Integrated Functional and Washing Routing Optimization for Cross-Contamination Removal in Digital Microfluidic Biochips

Hailong Yao, *Senior Member, IEEE*, Qin Wang, Yiren Shen, Tsung-Yi Ho, *Senior Member, IEEE*, and Yici Cai

***Abstract*—Digital microfluidic biochips (DMFBs) are gaining increasing attention with promising applications for automating and miniaturizing laboratory procedures in biochemistry. In DMFBs, cross-contamination of droplets with different biomolecules is a major issue, which causes significant errors in bioassays. Washing operations are introduced to clean the cross-contamination spots. However, existing works have oversimplified assumptions on the washing behavior, which either assume infinite washing capacity, or ignore the routing conflicts between functional and washing droplets. This paper proposes the first integrated functional and washing droplet routing flow, which considers practical issues including the finite washing capacity constraint, and the routing conflicts between functional and washing droplets. Washing droplets of different sizes are also proposed to wash the congested cross-contamination spots. Effectiveness of the proposed method is validated by real-life biochemical applications.**

***Index Terms*—Cross-contamination, digital microfluidic biochips (DMFBs), droplet routing, washing capacity constraint.**

I. Introduction

DIGITAL microfluidic biochips (DMFBs), as a revolutionary technique in the realization of the lab-on-a-chip (LoC), are emerging for the automation and miniaturization of laboratory in biochemistry [2], [3]. In an LoC platform, nanoliter-sized droplets are manipulated on an open surface of a 2-D array of electrodes using the electrowetting technology [3]. Compared with the traditional laboratory procedures, DMFB greatly reduces the analysis time and the sample and reagent consumption, and thus has many promising applications in biochemical analysis including enzymatic assays, DNA sequencing, cell-based assays, and immunoassays [3]–[6]. Besides, DMFBs can both be dynamically reconfigured for different types of sequential experiments, and be used for multiplexed assays at the same time. These merits significantly enhance the flexibility and throughput.

DMFBs are based on the electrowetting technology [3], which controls the wetting behavior of a polarizable or conductive liquid droplet by an electric field, so as to control the movement of the droplet. Fig. 1 shows an exemplary schematic of a DMFB. In Fig. 1(a), the cross-sectional view is given.

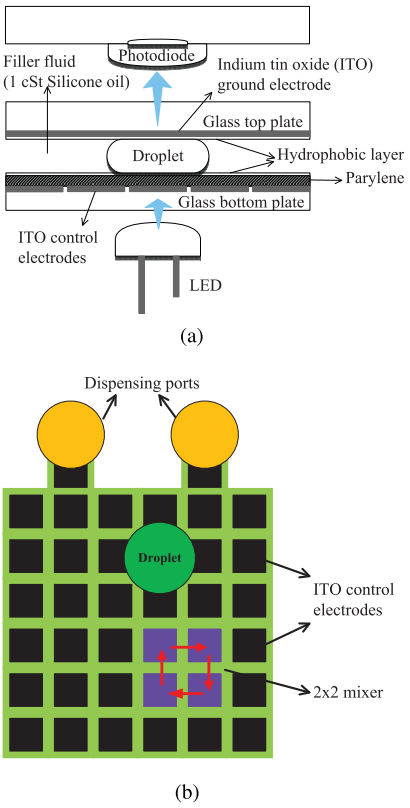


Fig. 1. Schematic of a DMFB [2], [3]. (a) Cross-sectional view. (b) Top view.

Manuscript received March 31, 2015; revised May 27, 2015, August 7, 2015, and October 24, 2015; accepted November 6, 2015. Date of publication November 29, 2015; date of current version July 15, 2016. The work of H. Yao was supported in part by the Tsinghua University Initiative Scientific Research Program under Grant 20141081203, in part by the Doctoral Fund of Ministry of Education of China under Grant 20111011328, and in part by the National Natural Science Foundation of China under Grant 61106104. The work of T.-Y. Ho was supported by the Taiwan Ministry of Science and Technology under Grant MOST 102-2221-E-007-149-MY3, Grant 103-2923-E-007-004-MY3, and Grant 104-2220-E-007-021. The work of Y. Cai was supported by the National Natural Science Foundation of China under Grant 61274031. A preliminary version of this paper appeared in [1]. Extensions beyond [1] include the path ordering method for addressing the deadlock issue, the adoption of washing droplets of different sizes with experimental validations, more detailed description of the proposed methods and algorithms, and updated computational simulation results. This paper was recommended by Associate Editor Y. Chen.

H. Yao, Q. Wang, and Y. Cai are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: hailongyao@tsinghua.edu.cn; woodythu@163.com; caiyc@mail.tsinghua.edu.cn). This work was done during Y. Shen's internship at Tsinghua University (e-mail: shenyiren36@gmail.com).

T.-Y. Ho is with the Department of Computer Science, National Tsinghua University, Hsinchu 30013, Taiwan (e-mail: tyho@cs.nthu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2504397

0278-0070 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

By applying a sequence of actuation voltages to control electrodes, the droplets between the top and bottom plates will move along adjacent cells as expected. Here, a cell refers to the square room of a control electrode. Utilizing the electrowetting technology, automatic biochemical experiments can be performed. Different sample and reagent droplets can be transported to the same cell for mixing and then transported to another cell for detection. A typical method for detection is to use LED and photodiode detector as shown in the figure.

Fig. 1(b) shows the top view of the DMFBs 2-D electrode array. The dispensing ports are used to input/output the droplets. As shown in the figure, a droplet could be so large that droplets sitting on adjacent electrodes will automatically mix together. Therefore, droplets are not allowed to be adjacent unless they are planned to mix together. Here, a mixer is needed for completely mixing two droplets. Fig. 1(b) shows a 2×2 mixer, which is a 2×2 electrode array circularly actuated by switching voltages. There are also other modules for the biochemical experiments, such as the mixers of other sizes, the storage cell, etc. For different types of sequential experiments, these modules can be dynamically reconfigured to different places by rescheduling the actuation voltages on the electrodes. As a result, the whole droplet routing problem can be decomposed into a series of subproblems, where in each subproblem the modules are located at prespecified positions for a specific experiment step. As the size of DMFB becomes larger and the bioassay becomes more complicated, computer-aided design methods are becoming necessary for the automated droplet path computation and scheduling.

In the past decade, noticeable advances have been made in computer-aided design methods for DMFBs, including resource binding, operation scheduling, module placement, and droplet routing [7]–[12]. Among the different stages in the automated design flow, droplet routing is a most important stage, which determines the final routing paths for droplets between reservoirs/dispensing ports, optical detectors, etc., and thus determines the correctness and performance (execution time) in implementing the assays. Previous droplet routing methods mostly focus on two basic routing constraints [7], [9]–[12]: 1) fluidic constraint to avoid unexpected mixing of two droplets during their transportation and 2) timing constraint to satisfy the maximum allowed transportation time of a droplet. A typical objective is to minimize the number of cells used for droplet routing, such that the number of driving electrodes can be minimized for power and interconnection savings.

The above-mentioned basic constraints do not consider the cross-contamination issue. Cross-contamination occurs between sequential droplet routes on their intersection spots. As functional (i.e., sample or reagent) droplets leave residues on cells (electrodes) along their paths, cross-contamination occurs when the routing paths have intersections. Although there is filler fluid (e.g., silicone oil) between the top and bottom plates, it is still unavoidable for functional droplets to leave residues along their paths, which causes significant contamination issue. This is especially true for many types of proteins and heterogeneous immunoassays, because proteins tend to adsorb the hydrophobic surface. As a result, the particles and liquid residues will probably lead to cross-contamination.

TABLE I
COMPARISON BETWEEN THE PROPOSED METHOD AND EXISTING WORKS

Methods	❶	❷	❸	❹	❺
[14]	Yes	No	No	No	No
[15]	Yes	No	No	No	No
[16]	Yes	No	Yes	No	No
[17]	Yes	No	N/A	No	No
[18]	Yes	No	No	No	No
[19]	Yes	Yes	No	Yes	No
Our Method	Yes	Yes	Yes	Yes	Yes

❶ Use Washing Droplets
❷ With Washing Capacity Limit
❸ Transport Functional and Washing Droplets Concurrently
❹ Consider Washing Capacity Consumption of Cross-Contamination Spot
❺ Consider Washing Capacity Consumption of All Residues
N/A: Not Clearly Stated in the Paper

Such cross-contamination will cause significant errors in assay outcome.

Therefore, routing paths of different nets[1] should ideally be disjoint from each other to avoid the number of cross-contamination spots. When disjoint routing paths are not available, which is very common due to the single routing layer, the so-called washing droplets are introduced for cleaning the prior droplets' residue before the latter droplet passes through the intersection spot [13]. Several droplet-routing methods have been proposed to consider the cross-contamination issue [14]–[18]. However, the above works have oversimplified assumptions that the washing droplets have unlimited washing capacity. In fact, the washing capacity of a washing droplet will decrease when residues are washed away from the electrodes. Thus, the capacity constraint for a washing droplet needs to be considered [19].

This paper proposes the first integrated functional and washing droplet routing flow considering the realistic washing capacity constraint. More importantly, functional routing and washing routing are simultaneously considered to resolve the routing conflicts. When the washing droplet is heading toward a specific cross-contamination spot, it should avoid the cells with residues of other functional droplets as much as possible. When the residues are unavoidable, the washing capacity will be consumed accordingly. In congested DMFB designs, certain cross-contamination spot may be surrounded by so many functional paths that the washing capacity of a small washing droplet may be exhausted before it reaches the spot for residue cleaning. Thus, this paper first proposes to use larger washing droplets with larger capacity to wash those congested spots. Table I shows the difference between the proposed method and existing works in cross-contamination avoidance for DMFBs. Major contributions of this paper are as follows.

