# Physical Co-Design of Flow and Control Layers for Flow-Based Microfluidic Biochips

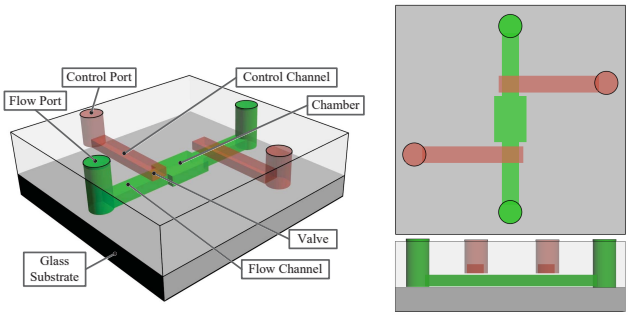
Qin Wang Hao Zou Hailong Yao *Senior Member, IEEE*
Tsung-Yi Ho *Senior Member, IEEE* Robert Wille *Senior Member, IEEE*
Yici Cai *Senior Member, IEEE*

***Abstract*—Flow-based microfluidic biochips are attracting increasing attention with successful applications in biochemical experiments, point-of-care diagnosis, etc. Existing works in design automation consider the flow-layer design and control-layer design separately, lacking a global optimization and hence resulting in degraded routability and reliability. This paper presents a novel integrated physical co-design methodology, which seamlessly integrates the flow-layer and control-layer design stages. In the flow-layer design stage, a sequence-pair-based placement method is presented, which allows for an iterative placement refinement based on routing feedbacks. In the control-layer design stage, the minimum cost flow formulation is adopted to further improve the routability. Besides that, effective placement adjustment strategies are proposed to iteratively enhance the solution quality of the overall control-layer design. Experimental results show that compared with the existing work, the proposed design flow obtains an average reduction of 40.44% in flow-channel crossings, 31.95% in total chip area, and 22.02% in total flow-channel length. Moreover, all the valves are successfully routed in the control-layer design stage.**

***Index Terms*—Flow-based microfluidic biochips, Flow-layer design, Control-layer design, Physical co-design**

## I. Introduction

The past decade has seen tremendous advances in flow-based microfluidic biochips, which automates the traditional laboratory procedures in molecular biology and biochemistry [2]–[4]. Noticeable merits of microfluidic biochips over traditional laboratory analysis platforms include (1) greatly saving the assay cost by reducing expensive samples/reagents to nano-liter or pico-liter volumes, (2) seamlessly integrating the automatic control logic for reduced human intervention and labor cost, (3) significantly increasing the sensitivity and accuracy of the biochemical assay, (4) potentially facilitating the portability characteristics for point-of-care diagnostic devices, and (5) naturally enabling the ultra-small chamber for single-cell capture and culture. As an example, microfluidic biochips have been developed for cancer detection and HIV/AIDS diagnosis [5], [6].




(a) 3D view. (b) Top and side views.

Fig. 1: Schematic of a flow-based microfluidic biochip.

Figure 1 shows the schematic of a flow-based microfluidic biochip based on the *multilayer soft lithography* technology, where the microfluidic functional units are fabricated by elastomer material (polydimethylsiloxane, PDMS) [7]. The 3D view of a demonstrative chamber for a cell culture is shown in Figure 1(a). The corresponding top and side views of the same chamber are given in Figure 1(b). On top of the glass substrate, there are two PDMS layers: (1) the *flow layer* that is composed of channels transporting the fluids used in the assay, and (2) the *control layer* with control channels that imposes the respectively desired control logic. Both control and flow channels are connected to input/output ports, which are punched holes through the PDMS material. A *microvalve* (referred to as *valve* hereafter) is located between the control and flow channels at their crossing region. To switch a valve, an external *pressure source* is injected through the *control pins* (also called *control ports*). Then hydraulic or pneumatic pressure ($\sim$10 psi) will be conducted through the control channel to the valve, squeezing the elastomer down into the flow channel to seal the fluidic flow. When the external pressure is released, the elastomer of the valve will restore back, and transportation of the fluid will resume due to the external pressure from the *flow port*. By controlling the actuation patterns on valves in a programmed way, fluidic operations are executed automatically [8].

