j}_{x,y} = 1, \ \forall n_j \in N_{l_i} \tag{24}$$

where $u^{n_j}_{x,y}$ is a 0-1 variable representing whether a cell (x,y) is on the routing path of n_j. $AC_{v^1_j}$ and $AC_{v^2_j}$ are adjacent cells of v^1_j and v^2_j on the grid, respectively.

Meanwhile, for any cell $(x,y) \in R$ on the routing path of n_j except for the start and end points, to ensure the connectivity of the path, exactly two adjacent cells of (x,y) should also be on the path, constained as

$$\begin{aligned} \sum_{(x',y') \in AC_{x,y}} u^{n_j}_{x',y'} = 2, \\ \forall n_j \in N_{l_i}, \forall (x,y) \in R \wedge u^{n_j}_{x,y} = 1 \wedge (x,y) \notin AC_{v^1_j} \cup AC_{v^2_j} \end{aligned} \tag{25}$$

where $AC_{x,y}$ is a set of adjancent cells of (x,y) on the grid.

Since flow path l_i cannot pass through any cell occupied by a fluidic port, we have

$$u^{n_j}_{x,y} + u^{f_k}_{x,y} + u^{w_k}_{x,y} \leq 1, \forall n_j \in N_{l_i}, \forall l_k \in L, \forall (x,y) \in R, \tag{26}$$

Furthermore, if there exists another flow path l_j, whose correponding fluid transportation/removal task needs to be performed concurrently

with that of l_i in the scheduling, denoted by $l_i \parallel l_j$, the two flow paths cannot share a waste port and should avoid forming any channel intersection. Thus, we have following constaints

$$u_{x,y}^{w_i} + u_{x,y}^{w_j} \leq 1, \forall (x,y) \in R_p, \forall l_i, l_j \in L \wedge l_i \parallel l_j \tag{27}$$

$$\sum_{n_k \in N_{l_i} \cup N_{l_j}} u_{x,y}^{n_k} \leq 1, \forall (x,y) \in R, \forall l_i, l_j \in L \wedge l_i \parallel l_j \tag{28}$$

Note that if either l_i or l_j above is an inputting task from an external source (e.g., s_1–s_3 in Fig. 2), denoted by $l_i \overset{s}{\parallel} l_j$, the two flow paths cannot share a flow port, constrained as

$$u_{x,y}^{f_i} + u_{x,y}^{f_j} \leq 1, \forall (x,y) \in R_p, \forall l_i, l_j \in L \wedge l_i \overset{s}{\parallel} l_j \tag{29}$$

Addionally, we refer to a device that is performing either an inputting/outputting task or a biochemical operation as an active device. Since flow path l_i cannot pass through an active device during its execution period, we have

$$u_{x,y}^{n_k} + u_{x,y}^{d_j} \leq 1, \forall n_k \in N_{l_i}, \forall (x,y) \in R_d, \forall d_j \in D_i \tag{30}$$

where D_i is a set of active devices during the exection of l_i's fluid transportation/removal task.

We use 0-1 varibales $S_{x,y}^c$ and $S_{x,y}^g$ to represent whether a cell (x,y) is occupied by a flow channel and a channel intersetion on the chip, respectively, which can be constained as

$$u_{x,y}^{n_i} \leq S_{x,y}^c, S_{x,y}^c \leq \sum_{n_j \in N} u_{x,y}^{n_j}, \forall n_i \in N, \forall (x,y) \in R \tag{31}$$

$$S_{x,y}^g = \begin{cases} 1, & u_{x,y}^{n_i} + u_{x,y}^{n_j} > 1 \wedge u_{x',y'}^{n_i} + u_{x',y'}^{n_j} = 1 \\ 0, & otherwise \end{cases} \tag{32}$$
$$\forall (x,y) \in R, \forall l_i, l_j \in L, \forall (x',y') \in AC_{x,y}$$

where N is the set of the end-to-end nets after spliting all the flow paths in L.