1) The first integrated functional and washing droplet routing flow is proposed while considering the realistic washing capacity constraint.
2) Effective functional and washing routing as well as compaction algorithms are proposed to derive satisfactory

[1]In the droplet routing context, a net refers to a set of electrodes to be connected, among which there may be more than one source electrodes and a single target electrode.

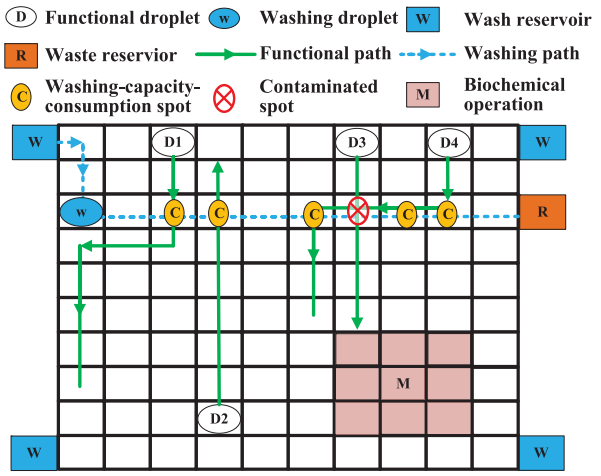


Fig. 2. Practical issues in washing operation with realistic capacity constraint.

paths with well-scheduled droplets for correct washing functionality.

3) A path ordering method is proposed to address the deadlock issue when scheduling both functional and washing paths.
4) Larger washing droplets with larger capacities are introduced to wash the congested cross-contamination spots.

The rest of this paper is organized as follows. Section II states the motivation and problem formulation. Section III gives the overview of the proposed flow. Section IV proposes the functional routing method. Section V proposes the washing routing method. Section VI gives the computational simulation results. Finally, the conclusion is drawn in Section VII.

II. Motivation and Problem Formulation

A. Motivation

Fig. 2 illustrates an example showing the practical issues in the washing operation with capacity constraint. In the figure, a washing droplet w is dispensed from the wash reservoir on the top left corner. In the experiments, we adopt the same configuration as [16] that there are four wash reservoirs at the four corners. In Fig. 2, the washing droplet w will clean the cross-contamination spot caused by functional droplets D_3 and D_4. However, the washing path intersects with the functional paths of droplets D_1 and D_2, respectively. Thus, it is possible for the residue of D_1 and D_2 to consume w's washing capacity, even if the routing paths are carefully synchronized. In this paper, we call the above issue as routing conflicts between functional and washing droplets. If we want to keep the washing droplet clean on its way, then we have to make D_1 and D_2 wait until w passes the washing-capacity-consumption spots. That may result in timing constraint violations on functional droplets D_1 or D_2. The same issue happens to D_3 and D_4. As the washing capacity of w is limited, we need to avoid the possible washing-capacity-consumption spots as many as possible, in order to wash more cross-contamination spots. Another important issue is that w needs to reach the cross-contamination spot after the first functional droplet passes the spot, as well as before the second functional droplet reaches the spot. Only in this way can the washing operation be meaningful. This paper first addresses both above important washing issues.

B. Problem Formulation

There are four constraints in contamination-aware functional and washing droplet routing: 1) the fluidic constraint; 2) the timing constraint; 3) the contamination constraint; and 4) the washing capacity constraint. Next, the four constraints are formally stated, where we assume (x_i^t, y_i^t) represents where droplet D_i is located at time t.

The fluidic constraint is used to prevent unexpected mixing between two droplets of different nets during droplet transportation. Then the static and dynamic fluidic constraints between different droplets D_i and D_j can be stated as follows:

$$\left|x_i^t - x_j^t\right| > 1 \quad \text{or} \quad \left|y_i^t - y_j^t\right| > 1 \tag{1}$$

$$\left|x_i^{t+1} - x_j^t\right| > 1 \quad \text{or} \quad \left|y_i^{t+1} - y_j^t\right| > 1$$

$$\text{or} \quad \left|x_i^t - x_j^{t+1}\right| > 1 \quad \text{or} \quad \left|y_i^t - y_j^{t+1}\right| > 1 \tag{2}$$

The timing constraint denotes the maximum allowed transportation time of a droplet from its source to target. The timing constraint is mainly used to ensure the bioassay's overall execution time. Typically, the shorter the routing paths of the droplets and the less waiting time for the droplets due to scheduling, the faster the bioassay execution time. Due to the high complexity of the simultaneous functional and washing droplet routing process, tight timing constraints may need to be relaxed for finding a feasible solution. Similar to previous works, the whole functional and washing droplet routing problem is partitioned into a series of subproblems [16]. Assume the maximum allowed transportation time is τ for all the functional and washing droplets in each subproblem. For any type of droplet D_i with source spot (x_i^S, y_i^S) and destination spot (x_i^D, y_i^D), the timing constraint is formulated as

$$\begin{aligned} \left(x_i^t, y_i^t\right) &= \left(x_i^S, y_i^S\right) \quad \text{for } t = 0 \\ \left(x_i^t, y_i^t\right) &= \left(x_i^D, y_i^D\right) \quad \text{for } t = \tau \end{aligned} \tag{3}$$

The contamination constraint is used to prevent cross-contamination between functional droplets. The liquid residue left by the first droplet should be washed away before the second droplet passes through the intersection spot. Therefore, the contamination constraint enforces the relative arriving times of the two functional droplets and the washing droplet at their intersection spot. Assume functional droplets D_1 and D_2 pass cross-contamination spot S_i at t_1 and t_2, respectively. Without loss of generality, assume $t_1 < t_2$. Assume a washing droplet passes S_i at t_w to wash the residue for avoiding cross-contamination. Then besides the fluidic constraints (1), (2), the functional and washing droplets should also satisfy the contamination constraint on S_i defined as

$$t_w > t_1 \quad \text{and} \quad t_2 > t_w \tag{4}$$

In real applications, the washing droplet gets dirty after several washing operations. Therefore, realistic washing capacity constraint needs to be considered, where the threshold is set for the washing droplets denoting the maximum allowed number

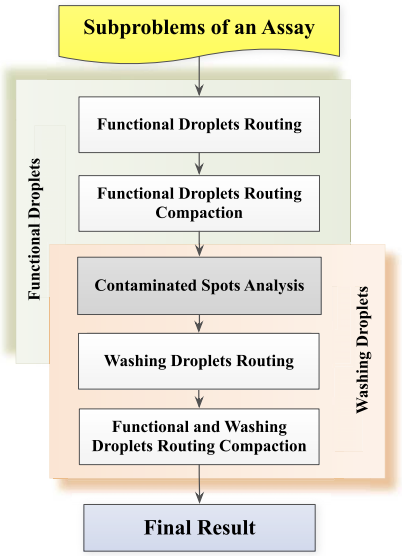


Fig. 3. Proposed functional and washing droplet routing flow.

of contaminated spots that a droplet could wash. Let ζ represent the washing capacity limit of a typical washing droplet. Assume a washing droplet washes N_o ordinary spots with residues and N_c cross-contamination spots before getting dirty. Then the realistic washing capacity constraint for the washing droplet is

$$N_o + N_c \leq \zeta \tag{5}$$

The contamination-aware functional and washing droplet routing problem of a DMFB can be formulated as follows.

Input: A list of nets to be connected, a set of washing droplets, a set of routing blockages, a set of reservoirs, the timing constraint, and the washing capacity constraint.

Objective: Compute the feasible routing and scheduling solution for all nets without violating the constraints, while minimizing the weighted sum of execution time, the number of cross-contamination spots, and the number of used cells for routing.[2]

Constraint: Fluidic constraint (1), (2), timing constraint (3), contamination constraint (4), and the capacity constraint of the washing droplet (5).

III. Algorithm Overview

Fig. 3 shows the overall flow of the proposed approach, which consists of five major steps: 1) functional routing; 2) functional droplets routing compaction; 3) cross-contamination spots analysis; 4) washing routing; and 5) functional and washing droplets routing compaction. In functional routing stage, we compute the routing paths for the nets from their source cells to their target cells, while minimizing the path length and the number of path intersections.

[2]The number of used cells should be minimized for better reliability, because each used cell needs to driven by the corresponding electrode. The less number of working electrodes, the less probability for functional errors and thus the better reliability. Here, functional errors refer to the wrong control logic either due to the errors in control pins or errors in the wires connecting electrodes to the control pins.

TABLE II
Notations Used in the Proposed Algorithms

Notations	Meaning
$\mathcal{D}$	List of functional droplets
D_i	The i^{th} functional droplet
$\mathcal{W}$	List of washing droplets
w_i	The i^{th} washing droplet
$\mathcal{S}$	List of cross-contamination spots
S_i	The i^{th} cross-contamination spot
t	The current clock cycle
$\mathcal{P}$	List of functional paths
P_i	The i^{th} functional path
T_c	Timing constraint for a subproblem

During functional droplet routing compaction, we propose an effective compaction algorithm to simultaneously schedule all the routing paths step by step, optimizing the overall execution time. The contaminated spots analysis step obtains the coordinates and the desired washing time-interval of each cross-contamination spot. Then, in the process of washing routing, we use the information of the cross-contamination spots to determine the washing order and compute the routing paths of the washing droplets. Then, we first apply a washing duration relaxation method to expand the lifetime of the cross-contamination spots without violating the specified timing constraint. Second, we propose the washing order decision technique to construct the routing paths for washing droplets, while considering the realistic washing capacity constraint. Finally, a routing compaction procedure is proposed to schedule all the functional and washing paths simultaneously for the final solution. The notations used in the following sections are given in Table II.

IV. Functional Routing and Compaction

During functional routing procedure, the routing paths for the set of nets are computed separately for each subproblem. Then the routing compaction procedure simultaneously schedules the routing paths. During functional routing, the number of path intersections needs to be minimized, because each intersection spot needs a washing droplet for the cleaning task. The less number of path intersection spots, the less washing tasks will be required. Therefore, the objective of functional routing is to find the routing paths with minimized lengths and number of intersections.