The valve is a critical functional unit in flow-based biochips, which is a basic building block to form complex fluid-handling functional units (also called *components*), such as mixers, switches, and multiplexers [8], [9]. Figure 2 shows a biochip example with a *mixer* and a flow-channel crossing, where

A preliminary version of this work appeared in [1]. Extensions beyond [1] include the control-layer design algorithms along with the physical co-design method for flow-based microfluidic biochips, new placement adjustment strategies for control-layer design to minimize the failed valves, a detailed description of the proposed methods, as well as comprehensive experimental evaluations.

Q. Wang, H. Zou, H. Yao, and Y. Cai are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (E-mail: woodythu@163.com, ahio@163.com, hailongyao@tsinghua.edu.cn, caiyc@tsinghua.edu.cn).

T.-Y. Ho is with the Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan. E-mail: tyho@cs.nthu.edu.tw.

R. Wille is with the Institute for Integrated Circuits, Johannes Kepler University Linz, Austria. E-mail: robert.wille@jku.at.

Copyright © 2017 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.

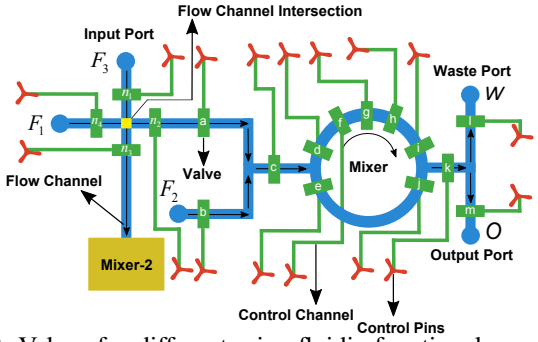



Fig. 2: Valves for different microfluidic functional operations.

valves are used for different functional operations [4], [9], [10]. In the figure, samples/reagents are dispensed from the input ports $F_1$, $F_2$, and $F_3$. For mixing two types of fluids, the following major steps are executed: (1) Inject the fluidic input from $F_1$ to the lower half mixer. (2) Inject the fluidic input from $F_2$ to the upper half mixer. (3) Seal the mixer and start the mixing operation. (4) Transport the mixed fluid from the lower half mixer to output port $O$. (5) Transport the mixed fluid from the upper half mixer to waste port $W$. These steps can be realized, e.g., by actuating the valves of this chip as follows:

1) Inject fluidic input from $F_1$ to the lower half mixer.
   - Open valves: $n_2$, $n_4$, $a$, $c$, $e$, $j$.
   - Closed valves: $n_1$, $n_3$, $b$, $d$, $i$, $k$.
   - Don't-care valves[1]: $f$, $g$, $h$, $n$, $l$, $m$.
2) Inject fluidic input from $F_2$ to the upper half mixer.
   - Open valves: $b$, $c$, $d$, $f$, $g$, $h$, $i$.
   - Closed valves: $a$, $e$, $j$, $k$.
   - Don't-care valves: $n_1$, $n_2$, $n_3$, $n_4$, $l$, $m$.
3) Seal the mixer and start the mixing operation.
   - Open valves: $d$, $e$, $i$, $j$.
   - Closed valves: $c$, $k$.
   - Don't-care valves: $n_1$, $n_2$, $n_3$, $n_4$, $a$, $b$, $l$, $m$.
   - Switching valves: $f$, $g$, $h$.
4) Transport the mixed fluid from the lower half mixer to the output port.
   - Open valves: $b$, $c$, $e$, $j$, $k$, $m$.
   - Closed valves: $a$, $d$, $i$, $l$.
   - Don't-care valves: $n_1$, $n_2$, $n_3$, $n_4$, $f$, $g$, $h$.
5) Transport the mixed fluid from the upper half mixer to the waste port.
   - Open valves: $b$, $c$, $d$, $f$, $g$, $h$, $i$, $k$, $l$.
   - Closed valves: $a$, $e$, $j$, $m$.
   - Don't-care valves: $n_1$, $n_2$, $n_3$, $n_4$.