Finally, a biochip architecture with minimized cost can be generated by solving the following optimization problem

$$\text{minimize } \gamma \sum_{(x,y) \in R} S_{x,y}^c + \xi \sum_{(x,y) \in R} S_{x,y}^g \tag{33}$$

$$\text{subject to (20)–(32)} \tag{34}$$

where γ and ξ are two weighting factors.

4 EXPERIMENTAL RESULTS

The proposed synthesis flow PathDriver was implemented in C++ language and tested on a PC with 2.50-GHz CPU and 8-GB memory. Six benchmarks were used to verify the proposed method as shown in Table 2, in which PCR (Polymerase Chain Reaction), IVD (In-Vitro Diagnostics) and ProteinSplit are real-world biochemical applications [16] and the other three assays are synthetic benchmarks. Parameters $|O|$, $|E|$, and $|device|$ in Table 2 are the numbers of biochemical operations and edges in the sequencing graph and the number of corresponding devices in the device library, respectively. The parameters used in this paper are set as follows: α=0.3, β=0.7, γ=0.55, ξ=0.45, and T_p=1 s. Moreover, the runtime on each benchmark was limited to 30 minutes for the solver to return the best-effort results.

Since there is no existing work targeting the architectural synthesis of microfluidic biochips considering volume management, excess/waste fluid removal, and flow-path planning simultaneously, we implemented another synthesis method, called LSA, to verify the effectiveness of the proposed PathDriver. LSA integrates volume management and fluid removal into the well-know list scheduling algorithm to generate an efficient scheduling. The latter has been widely adopted in prior high-level synthesis work such as [7], [11], [14], etc. Afterwards, simulated annealing algorithm [18] is employed to generate placement results of devices and A*-search method is applied to complete the corresponding flow-path planning. We ran both algorithms on the aforementioned benchmarks. Table 3 shows the corresponding comparison results, where columns T_{assay}, N_{path}, N_{port}, $N_{intersection}$, $L_{channel}$ are the completion time of bioassays, the total number of flow paths, the number of fluidic ports used in the chip, the number of channel intersections, and the total length of flow channels, respectively. I_m (%) in Table 3 provides the relative improvement of PathDriver over LSA.

Table 2: Benchmarks used in the experiments

Benchmarks $\|O\|$/(|mixer|, |heater|, |filter|, |separator|, |detector|, |storage|)/$\|E\|$		
PCR 7/(4,0,0,0,0,0,1)/15	IVD 12/(4,0,0,0,0,4,1)/24	ProteinSplit 14/(4,0,0,3,3,3,1)/27
Synthetic1 10/(4,2,3,0,2,2,1)/15	Synthetic2 15/(4,3,2,0,3,3,1)/21	Synthetic3 20/(4,4,3,4,4,2,1)/28

In high-level synthesis, it can seen from Table 3 that PathDriver achieves a 7.55%–13.64% reduction in terms of the completion time of bioassays, with an average reduction of 9.59%. Moreover, the number of flow paths that need to be constructed is reduced by 40.75% on average, thereby reducing the complexity of chip architecture significantly. This result is due to two reasons: 1) although list scheduling has the advantage of being easy to be implemented, the corresponding solution quality is limited due to the adoption of a greedy strategy. In contrast, PathDriver considers the scheduling of all the operations specified in the sequencing graph in a unified manner, thus reducing the completion time of bioassays and 2) in PathDriver, by binding operations to appropriate devices and executing them in optimized time windows, unnecessary fluid caching in a storage and waste fluid removal can be avoided, thereby reducing the number of flow paths.

Similarly, in physical design, PathDriver achieves 40.99%, 57.98%, and 45.82% reduction on average with respect to the number of fluidic ports, the number of channel intersections, and the total channel length, respectively. This is due to the following reasons: 1) since the number of flow paths has been reduced significantly during high-level synthesis, the complexity of physical design is reduced accordingly, thereby avoiding introducing unnecessary fluidic ports and channel intersections and 2) flow channels are shared among flow paths as long as no conflict is introduced. As shown in Fig. 10, PathDriver improves the channel sharing rate by 61% on average compared with LSA.