A. Functional Path Routing

In the proposed flow, the routing paths of functional droplets are first computed. The functional routing method is based on the classic A* searching algorithm (i.e., the Lee-style maze routing with the A* cost function). An A* search algorithm was proposed in [20], which allows for simultaneous motion of multiple droplets and thus is able to obtain globally optimal solution. However, the runtime may not be endurable for large designs due to the exponentially increasing solution space. As mentioned in Section II, timing constraint, fluidic constraint, and contamination constraint need to be observed. Although the functional droplets will be scheduled later to satisfy those

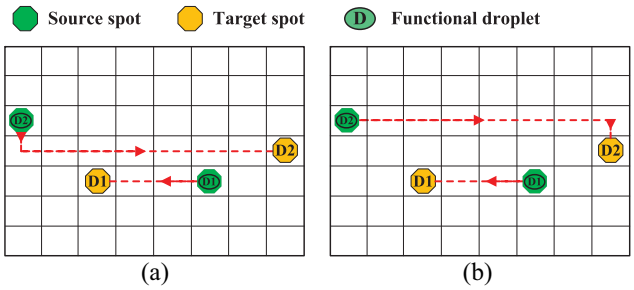


Fig. 4. Adjacent versus nonadjacent routing paths. (a) Due to adjacent routing paths, fluidic-constraint violation between droplets D_1 and D_2 cannot be resolved by droplet scheduling. (b) Using nonadjacent routing paths, there are no fluidic-constraint violations and no need for droplet scheduling.

constraints, good functional routing solutions will facilitate the scheduling process and help avoid constraint violations.

For fluidic constraint, droplets cannot be horizontally, vertically, or diagonally adjacent to each other at any time during transportation, except for those that they are expected to be mixed together. Rescheduling of the droplets (i.e., stalling one droplet to make way to the other droplet) may not always resolve the fluidic-constraint violations. We present to compute nonadjacent routing paths for different droplets to guarantee the fluidic constraint. Fig. 4 shows an example, where different droplet routing paths for droplet D_2 have different effects on droplet D_1. In Fig. 4(a), the two routing paths are adjacent to each other, which makes the fluidic-constraint violation between D_1 and D_2 unavoidable even with droplet scheduling. In Fig. 4(b), a different solution of D_2 obtains nonadjacent routing paths, which easily avoids the fluidic-constraint violation even without the need for droplet scheduling. To obtain nonadjacent routing paths, in the proposed routing method, we set the surrounding cells of routed paths as used. In this way, the A* searching algorithm will be encouraged to choose unused cells, which preferably computes nonadjacent droplet routing paths.

The timing constraint gives an upper-bound threshold on droplets' transportation time along their paths. This constraint is used to ensure the total execution time of an assay. During A* searching algorithm, those paths that violate the timing constraint are pruned away to avoid timing constraint violation. As a result, the proposed routing method will choose paths with used cells rather than long paths violating the timing constraint.

For avoiding violations to the cross-contamination constraint, it would be helpful to reduce the number of path intersections for saving the washing efforts. The proposed routing algorithm is modified to avoid path intersections as many as possible. For each already routed functional path, all the cells along the path are set as used. Then higher routing cost can be set to the used cells to avoid the path intersections. In A* searching algorithm, the routing cost of the current searching cell c_i is computed as follows:

$$F(c_i) = G'(c_i) + H(c_i)$$
$$G'(c_i) = G(c_i) + C_u \times U(c_i) \quad (6)$$

where $G(c_i)$ denotes the path length from the source cell to c_i, $H(c_i)$ denotes the estimated path length from c_i to the target cell, $U(c_i)$ is a binary (0/1) variable denoting whether c_i is set as used, and C_u is the user-defined parameter for the cost of selecting a used cell. Typically, C_u is set to be 4 for choosing a used cell, i.e., when the routing path has to detour more than four cells, it will prefer to choose a used cell instead.

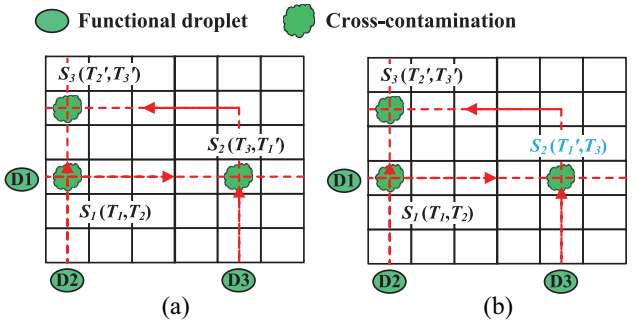


Fig. 5. Potential washing deadlock and path ordering for resolving the deadlock. (a) Washing deadlock without considering path ordering, i.e., any droplet may arrive earlier at the cross-contamination spot. (b) Washing deadlock is resolved by path ordering $D_1 < D_2 < D_3$, i.e., at the cross-contamination spots, D_1 is required to arrive earlier than D_2 and D_2 earlier than D_3.

B. Path Ordering

Cross-contamination occurs when different functional droplets pass the same cell. To successfully clean the cell at the cross-contamination spot, a washing droplet should arrive at the spot within the time interval between two sequentially arriving functional droplets. We call this time interval as washing duration for each cross-contamination spot, which represents the feasible washing interval for the washing operation.

Fig. 5 shows an example of a potential deadlock between the functional paths, where a feasible washing solution does not exist. In Fig. 5(a), there are three functional paths crossing each other at cross-contamination spots S_1–S_3, with corresponding washing durations (T_1, T_2), (T_3, T_1'), and (T_2', T_3'). The washing durations are computed according to the actual path lengths. For example, for cross-contamination spot S_2, functional droplet D_3 reaches the spot earlier than D_1, which results in the washing duration (T_3, T_1'), i.e., a washing droplet is needed to wash S_2 after D_3 passes through the spot and before D_1 reaches the spot. When the washing droplet cannot reach S_2 on time, we need to fall back and stall the latter droplet D_1. In congested designs, there may not be a good place for D_1 to stall halfway without violating the fluidic constraint. Therefore, the safe position to stall D_1 is at its source position. However, when we stall D_1 at its source position, cross-contamination spot S_1 will be affected. At S_1, a washing droplet is needed to wash the spot after D_1 passes the spot and before D_2 reaches the spot. As a result, D_2 also needs to fall back and stall at its source position, which in turn affects cross-contamination spot S_3. Then, to successfully wash S_3, D_3 needs to fall back at its source position, which will postpone the washing of S_2. In summary, the washing of S_2 affects S_1, S_1 affects S_3, and S_3 affects S_2, i.e., a deadlock is formed that cannot be resolved.

We propose a path ordering method to resolve the potential washing deadlocks. As shown in Fig. 5(b), the potential washing deadlock can be resolved by path ordering $D_1 < D_2 < D_3$,

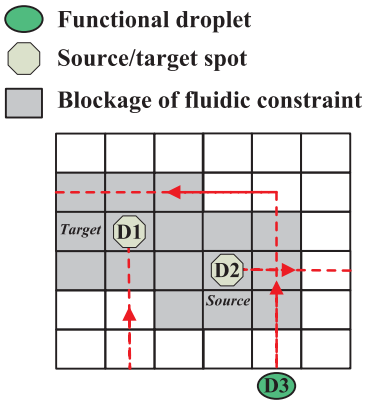


Fig. 6. Fluidic constraint for functional droplets at source/target positions.

Algorithm 1: Functional Path Order Computation Algorithm (Called in Algorithm 2)

Input: List of functional paths $\mathcal{P}$.
Output: The sorted functional paths.
1 Construct a directed acyclic graph *DAG* for paths $\mathcal{P}_1$ satisfying the path ordering rule;
2 Perform *topological sorting* on *DAG* to obtain an ordering of $\mathcal{P}_1$;
3 Sort the remaining paths in $\mathcal{P}_2 = \mathcal{P} - \mathcal{P}_1$ in non-ascending order of their lengths;
4 Perform *mergesort* on $\mathcal{P}_1$ and $\mathcal{P}_2$ according to their lengths.

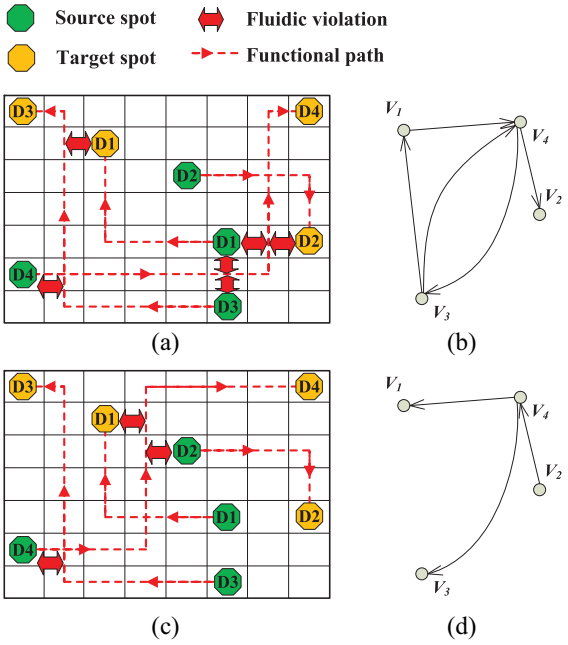


Fig. 7. Rip-up and rerouting for cycle removal in DAG. (a) Original functional paths. (b) Directed graph corresponding to (a). (c) Functional paths after rip-up and rerouting path of D_4. (d) DAG corresponding to (c) without cycles.

i.e., at any cross-contamination spot, D_1 is required to arrive earlier than D_2 and D_2 is required to arrive earlier than D_3. In this case, the washing duration of S_2 is changed to (T_1', T_3) because $T_1' < T_3$. Then stalling any droplet at its source position does not introduce deadlocks. In an extreme case, we may stall the droplets such that D_2 (D_3) waits at its source position until D_1 (D_2) reaches its target. Therefore, a valid washing solution is always guaranteed. Any path ordering solution along with the updated washing durations can be used to resolve such deadlocks.