Note that, in Figure 2, the mixer consists of a *micropump*, which is a combination of three valves $f$, $g$ and $h$. By actuating the three valves in a peristaltic sequence at a high frequency ($\sim$100 Hz), the two types of fluids are forced to circulate around for mixing [10]. In flow-based microfluidic biochips, air in the PDMS microchannels can be pushed through a closed valve, but the fluid cannot bypass a closed valve. For example, when injecting the fluidic input from $F_1$ to the lower half part of the mixer, valve $k$ can be closed without affecting the desired functionality. A typical flow-based microfluidic biochip may integrate hundreds or even thousands of valves, which motivates automated design methods to realize the desired functionality and reduce the turnaround time.

On the control layer, valves are controlled by the hydraulic or pneumatic pressure injected from the control pins through control channels. The mapping from valves to control pins is called *valve addressing*. One of the typical microvalve addressing method is *direct addressing*, where valves map to control pins in a one-to-one correspondence. As shown in Figure 2, a typical mixer has 9 valves. Using the direct-addressing scheme, these valves are connected to 9 different control pins by 9 independent control channels. Besides, for each flow-channel intersection/crossing, four extra valves are needed for multiplexing (see Figure 2). As the control channels between valves and control pins are routed on a single control layer, the huge number of valves makes it extremely difficult to route all the channels without crossings. Since channel crossings are not allowed on the control layer, the huge number of valves and unfavorable valve positions may cause routing failures on the control layer. Therefore, to improve the solution quality of control-layer routing, we should reduce the number of valves (by reducing the flow-channel crossings), and place the valves to favorable positions (by refining the flow-layer component placement). Then, the reduced valves yield a reduction of the needed control pins. In this work, we propose a co-design methodology for both the flow and control layers, which effectively incorporates these feedbacks and, by this, iteratively refines the overall design quality.

In the past decade, design automation tools have been introduced to deal with design challenges of biochips [11], and noticeable research progress has been made in the development of automated design methods for digital microfluidic biochips [12]–[16]. For flow-based microfluidic biochips, placement/routing iterations are performed in [17] to reduce flow-channel crossings, and control-layer optimization is investigated in [18]. In [19], [20], distributed storage is investigated during synthesis. Furthermore, dynamic construction of devices and flow channels on a fully programmable valve array is explored in [21], [22]. Fault models and test of manufacturing defects for flow-based biochips are discussed in [23]. In fact, the design of flow-based biochips is mainly conducted manually thus far. As discussed in [24]–[28], this does not only delay product deployment, but also hinders the exploration of the design solution space defined by the current fabrication technology. Moreover, the design challenges for both, the flow layer and the control layer, are usually considered separately, which frequently leads to unsatisfactory results. All this motivates the development of automated design methods for flow-based microfluidic biochips.

The first routing algorithm for flow-based microfluidic biochips was proposed based on the min-cost max-flow formulation [29]. Minhass et al. proposed a synthesis method for minimizing the number of control pins [30]. Hu et al. proposed a routing method, which minimizes both the number of control pins and the pressure-propagation delay [10]. Architectural synthesis and resource binding methods have been proposed in [31]–[33]. Lin et al. proposed a routing

[1]A don't-care valve is a valve whose actuation status (either open or closed) does not influence the prescribed execution of an assay.