Finally, Fig. 11 shows the comparison results on the number of extra valves introduced into the chip. Since 2–4 valves need to be placed at a channel intersection to direct fluid flow towards the right direction [16], these valves need to be controlled independently during the execution of bioassays. It can be seen that PathDriver achieves 44% reduction on average, thus reducing the complexity of chip control significantly.

5 CONCLUSION

We have investigated the architectural synthesis of continuous-flow microfluidic biochips considering fluid volume management, excess/waste fluid removal, and flow-path planning simultaneously, and proposed a path-driven synthesis flow called PathDriver to generate a practical biochip architecture systematically. With the proposed design automation method, actual fluid transportation and removal can be realized

Table 3: Comparison results between PathDriver (PD) and LSA with respect to high-level synthesis and physical design

Benchmark	High-Level Synthesis						Physical Design								
	T_{assay} (s)			N_{path}			N_{port}			$N_{intersection}$			$L_{channel}$ (mm)		
	LSA	PD	I_m (%)	LSA	PD	I_m (%)	LSA	PD	I_m (%)	LSA	PD	I_m (%)	LSA	PD	I_m (%)
PCR	25	23	8.00	32	17	46.88	11	8	27.27	11	5	54.55	130	80	38.46
IVD	33	30	9.09	30	18	40.00	11	4	63.64	5	2	60.00	200	60	70.00
ProteinSplit	101	93	7.92	54	25	53.70	11	6	45.45	9	3	66.67	100	60	40.00
Synthetic1	44	39	11.36	31	22	29.03	13	9	30.77	16	6	62.50	210	110	47.62
Synthetic2	53	49	7.55	34	24	29.41	11	6	45.45	8	5	37.50	170	70	58.82
Synthetic3	66	57	13.64	44	24	45.45	9	6	33.33	6	2	66.67	100	80	20.00
Average	–		9.59	–		40.75	–		40.99	–		57.98	–		45.82

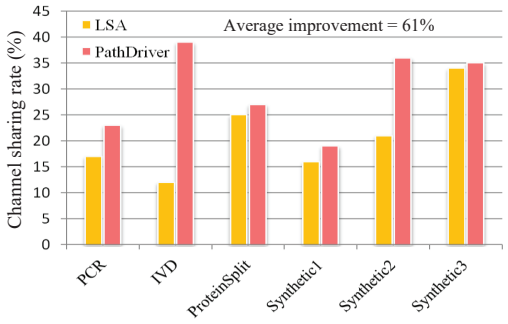

Figure 10: Comparison on the channel sharing rate.

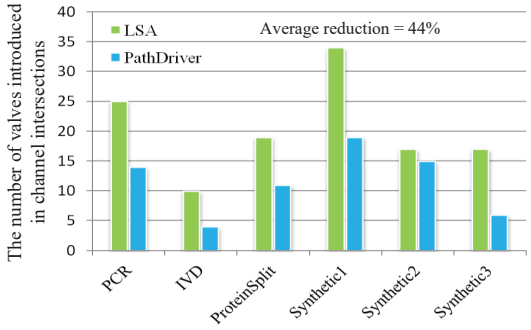

Figure 11: Comparison on the number of valves introduced in channel intersections.

through an efficient flow-path network with minimized cost. Meanwhile, fluid volumes are constrained strictly to ensure the correctness of assay outcomes. Experimental results have confirmed that PathDriver leads to chip architectures with both high efficiency and low cost. Future work will focus on the establishment of an open-source ecosystem for continuous-flow microfluidic biochips [19] as well as the security-related issues [20].

ACKNOWLEDGMENT

This work is supported in part by Deutsche Forschungsgemeinschaft (DFG) through TUM International Graduate School of Science and Engineering (IGSSE), the Technical University of Munich–Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763, and the National Natural Science Foundation of China under Grants No. 61877010 and No. 11501114.

REFERENCES

[1] B. Zheng, L. S. Roach, and R. F. Ismagilov, "Screening of protein crystallization conditions on a microfluidic chip using nanoliter-size droplets," *Journal of the American chemical society*, vol. 125, no. 37, pp. 11 170–11 171, 2003.