Another issue that affects the droplet scheduling is the fluidic constraint on the source and target positions of the functional droplets. In each subproblem, source and target positions of functional droplets are typically located inside the 2-D biochip array. Therefore, fluidic constraint also needs to be satisfied for functional droplets at their source/target positions. Fig. 6 shows an example, where D_1 is located at it target position and D_2 is at its source position. The shaded cells denote the blockages caused by D_1 and D_2 according to the fluidic constraint. To avoid unexpected droplet mixing, D_3 cannot pass the shaded cells of D_2 unless D_2 leaves its source position first. Besides, D_3 cannot pass the shaded cells of D_1 unless D_1 stalls somewhere without reaching its target to let D_3 pass first. Therefore, we have the following path ordering rule: droplet A needs to be scheduled earlier than droplet B if any of the following conditions are satisfied: 1) A's source position blocks B's routing path and 2) B's target position blocks A's routing path.

When the functional paths are successfully computed, Algorithm 1 is proposed to sort all functional paths. First, the path ordering rule is examined for all the source/target positions of the functional droplets. Then, a directed acyclic graph DAG is constructed on the related paths as follows: when functional droplet D_1 needs to be scheduled earlier than functional droplet D_2, two nodes V_1 and V_1 will be added into DAG corresponding to the paths of D_1 and D_2, and a directed edge will be added from V_1 to V_2. Please note that it is possible to have cycles in the constructed graph. The following methods can be used to remove the cycles: 1) rip-up and rerouting based on the negotiation strategy [21], [22]; 2) routing concession method [7]; and 3) placement refinement based on virtual topology for deadlock-free routing solutions [23]. In the experiments, the rip-up and rerouting method successfully resolves all the cycles. Fig. 7 shows an example, where the constructed directed graph [Fig. 7(b)] for the original functional paths [Fig. 7(a)] contains cycles. To remove the cycles, we iteratively rip-up and reroute each functional path belonging to the cycles until they could be eliminated without introducing new cycles. To avoid obtaining the same routing path as the original one, the router sets the conflicting cells along the original path with larger routing cost. Fig. 7(c) shows a solution by rip-up and rerouting the path of D_4. During rip-up and rerouting, higher routing cost is set to cells along the original path having fluidic violations with D_1's source spot, D_2's target spot, and D_3's source spot. When the new path is computed as shown in Fig. 7(c), the new corresponding DAG without any cycle is shown in Fig. 7(d).

Next, topological sorting will be performed on DAG to obtain an ordering of paths $\mathcal{P}_1$ [24]. The remaining paths $\mathcal{P}_2$ are sorted according to their path lengths. The longer the path length is, the smaller the order is for the corresponding droplet. Finally, the two sorted list of functional paths are merged together according to their lengths by mergesort. The topological sorting algorithm on DAG(V, E) runs in time

Algorithm 2: Functional Path Ordering and Washing Duration Computation Algorithm

Input: Lists of functional paths $\mathcal{P}$ and cross-contamination spots $\mathcal{S}$.
Output: The scheduled paths with feasible washing durations at the cross-contamination spots.

```
1  Sort the paths in P by Algorithm 1;
2  Set each functional path P_i ∈ P a sorted order o_i;
3  while true do
4      Set ordered ← true;
5      for j = 1 to |S| do
6          Find first and second functional paths P_1 and P_2
           related to cross-contamination spot S_j;
7          Compute the arrival times T_1 and T_2 at S_j
           corresponding to P_1 and P_2;
8          Obtain the order values of P_1 and P_2 as o_1 and o_2,
           respectively;
9          if o_1 < o_2 and T_1 + 1 < T_2 then
10             Set the duration of S_j as (T_1, T_2);
11         else if o_1 > o_2 and T_1 > T_2 + 1 then
12             Set the duration of S_j as (T_2, T_1);
13             Switch between the first and second functional
               paths for S_j;
14             Set ordered ← false;
15         else
16             if o_1 < o_2 then
17                 Stall P_2 at its source position by T_1 − T_2 + 3;
18                 Set the duration of S_j as (T_1, T_1 + 3);
19             else
20                 Stall P_1 at its source position by T_2 − T_1 + 3;
21                 Set the duration of S_j as (T_2, T_2 + 3);
22                 Switch between the first and second functional
                   paths for S_j;
23             Set ordered ← false;
24     if ordered = true then
25         break;
```

Algorithm 3: Routing Compaction Algorithm

Input: List of functional paths $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ and timing constraint T_c.
Output: The scheduled paths $\mathcal{P}$ without violations.

```
1  Set loopFlag ← true;
2  Set n to be the number of functional paths in P (|P|);
3  while loopFlag do
4      Set restart ← false;
5      for t = 0 to T_c do
6          Move each droplet forward by one cell along its path;
7          if all the droplets reach their targets then
8              Set loopFlag ← false;
9              break;
10         for i = 0 to n do
11             for j = i + 1 to n do
12                 if P_i and P_j violate fluidic constraint at t then
13                     Set restart ← true;
14                     Call Algorithm 4 for P_i and P_j;
15             if restart = true then
16                 break;
17         if restart = true then
18             break;
19     if loopFlag = false then
20         break;
21     Reset the functional paths in P;
```

$O(|V| + |E|)$. DAG(V, E) is typically a sparse graph in the experiments, i.e., $|E| \approx |V|$. In line 3, functional paths $\mathcal{P}_2$ are sorted in $O(|\mathcal{P}_2| \cdot \log |\mathcal{P}_2|)$ time. Then in line 4, the one-pass mergesort procedure on $\mathcal{P}_1$ and $\mathcal{P}_2$ runs in $O(|\mathcal{P}_1| + |\mathcal{P}_2|)$ time. Therefore, the overall time complexity of Algorithm 1 is $O(|\mathcal{P}_2| \cdot \log |\mathcal{P}_2|)$.

When the functional paths and their related droplets are sorted in order, for the cross-contamination spots, we iteratively stall the droplet with larger order value to relax the washing durations. For any cross-contamination spot, we only allow the droplet with smaller order to pass the spot earlier. In this way, we can avoid the above-mentioned deadlocks and fluidic-constraint violations. In an extreme case, we can sequentially schedule functional droplets one by one according to their orders, and wash away all the cross-contamination spots of prior functional droplets before the latter functional droplet starts out. Therefore, the proposed path ordering method always guarantees a feasible washing solution.

Algorithm 2 shows our proposed functional path ordering and washing duration computation algorithm to compute the washing durations with the potential washing deadlocks avoided. The proposed algorithm first sorts the functional paths to obtain the orders, and then iteratively checks the washing duration of each cross-contamination spot. For each cross-contamination spot, the first and second droplets passing through the spot are checked according to their assigned orders. The corresponding washing duration is also examined. If there are any violations in the assigned order and/or washing duration, the function path with higher order value will be stalled. The iteration continues until all the cross-contamination spots are valid. As mentioned above, the sorting step by Algorithm 1 takes $O(|\mathcal{P}_2| \cdot \log |\mathcal{P}_2|)$ time. And in the worst case, the iterative checking on the cross-contamination spots takes $O(|\mathcal{S}|^2)$. Therefore, Algorithm 2 runs in $O(|\mathcal{P}_2| \cdot \log |\mathcal{P}_2| + |\mathcal{S}|^2)$ time.

C. Functional Path Compaction

When the droplets are sorted and scheduled, there may still be unexpected mixing between functional droplets. Therefore, we propose the compaction process to obtain the further scheduled solution for the movement of each droplet. At each time step, the droplet can either move forward one cell along the routing path or stall at the current cell. During the movements of the droplets, unexpected droplet mixing must be avoided. Furthermore, the overall execution time needs to be minimized to finish the bioassay as soon as possible. To achieve the above objectives, we propose a effective compaction algorithm to schedule all the routing paths simultaneously. Compared with the previous compaction approach [16], a new feature of our method is that the conflicts between droplets are resolved in a global manner.

Algorithm 3 shows our routing compaction algorithm. Our simultaneous approach checks the conflicts between droplets

Algorithm 4: Path Scheduling Algorithm (Called in Algorithm 3)

Input: List of functional paths $\mathcal{P}$, conflict paths P_i and P_j, and current clock t.
Output: The scheduled paths P_i' and P_j'.

```
1  Set id ← −1;
2  Set pos ← −1;
3  Compute the droplet order o_i and o_j for P_i and P_j, respectively;
4  if o_i < o_j and P_j[t] is not at source spot then
5      Set id ← j;
6  else
7      Set id ← i;
8  if id = i then
9      Set s_i ← 0;
10     for k = t − 1 to 1 do
11         if P_i[k] has no conflict with any other path then
12             Set s_i ← k;
13             break;
14     Set pos ← s_i;
15 else
16     Compute s_j for P_j similar to Lines 9-14;
17 Append 3 stalls to P_id at pos.
```

for each step of droplet movement. If fluidic-constraint violation occurs between two functional droplets, the one with larger droplet order value will chosen to fall back and wait (lines 4–7 in Algorithm 4). In Algorithm 4, a preferred stall position is computed such that it has no violations with any other functional paths, i.e., the stall of the droplet will not block in the way of other droplets. Therefore, the violations between droplets can be iteratively resolved.

Now we analyze the time complexity of the proposed algorithm. The outer loop counts t from 0 to T_c. The inner loops are used to check the fluidic constraint for each pair of droplets. Therefore, the time complexity of inner loops is $O(n^2)$, where n denotes the number of paths in the subproblem. The path scheduling method in Algorithm 4 takes $O(K)$ time, where K denotes the path length in the worst case. Furthermore, when we solve one conflict of two paths, the algorithm will be restarted. Assume that the number of restarts is n_r. Then the overall time complexity of the algorithm is $O(n_r \cdot T_c \cdot n^2 \cdot K)$.

V. Washing Droplet Routing

After the above-mentioned functional routing stage, a simultaneous washing droplet routing and compaction method is proposed to clean the cross-contamination spots. Unlike previous works, our washing droplet routing method considers the realistic washing capacity constraint and the routing conflicts with functional droplets. It is necessary to perform washing routing and functional routing simultaneously, because otherwise a separate washing routing stage after each functional droplet's routing will greatly increase the overall assay execution time. Moreover, disjoint paths are not always available, and often with large detouring path length. Therefore, simultaneous washing and functional routing is of great importance for avoiding cross-contamination and enhancing the performance of assay execution.

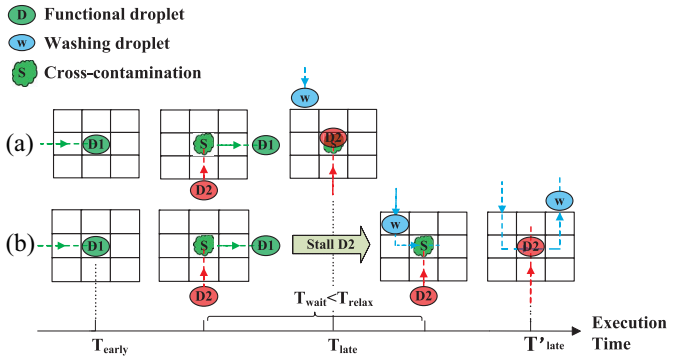


Fig. 8. Two functional droplets cross the same cell, forming cross-contamination spot S. (a) Washing droplet fails to clean the cross-contamination spot on time. (b) By stalling droplet D_2, the residue is successfully washed by w without droplet routing conflicts.

In the following sections, we first propose the washing duration relaxation method, which enlarges the feasible washing durations on the cross-contamination spots, thus facilitating the washing operations. Then we propose the washing droplet routing method, which determines the washing order on the cross-contamination spots and computes the corresponding washing paths, while satisfying the timing constraint and minimizing the total path length. After computing all the washing paths, we compact the washing paths together with the functional paths, where the arrival order of droplets at the cross-contamination spots is adjusted to successfully finish the washing tasks without contamination violations.

A. Washing Duration Relaxation

The initial washing duration for a cross-contamination spot after functional routing and compaction can be represented as follows:

$$T_{\text{washing}} = (T_{\text{early}}, T_{\text{late}}) \tag{7}$$

where T_{early} represents the arrival time of the first functional droplet (e.g., D_1 in Fig. 8), and T_{late} represents the arrival time of the second functional droplet (e.g., D_2 in Fig. 8). However, the washing droplets may not be able to finish the cleaning task within the designated washing duration. One possible reason is that the cross-contamination spot is too far away from the washing reservoir. To solve this problem, the algorithm in [15] seeks to relax the washing duration by delaying the arrival time of the latter functional droplet. However, the delayed functional droplet may violate the timing constraint. In this paper, we propose a washing duration relaxation method to guarantee the timing constraint. Let T_{used} be the time used for transporting the second functional droplet from its source cell to the target cell. Let T_c be the timing constraint. Then the maximum allowed relaxation time of the cross-contamination spot is $T_{\text{relax}} = T_c - T_{\text{used}}$. So the relaxed washing duration for a cross-contamination spot is computed as follows:

$$T_{\text{washing}}^{\text{relax}} = \left(T_{\text{early}}, T_{\text{late}} + T_{\text{relax}}\right) \tag{8}$$

Fig. 8 illustrates the washing duration relaxation process. In Fig. 8(a), functional droplets D_1 and D_2 arrive at cross-contamination spot S on time T_{early} and T_{late}, respectively.

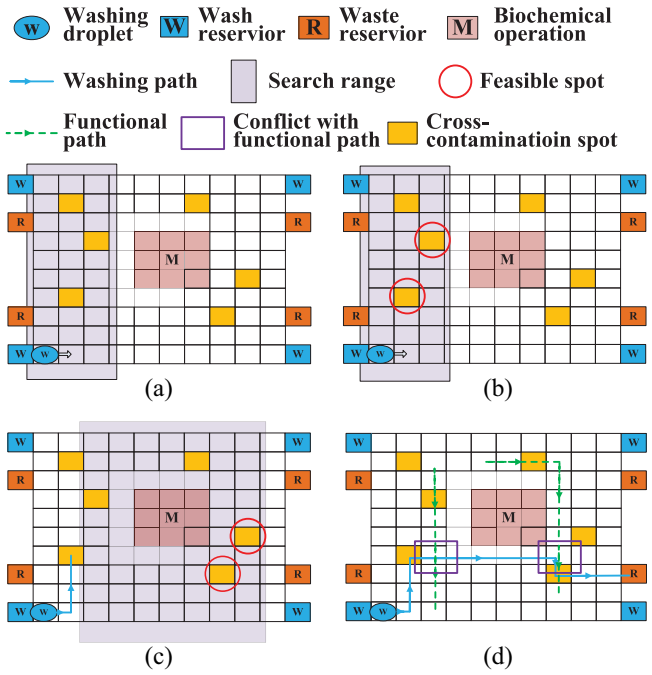


Fig. 9. Washing order decision method for washing droplet routing. (a) Washing droplet starts from the source with the searching range initialized. (b) Two feasible cross-contamination spots are found satisfying the washing duration. (c) Washing droplet moves to the best cross-contamination spot chosen from the candidates and a new searching operation starts. (d) Finish the washing path construction when the washing capacity constraint is met until reaching the biochip boundary.

But the washing droplet w fails to reach S in the washing duration T_{washing} to finish its cleaning task. So we need to stall D_2 before it arrives at S and let w wash away the residue first. In Fig. 8(b), T_{wait} is the time for stalling D_2, which should not exceed T_{relax}. After the adjustment, the new arrival time of D_2 is $T'_{\text{late}} > T_{\text{late}}$. Moreover, T'_{late} does not exceed $T_{\text{late}} + T_{\text{relax}}$ because $T_{\text{wait}} \leq T_{\text{relax}}$, which ensures the timing constraint. Thus, the relaxed washing duration for each cross-contamination spot facilitates the scheduling of the functional and washing droplets and avoids the timing constraint violation.

B. Washing Order Decision and Washing Path Computation

In real applications, after cleaning a certain number of cross-contamination spots, the washing droplet will become saturated and cannot wash anymore. Moreover, the washing droplet has to clean the cross-contamination spots within the required washing duration. Therefore, the washing order of the cross-contamination spots is critical, which determines the number of spots a washing droplet could successfully clean without violating the timing constraint and capacity constraint. We propose a method to compute the washing order and the washing paths concurrently.

Fig. 9 shows the washing droplet routing process. One washing droplet is dispensed from the wash reservoir. Then the feasible cross-contamination spots are searched in several neighboring columns (e.g., 3) of the biochip array [Fig. 9(a)]. Here, feasible cross-contamination spots refer to the spots with feasible relaxed washing durations that the washing droplet can reach in time. In Fig. 9(b), we obtain two feasible cross-contamination spots as candidates. Then one of these spot candidates is chosen as the washing target according to the following equation:

$$\text{Cost}_S = \alpha \cdot \frac{L}{L_c} + \beta \cdot \frac{T_{\text{early}}}{T_c} \tag{9}$$

where L represents the length of the routing path from the washing droplet's current position to the cross-contamination spot,[3] T_{early} means the arriving time of the first functional droplet as defined above, T_c means the timing constraint as defined above, and L_c means the designated length constraint. We assume the droplets move one cell at each clock cycle, and set L_c to be equal to T_c. The cross-contamination spot with the minimum cost Cost_S is chosen as the intermediate routing target [see Fig. 9(c)]. The intrinsic idea of (9) is to choose the cross-contamination spot both close to the current washing droplet's position and with small T_{early}. It is easy to understand that a close spot helps reduce the path length such that the spot does not need to wait long for washing. Moreover, the smaller T_{early} is, the earlier the contamination happens, and thus the washing droplet does not need to wait long to wash away the generated residue. In this case, after washing the spot, more time is left for the washing droplet to clean other cross-contamination spots. α and β are user-defined parameters, which are set to be 2 and 0.5 in the experiments, respectively.

As shown in Fig. 9(c), after one cross-contamination spot is determined along with the routing path, a new searching area (denoted as the shaded rectangle) is constructed to find the next set of feasible cross-contamination spots. This time the searching area could be modified to be larger according to the number of feasible candidates in the area. Besides, the crossings between the washing path and the existing functional paths are recorded. Such crossings may result in washing capacity consumption for the washing droplet. Thus, we need to subtract the consumption from the washing capacity. The searching and recording process is repeated until the biochip boundary is reached or the washing capacity is exhausted. Fig. 9(d) shows an example of a complete washing path from wash reservoir to waste reservoir. It has two routing conflicts with existing functional paths, where each conflict possibly consumes one washing capacity. Then a new washing droplet is dispensed from another wash reservoir in clockwise order to clean the remaining cross-contamination spots. The process is repeated until all the cross-contamination spots are finished.

The washing droplet routing algorithm is summarized in Algorithm 5. The algorithm iteratively dispenses washing droplets from the reservoirs for the cleaning task until no cross-contamination spots are left. First, we initialize the washing droplet and prepare to record its routing path (lines 3–6). Then, in line 7, we enter a for-loop to iteratively check the searching areas to wash as many feasible cross-contamination spots as possible. In line 8, the set of cross-contamination spots in current searching area are computed. In line 9, Algorithm 6

[3] A* routing algorithm (i.e., Lee-style maze routing with the A* cost function) is used to compute the routing paths of the washing droplet from its current position to the candidate cross-contamination spots.

Algorithm 5: Washing Droplet Routing Algorithm

Input: List of cross-contamination spots $\mathcal{S}$ with relaxed washing duration, and list of functional paths $\mathcal{P}_{\mathcal{F}}$.
Output: List of routing paths $\mathcal{P}_{\mathcal{W}}$ for the washing droplets.