[2] L.-H. Hung, K. M. Choi, W.-Y. Tseng, Y.-C. Tan, K. J. Shea, and A. P. Lee, "Alternating droplet generation and controlled dynamic droplet fusion in microfluidic device for cds nanoparticle synthesis," *Lab on a Chip*, vol. 6, no. 2, pp. 174–178, 2006.

[3] S. Bhattacharjee, R. Wille, J.-D. Huang, and B. B. Bhattacharya, "Storage-aware algorithms for dilution and mixture preparation with flow-based lab-on-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 4, pp. 816–829, 2020.

[4] Global microfluidics market will surpass usd 12,380 million by 2025. [Online]. Available: https://www.zionmarketresearch.com/.

[5] Y. Zhu, X. Huang, B. Li, T.-Y. Ho, Q. Wang, H. Yao, R. Wille, and U. Schlichtmann, "Multicontrol: Advanced control logic synthesis for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2019, doi: 10.1109/TCAD.2019.2940688.

[6] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic large-scale integration," *Science*, vol. 298, no. 5593, pp. 580–584, 2002.

[7] Z. Chen, X. Huang, W. Guo, B. Li, T.-Y. Ho, and U. Schlichtmann, "Physical synthesis of flow-based microfluidic biochips considering distributed channel storage," in *Proc. Design, Autom., and Test Europe Conf.*, 2019, pp. 1525–1530.

[8] C. Liu, X. Huang, B. Li, H. Yao, P. Pop, T.-Y. Ho, and U. Schlichtmann, "DCSA: Distributed channel-storage architecture for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2020, doi: 10.1109/TCAD.2020.2994267.

[9] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 1, pp. 55–68, 2017.

[10] M. Li, T.-M. Tseng, Y. Ma, T.-Y. Ho, and U. Schlichtmann, "VOM: Flow-path validation and control-sequence optimization for multilayered continuous-flow microfluidic biochips," in *Proc. Int. Conf. Comput.-Aided Des.*, 2019, pp. 1–8.

[11] W. H. Minhass, J. McDaniel, M. Raagaard, P. Brisk, P. Pop, and J. Madsen, "Scheduling and fluid routing for flow-based microfluidic laboratories-on-a-chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 3, pp. 615–628, 2018.

[12] Q. Wang, H. Zou, H. Yao, T.-Y. Ho, R. Wille, and Y. Cai, "Physical co-design of flow and control layers for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 6, pp. 1157–1170, 2018.

[13] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," in *Proc. Asia and South Pacific Des. Autom. Conf.*, 2017, pp. 530–535.

[14] X. Huang, T.-Y. Ho, W. Guo, B. Li, and U. Schlichtmann, "Minicontrol: Synthesis of continuous-flow microfluidics with strictly constrained control ports," in *Proc. Design Autom. Conf.*, 2019, pp. 145:1–6.

[15] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1588–1601, 2017.

[16] X. Huang, T.-Y. Ho, K. Chakrabarty, and W. Guo, "Timing-driven flow-channel network construction for continuous-flow microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 6, pp. 1314–1327, 2019.

[17] Programmable microfluidics. [Online]. Available: http://groups.csail.mit.edu/cag/biostream/.

[18] K. A. Dowsland and J. Thompson, "Simulated annealing," *Handbook of natural computing*, pp. 1623–1655, 2012.

[19] X. Huang, C.-C. Liang, J. Li, T.-Y. Ho, and C.-J. Kim, "Open-source incubation ecosystem for digital microfluidics—status and roadmap," in *Proc. Int. Conf. Comput.-Aided Des.*, 2019, pp. 1–6.

[20] C. Dong, L. Liu, H. Liu, W. Guo, X. Huang, S. Lian, X. Liu, and T.-Y. Ho, "A survey of DMFBs security: State-of-the-art attack and defense," in *Proc. Int. Symp. Quality Electron. Des.*, 2020, pp. 14–20.