1 **while** $\mathcal{S}$ *is not empty* **do**
2 Dispense a new washing droplet w_k from one of the reservoirs;
3 Initialize washing path P_k for w_k;
4 Set current spot S_c to w_k's current position;
5 Set next spot $S_n \leftarrow NULL$;
6 Set $o_1 \leftarrow -\infty$, $o_2 \leftarrow +\infty$;
7 **for** *Searching area* R_i *in searching order* **do**
8 Compute the set of testing cross-contamination spots $\mathcal{S}_{\mathcal{T}} = \{S_j | S_j \in R_i\}$;
9 Call Algorithm 6 to compute feasible spots $\mathcal{S}_{\mathcal{F}}$ from $\mathcal{S}_{\mathcal{T}}$ with the washing paths;
10 **if** $|\mathcal{S}_{\mathcal{F}}| = 0$ **then**
11 **continue**;
12 Compute the best spot from $\mathcal{S}_{\mathcal{F}}$ according to Equation (9) and assign it to S_n;
13 Obtain the washing path $P_{c,n}$ from S_c to S_n;
14 Accumulate w_k's washing capability consumption along path $P_{c,n}$;
15 **if** w_k*'s washing capacity is violated* **then**
16 **break**;
17 Find the first and second functional paths P_1 and P_2 related to S_n;
18 Set $o_1 \leftarrow max\{o_1, order(P_1)\}$, $o_2 \leftarrow min\{o_2, order(P_2)\}$;
19 Append path $P_{c,n}$ to the end of P_k;
20 Set $S_c \leftarrow S_n$;
21 Remove S_n from $\mathcal{S}$;
22 **if** P_k *is not empty* **then**
23 Route from S_c to one of the waste reservoir;
24 Append the routed path to the end of P_k;
25 Append P_k to $\mathcal{P}_{\mathcal{W}}$;

Algorithm 6: Feasible Cross-Contamination Spot Computation Algorithm (Called in Algorithm 5)

Input: Lists of testing cross-contamination spots $\mathcal{S}_{\mathcal{T}}$ and functional paths $\mathcal{P}_{\mathcal{F}}$, current spot S_c, and path orders o_1 and o_2.
Output: Lists of feasible cross-contamination spots $\mathcal{S}_{\mathcal{F}}$ and washing paths $\mathcal{P}'_{\mathcal{W}}$.

1 **for** $i = 1$ *to* $|\mathcal{S}_{\mathcal{T}}|$ **do**
2 Set cross-contamination spot $S_i \leftarrow \mathcal{S}_{\mathcal{T}}[i]$;
3 Find the first and second functional paths P_1 and P_2 related to S_i;
4 Set $o'_1 \leftarrow max\{o_1, order(P_1)\}$, $o'_2 \leftarrow min\{o_2, order(P_2)\}$;
5 **if** $o'_1 \geq o'_2$ **then**
6 **continue**;
7 Compute the washing path P_i from S_c to S_i;
8 Set *flag* $\leftarrow$ *true*;
9 **for** $j = 1$ *to* $|\mathcal{P}_{\mathcal{F}}|$ **do**
10 **if** P_j*'s source position violates fluidic constraint with* P_i *and* $order(P_j) > o'_1$ **then**
11 Set *flag* $\leftarrow$ *false*;
12 **break**;
13 **if** P_j*'s target position violates fluidic constraint with* P_i *and* $order(P_j) < o'_2$ **then**
14 Set *flag* $\leftarrow$ *false*;
15 **break**;
16 **if** *flag* $=$ *true* **then**
17 Schedule P_i according to the washing duration of S_i;
18 Append S_i to the end of $\mathcal{S}_{\mathcal{F}}$;
19 Append P_i to the end of $\mathcal{P}'_{\mathcal{W}}$;

is called to compute the feasible cross-contamination spots from the testing spots.

In Algorithm 6, the testing spots are iteratively checked. For each testing spot, we first check the compatibility in the related path orders. The idea is to iteratively squeeze the order values of the two related functional paths. In this way, we will be able to assign an order to the washing droplet without introducing deadlocks between washing and functional paths (see Section V-C). After checking the path orders, the washing path P_i is computed for spot S_i. Then in lines 8–15, fluidic constraints are checked between washing path P_i and the source/target positions of all functional paths. As stated in Section IV-B, the orders of the paths need to be sorted carefully to observe the fluidic constraint. The washing paths should follow the same path ordering rule. If P_i passes the checking process, it will be scheduled for the washing duration required at S_i. The scheduling method is similar to lines 8–17 in Algorithm 4. Finally, a valid cross-contamination spot along with the washing path is found and stored.

Then in line 12 of Algorithm 5, we choose the best destination from the feasible spots based on (9), and obtain the corresponding washing path. Next, the washing capacity consumption is computed and checked. If the washing path is valid, we will update the path order values, append the washing path to the end of washing path list, and delete the finished cross-contamination spot. Finally, the washing path to the waste reservoir is computed for discarding the washing droplet. Please note that when there are more than one washing droplet dispensed from the same reservoir, the latter washing droplet is delayed by two clock cycles to avoid unexpected droplet mixing.

Now we analyze the time complexity of Algorithm 5. We first sort the cross-contamination spots according their column indices. Therefore, to find the feasible cross-contamination spots, we only need to scan the columns sequentially in the designated searching area. Let w and h denote the width and height of the biochip array, respectively. And let $|\mathcal{S}|$ represent the number of cross-contamination spots. The time complexity of sorting and searching for feasible cross-contamination spots is $O(|\mathcal{S}|)$ using bucket sort. The routing paths of the washing droplet are computed using A* routing (Lee-style maze routing with the A* cost function), where the worst-case time complexity is $O(k \cdot w \cdot h)$. Here k represents the average number of routing paths for each cross-contamination spot. In Algorithm 6, the checking process for fluidic-constraint violations takes $O(|\mathcal{P}_{\mathcal{F}}|)$ time, and the path scheduling process for the washing paths takes $O(K)$ time, where K denotes the worst-case path length. In the worst case, each washing droplet can only clean one cross-contamination spot in its washing path, i.e., the algorithm will be finished in $|\mathcal{S}|$ rounds. Therefore, the overall time complexity for one subproblem is $O(|\mathcal{S}| \cdot k \cdot w \cdot h \cdot (K + |\mathcal{P}_{\mathcal{F}}|))$.

C. Simultaneous Functional and Washing Path Compaction

When the washing paths are computed, there may still be fluidic-constraint violations between washing and functional routing paths. Therefore, we perform a final compaction step on all the functional and washing paths. To avoid the deadlock problem mentioned in Section IV-B, we insert each washing path into the sorted functional paths with an order value between o_1 and o_2 computed in Algorithm 5. Then, all the functional and washing paths are sorted and each path has a new order. Finally, Algorithm 3 is called to compact all the paths simultaneously. When there are routing violations, the conflicting path (either functional or washing path) with the higher order value will be stalled. Besides, to guarantee the washing duration constraint for the cross-contamination spots, the washing feasibility is validated when the droplets reach those spots during clock forwarding from 0 to T_c. When the first functional droplet or washing droplet is stalled to make washing impossible, the latter droplet(s) will be stalled accordingly. The merits of having an order for each droplet is that, whenever a conflict occurs, we only need to select the path with higher order to stall without worrying about the deadlock issue.

Theorem 1: Using the proposed path ordering method, Algorithm 3 will always converge with a feasible functional and washing routing solution.

Proof: The proposed path ordering method first assigns each functional path an order. Then according to the washing relation, each washing path is also given an order value as follows. Assume the washing path P_k washes two cross-contamination spots S_1 and S_2. And assume the first functional path passing through S_1 is $P_{1,1}$ and the second one $P_{1,2}$. Similarly, assume the corresponding functional paths for S_2 are $P_{2,1}$ and $P_{2,2}$, respectively. Let the corresponding orders of the paths be o_k, $o_{1,1}$, $o_{1,2}$, $o_{2,1}$, and $o_{2,2}$, respectively. From Algorithms 5 and 6, we have $o_1 = \max\{o_{1,1}, o_{1,2}\}$ and $o_2 = \min\{o_{2,1}, o_{2,2}\}$. Therefore, the order of P_k is set to be o_k satisfying $o_1 < o_k < o_2$, i.e., P_k is inserted in between the functional paths without affecting the original sequential order. There are three cases when we stall a path: 1) if the path is the first one to pass some cross-contamination spots, all the related washing paths and second functional paths must have higher order values and need to be stalled; 2) if the path is a washing path, all the related second functional paths must have higher order values and need to be stalled; and 3) if the path is the second one to pass some cross-contamination spots, no related paths need to be stalled because it has the highest order value. Therefore, the deadlock shown in Fig. 5 will never occur. In an extreme case, the functional and washing droplets can be walked to their targets one by one without concurrency, which guarantees a feasible functional and washing routing solution. Therefore, by stalling the paths according to their order values, Algorithm 3 will always converge with a feasible solution. ■

Fig. 10 shows an illustrative example of the functional and washing droplet routing process for the example in Fig. 7(c). Assume the order of functional droplets is $(D_2 < D_4 < D_1 < D_3)$. And assume washing droplets w_1 and w_2 are computed to wash the cross-contamination spots of (D_4, D_3) and (D_4, D_1). According to Section V-C, a feasible order for all the droplets is $(D_2 < D_4 < w_1 < w_2 < D_1 < D_3)$. Fig. 10(a) shows the initial status with computed washing paths at time $t = 0$. At $t = 1$ [Fig. 10(b)], all the droplets are attempted to forward by one step. However, when moving forward w_1 by one step, there will be fluidic-constraint violations between w_1 and D_4. To resolve the violation between w_1 and D_4, we will stall w_1 with larger droplet order as shown in Algorithm 4. Therefore, we make three stalls for w_1 at the wash reservoir. All the remaining droplets are successfully transported by one step. Then at $t = 2$ [Fig. 10(c)], we

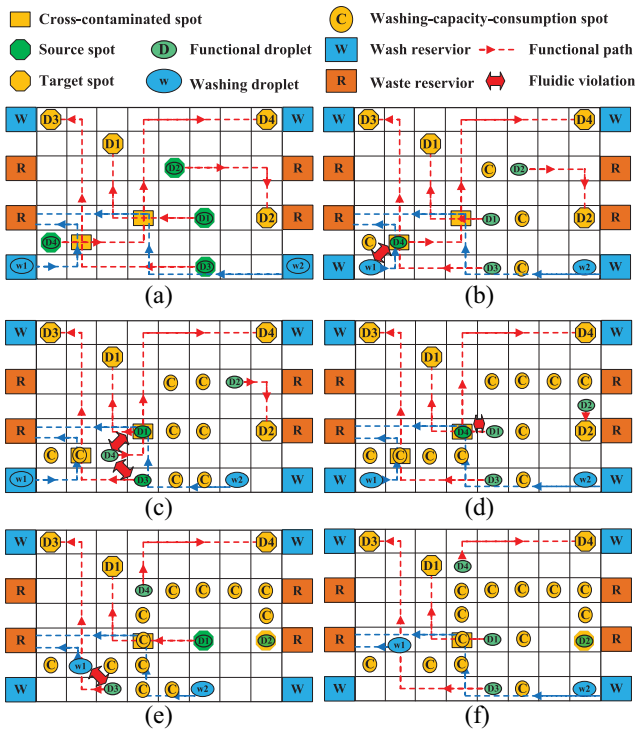


Fig. 10. Illustrative example. (a) Initial status with computed washing paths at time $t = 0$. Compaction at time (b) $t = 1$, (c) $t = 2$, (d) $t = 4$, (e) $t = 6$, and (f) $t = 7$.

TABLE III
COMPACTION RESULTS OF THE EXAMPLE IN FIG. 10. "1" REPRESENTS MOVING THE DROPLET FORWARD BY ONE STEP, AND "0" REPRESENTS STALL THE DROPLET AT CURRENT POSITION. TOTAL 22 STEPS ARE GIVEN, i.e., FROM $t = 1$ TO $t = 22$

Droplets	Control sequence
D_1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
D_2	1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
D_3	0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
D_4	1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
w_1	0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
w_2	0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0

TABLE IV
STATISTICS OF THE ROUTING BENCHMARKS

Circuit	Size	#Sub	#Net	#D_{max}	#Reservoir
in-vitro_1	16×16	11	28	5	4
in-vitro_2	14×14	15	34	6	4
protein_1	21×21	64	181	6	4
protein_2	13×13	68	137	6	4

TABLE V
COMPUTATIONAL SIMULATION RESULTS WITH VERSUS WITHOUT WASHING CAPACITY LIMIT

Bioassay	#Cont.	Without Capacity Limit							With Capacity Limit						
		$\#\mathcal{W}_{vio}$	$\#\mathcal{W}$	Error	$\mathcal{S}_{fail}$	#UC	T_r	CPU	$\#\mathcal{W}_{vio}$	$\#\mathcal{W}$	Error	$\mathcal{S}_{fail}$	#UC	T_r	CPU
in-vitro_1	31	7	8	88%	0	548	293	0.02	0	31	0%	14	571	444	0.03
in-vitro_2	34	7	10	70%	0	617	351	0.03	0	34	0%	7	582	432	0.04
protein_1	69	19	23	83%	0	3042	1591	0.08	0	69	0%	9	3438	1724	0.13
protein_2	75	11	25	44%	0	2040	1336	0.05	0	75	0%	7	2046	1318	0.08
Total	209	44	66	67%	0	6247	3571	0.18	0	209	0%	37	6637	3918	0.29

attempt to move all droplets by one step except w_1, which is stalled by three steps. However, when moving forward D_1 by one step, there will be fluidic-constraint violation with D_4. To resolve the violation between D_1 and D_4, we will stall D_1 with larger droplet order. Therefore, we make three stalls for D_1 at its source spot. Each time the fluidic-constraint violation occurs, one of the droplets will be stalled and another loop will be restarted (see Algorithm 3). In another round of the compaction loop, we will make three stalls for D_3 at its source spot to avoid the fluidic-constraint violation with D_4. Because D_3 is stalled at its source spot, in future compaction steps at $t = 2$, w_2 will also be stalled at its source spot due to the violation with D_3. At $t = 4$ [Fig. 10(d)], all the droplets are attempted to forward by one step. Due to the fluidic-constraint violation with D_4, D_1 is stalled for another three steps. Then at $t = 6$ [Fig. 10(e)], a fluidic-constraint violation occurs again between w_1 and D_3. As a result, D_3 will be stalled again, and w_2 will also be stalled accordingly. At $t = 7$ [Fig. 10(f)], washing droplet w_1 successfully washes the cross-contamination spot for D_3. In the following compaction steps, D_1 will be stalled several times until w_2 passes the cross-contamination spot to observe the contamination constraint (see Section V-C). Table III shows the final scheduling results for all the droplets.

VI. COMPUTATIONAL SIMULATION RESULTS

We have implemented our integrated functional and washing droplet routing flow in C++ on a 2.60 GHz 32-core Intel Xeon Linux workstation with 132 GB memory. Only a single thread is used for the experiments. Four commonly used bioassays are tested to verify our approach. Table IV shows the details of the benchmarks, where "Size" represents the size of DMFB array, "#Sub" gives the number of subproblems, "#Net" gives the number of nets, "$\#D_{\max}$" records the maximum number of droplets within one subproblem, and "#Reservoir" denotes the number of wash reservoirs. In the experiments, the washing capacity constraint for each washing droplet is set to be 4. Besides, to fully test the performance of the proposed washing flow, we allow the functional paths to have intersections between each other.

In the first experiment, we compute the number of washing droplets violating the capacity constraint. This experiment verifies the importance of washing capacity constraint and the effectiveness of our method. Table V shows the comparison results of our routing flow with versus without the washing capacity constraint. "#Cont." gives the number of cross-contamination spots, "$\#\mathcal{W}_{\text{vio}}$" gives the number of washing droplets that conducted the washing task with violated capacity constraint, "$\#\mathcal{W}$" gives the total number of used washing droplets, "Error" gives the error rate calculated by "$\#\mathcal{W}_{\text{vio}}/\#\mathcal{W}$," "$\mathcal{S}_{\text{fail}}$" gives the number of cross-contamination spots that fail to be washed, "#UC" gives the number of used cells for routing, "T_r" gives the execution time for bioassays (i.e., the number of clock cycles), and "CPU" gives the CPU time in seconds (s). The results show that this paper is effective with significant improvement, which reduces all the error rates to 0. Without the capacity constraint, overall there are 67% invalid washing droplets violating the capacity constraint. Using our algorithm, all the washing operations are valid within the capacity limit. From the results, there are also some cross-contamination spots that fail to be washed. In those cases, there are so many functional paths in the way blocking the washing droplet and consuming its capacity that the washing capacity is exhausted before reaching the cross-contamination spot. In such cases, a larger washing droplet could be adopted to wash the congested cross-contamination spots (see Fig. 11 for details).

TABLE VI
COMPARISON RESULT BETWEEN [16] AND OUR METHOD

Bioassay	[16]				Our Method Without Capacity Limit			
	#Cont.	#UC	T_r	CPU	#Cont.	#UC	T_r	CPU
in-vitro_1	21	351	193	0.58	31	548	293	0.02
in-vitro_2	5	281	191	0.39	34	617	351	0.03
protein_1	82	2213	1394	2.58	69	3042	1591	0.08
protein_2	61	1362	1108	1.49	75	2040	1336	0.05
Total	169	4207	2886	5.04	209	6247	3571	0.18

In the second experiment (Table VI), we compare our approach (the capacity limit is removed) with state-of-the-art contamination-aware droplet routing method in [16], which does not consider the washing capacity constraints. The method in [16] seems to perform better than our proposed washing droplet routing method. That is because the minimum cost circulation problem formulation is used to schedule optimal and correct wash operations. However, the problem we are addressing in this paper is much more difficult than the one in [16]. In our problem: 1) washing droplets have realistic washing capacity constraints and 2) functional and washing droplets are transported simultaneously, while the realistic washing capacity consumptions are considered for all residues along the path (i.e., not only residues at the intersection sites as in [16]). The problem is so difficult that there is not an easy way to modify the method in [16] and formulate our problem as a minimum cost circulation problem. Based on the above considerations, the overhead (i.e., number of used cells and the execution time) is reasonable. Besides,

TABLE VII
DIAGONAL SEARCHING VERSUS HORIZONTAL SEARCHING

Bioassay	#Cont.	Our Method (Diagonal)							Our Method (Horizontal)						
		$\#\mathcal{W}_{vio}$	$\#\mathcal{W}$	Error	$\mathcal{S}_{fail}$	#UC	T_r	CPU	$\#\mathcal{W}_{vio}$	$\#\mathcal{W}$	Error	$\mathcal{S}_{fail}$	#UC	T_r	CPU
in-vitro_1	31	0	6	0%	21	481	443	0.08	0	9	0%	14	571	444	0.03
in-vitro_2	34	0	2	0%	32	420	460	0.09	0	12	0%	7	582	432	0.04
protein_1	69	0	6	0%	59	2362	2330	0.40	0	40	0%	9	3438	1724	0.13
protein_2	75	0	21	0%	32	1846	1666	0.16	0	32	0%	7	2046	1318	0.08
Total	209	0	35	0%	144	5109	4899	0.73	0	93	0%	37	6637	3918	0.29

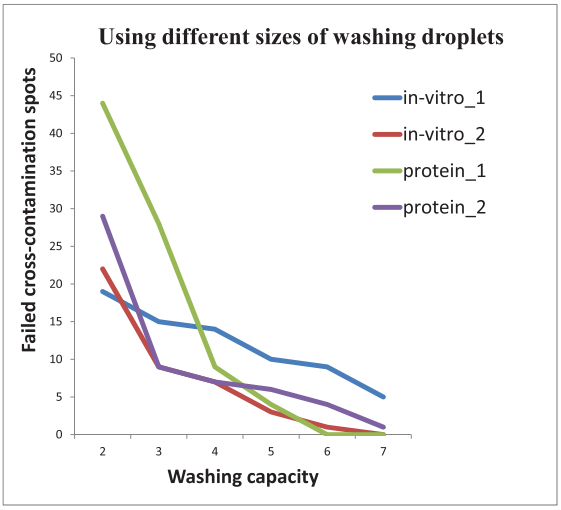


Fig. 11. Computational simulation results on different sizes of the washing droplets.

the runtime of our method is much faster with 28× speedup in average.

In the third experiment, we compare two approaches of constructing the washing paths. The first method finds the washing paths by diagonal searching. That is, the next destination spot is found for washing droplets in the diagonal direction in the 2-D biochip array. The second method (our proposed method) finds the washing paths by horizontal searching. That is, the next destination spot is found in the horizontal direction. The results in Table VII show that the horizontal searching has a better performance in CPU time than diagonal direction. This is because the horizontal searching has a smaller searching range in each step and thus is more efficient than the diagonal one. Moreover, the horizontal searching method results in fewer failed cross-contamination spots. We attribute this to the fact that horizontal searching method has more flexibility in the y-axis (i.e., searching both up and down) than the monotone diagonal searching. Since congested cross-contamination spots are generated during functional routing, the merits of the additional searching flexibility in horizontal searching method become notable.

Fig. 11 shows the computational simulation results using different sizes of washing droplets. From the figure, using a larger washing droplet, i.e., with larger washing capacity, the cross-contamination spots are more likely to be successfully washed away. However, with small washing droplets, some cross-contamination spots fail to be successfully washed. This is because the fact that it is usual for some functional paths to surround a specific cross-contamination spot and consume certain number of washing capacity before the washing droplet could reach the spot. In such cases, it is possible to use multiple small washing droplets to wash a single cross-contamination spot. But that would result in delayed execution time. Therefore, this paper proposes to use large washing droplets to perform the washing tasks for congested cross-contamination spots. From the figure, all the cross-contamination spots in benchmarks protein_1 and in-vitro_2 can be successfully washed with a larger washing droplet of washing capacity 7. For a washing droplet with washing capacity greater than 7, the washing droplet would be so large that it will take multiple electrodes in space. That is left for future works.

VII. CONCLUSION

Existing works in droplet routing have oversimplified assumptions on the washing droplet's behavior and constraints. Other simple solutions, such as following each functional droplet immediately with a washing droplet, consume much more washing resource and also greatly worsen the execution time. This paper proposes the first integrated droplet routing flow, which not only considers the realistic washing capacity constraint, but also resolves the routing conflicts between washing and functional droplets. Real-life biochemical applications validate the proposed flow's effectiveness and efficiency. Besides, washing droplets of different sizes are also introduced and experimented with notable improvements. Our future work includes new methods for using extremely large washing droplets, which take multiple electrodes to drive.

REFERENCES

[1] Q. Wang, Y. Shen, H. Yao, T.-Y. Ho, and Y. Cai, "Practical functional and washing droplet routing for cross-contamination avoidance in digital microfluidic biochips," in *Proc. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2014, pp. 1–6.

[2] K. Chakrabarty and F. Su, *Digital Microfluidic Biochips*. Boca Raton, FL, USA: CRC Press, 2007.

[3] R. B. Fair *et al.*, "Chemical and biological applications of digital-microfluidic devices," *IEEE Design Test Comput.*, vol. 24, no. 1, pp. 10–24, Jan./Feb. 2007.

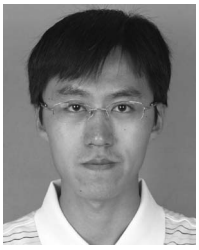
[4] V. Srinivasan, V. K. Pamula, and R. B. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab Chip*, vol. 4, no. 4, pp. 310–315, 2004.

[5] I. Barbulovic-Nad, H. Yang, P. S. Park, and A. R. Wheeler, "Digital microfluidics for cell-based assays," *Lab Chip*, vol. 8, no. 4, pp. 519–526, 2008.

[6] V. Srinivasan, V. K. Pamula, P. Paik, and R. B. Fair, "Protein stamping for MALDI mass spectrometry using an electrowetting-based microfluidic platform," in *Proc. SPIE 5591, Lab-on-a-Chip: Platforms Devices Appl.*, 2004, pp. 26–32.

[7] M. Cho and D. Z. Pan, "A high-performance droplet routing algorithm for digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1714–1724, Oct. 2008.

[8] F. Su and K. Chakrabarty, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2005, pp. 825–830.
[9] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proc. Design Autom. Test Europe (DATE)*, Munich, Germany, 2006, pp. 1–6.
[10] T. Xu and K. Chakrabarty, "Integrated droplet routing in the synthesis of microfluidic biochips," in *Proc. Design Autom. Conf.*, San Diego, CA, USA, 2007, pp. 948–953.
[11] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "BioRoute: A network-flow-based routing algorithm for the synthesis of digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 11, pp. 1928–1941, Nov. 2008.
[12] P.-H. Yuh, S. S. Sapatnekar, C.-L. Yang, and Y.-W. Chang, "A progressive-ILP-based routing algorithm for the synthesis of cross-referencing biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 9, pp. 1295–1306, Sep. 2009.
[13] M. Campàs and I. Katakis, "DNA biochip arraying, detection and amplification strategies," *TrAC Trends Anal. Chem.*, vol. 23, no. 1, pp. 49–62, 2004.
[14] Y. Zhao and K. Chakrabarty, "Cross-contamination avoidance for droplet routing in digital microfluidic biochips," in *Proc. Design Autom. Test Europe (DATE)*, Nice, France, 2009, pp. 1290–1295.
[15] Y. Zhao and K. Chakrabarty, "Synchronization of washing operations with droplet routing for cross-contamination avoidance in digital microfluidic biochips," in *Proc. Design Autom. Conf.*, Anaheim, CA, USA, 2010, pp. 635–640.
[16] T.-W. Huang, C.-H. Lin, and T.-Y. Ho, "A contamination aware droplet routing algorithm for the synthesis of digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 11, pp. 1682–1695, Nov. 2010.
[17] C. C.-Y. Lin and Y.-W. Chang, "Cross-contamination aware design methodology for pin-constrained digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 6, pp. 817–828, Jun. 2011.
[18] Y. Zhao and K. Chakrabarty, "Cross-contamination avoidance for droplet routing in digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 6, pp. 817–830, Jun. 2012.
[19] D. Mitra, S. Ghoshal, H. Rahaman, K. Chakrabarty, and B. B. Bhattacharya, "On residue removal in digital microfluidic biochips," in *Proc. Great Lakes Symp. VLSI*, Boston, MA, USA, pp. 391–394, 2011.
[20] K. F. Böhringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 2, pp. 334–344, Feb. 2006.
[21] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. ACM Symp. Field-Program. Gate Arrays*, Napa Valley, CA, USA, 1995, pp. 111–117.
[22] H. Yao, T.-Y. Ho, and Y. Cai, "PACOR: Practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6.
[23] D. T. Grissom and P. Brisk, "Fast online synthesis of digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 3, pp. 356–369, Mar. 2014.
[24] (Nov. 8, 2015). *Boost C++ Libraries*. [Online]. Available: http://www.boost.org/



Hailong Yao (SM'15) received the B.S. degree in computer science and technology from Tianjin University, Tianjin, China, in 2002, and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2007.

From 2007 to 2009, he was a Post-Doctoral Research Scholar with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA, USA. He joined the Department of Computer Science and Technology, Tsinghua University, as an Assistant Professor in 2009. He has published over 40 research papers. His current research interests include computer-aided design for microfluidic biochips, very large-scale integration physical design, design-manufacturing interface, and analog routing.

Prof. Yao was a recipient of two best paper award nominations at IEEE/ACM International Conference on Computer-Aided Design (ICCAD) in 2006 and 2008, and a best paper award nomination at IEEE International Symposium on Quality Electronic Designs (ISQED) in 2011.

Qin Wang received the B.S. degree in software engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing.

His current research interests include physical design and design automation for microfluidic biochips.

Yiren Shen received the B.S. degree in electronic engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently pursuing the M.S. degree in computer engineering with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA.

His current research interests include the computer-aided design for very large-scale integration circuits and high performance computing.

Tsung-Yi Ho (SM'12) received the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2005.

He is a Professor with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. From 2007 to 2014 and in 2015, he was with National Cheng Kung University, Tainan, Taiwan, and National Chiao Tung University, Hsinchu. He has presented eight tutorials and contributed eight special sessions in ACM/IEEE conferences, all in design automation for microfluidic biochips. His current research interests include design automation and test for microfluidic biochips and nanometer integrated circuits.

Prof. Ho was a recipient of the Best Paper Award at the Very Large Scale Integration Test Symposium in 2013 and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS in 2015. He currently serves as an ACM Distinguished Speaker, a Distinguished Visitor of the IEEE Computer Society, the Chair of the IEEE Computer Society Tainan Chapter and ACM SIGDA Taiwan Chapter, and an Associate Editor of *ACM Journal on Emerging Technologies in Computing Systems*, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and a Guest Editor of the IEEE DESIGN AND TEST OF COMPUTERS.

Yici Cai received the B.S. degree in electronic engineering and the M.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 1983 and 1986, respectively, and the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2007.

She has been a Professor with the Department of Computer Science and Technology, Tsinghua University. Her current research interests include design automation for very large-scale integrated circuits algorithms and theory, power/ground distribution network analysis and optimization, high performance clock synthesis, and low power physical design.