algorithm minimizing the weighted sum of the maximum and total channel lengths [34]. Tseng et al. proposed a top-down synthesis method for flow-based microfluidic biochips considering valve-switching minimization [35]. McDaniel et al. proposed a simulated annealing-based placement algorithm for flow-based biochips [36]. The proposed algorithm assumes two-dimensional rectangular grids for placing the components, which greatly reduces the solution space and thus degrades the placement quality. The first co-design concept for flow-based microfluidic biochips has been proposed in [37]. However, its placement adjustment strategies for both flow layer and control layer are not so effective, eventually leading to a large number of flow-channel crossings as well as an increased chip area. The SAT-based method which actually can minimize the number of flow-channel crossings has been proposed in [17]. However, the method is not that scalable and, hence, not applicable for large designs. Furthermore, its method does not consider the control-layer design. Overall, existing works cannot obtain satisfactory design solutions with guaranteed solution quality for flow-based microfluidic biochips.

In this work, we proposed a physical co-design flow, which addresses the problems and shortcomings discussed above. The proposed methodology generates satisfactory designs with a significantly enhanced solution quality with respect to flow-channel crossings, control-routing success rate, total chip area, total channel length, etc. Major contributions of the paper are as follows:

- A novel co-design methodology for flow-based microfluidic biochips is proposed, which seamlessly integrates the flow-layer and the control-layer design stages. In particular, valves are identified as the bridge between the flow layer and the control layer, which carries the control-layer routing feedbacks to the flow-layer design. Thus, interactions are induced between the two layers allowing for an iterative design optimization.
- At the flow-layer design stage, an effective placement algorithm is adopted, which is based on the state-of-the-art sequence-pair representation. Moreover, according to the routing feedbacks, an efficient polynomial-time placement adjustment algorithm is proposed to incrementally adjust the placement solution for enhanced routing quality, where the relative positions of the components are reserved.
- At the control-layer routing stage, an effective bounding-box-based placement adjustment algorithm is proposed to further adjust the component placement allowing for an improved control-layer routing solution using the direct-addressing scheme.

The remainder of the paper is organized as follows. Section II presents the background and problem formulation. Section III presents the overall design flow of the proposed method. Section IV presents the details of the flow-layer design stage with the placement adjustment method for improving the flow-layer routing solution. Section V gives the details of the control-layer design method and further placement adjustment strategies for improving the control-layer routing solution. Section VI presents and discusses the experimental results. Finally, a conclusion is drawn in Section VII.

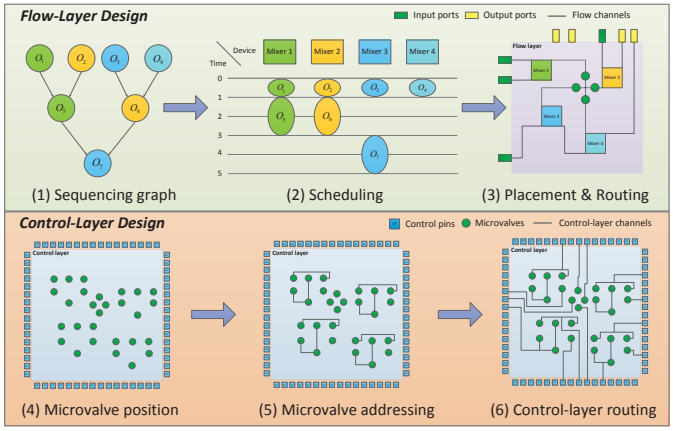



Fig. 3: Regular design flow for flow-based microfluidic biochips [37].

## II. Background and Problem Formulation

Figure 3 shows the regular design automation framework for flow-based microfluidic biochips, which consists of six design steps: (1) sequencing graph generation, (2) scheduling and resource binding, (3) flow-layer placement and routing, (4) computation of the microvalves' positions, (5) microvalve addressing, and (6) control-layer routing. In the overall design framework, valves are the critical components that closely couple the two major stages, i.e., flow-layer design and control-layer design. Inferior flow-layer placement and routing results force valves to be placed at unfavorable positions, which will cause great trouble to the control-layer design, or even result in control-layer routing failures.

Figure 4 provides an illustration of the problem to be solved in this paper. More precisely:

**Given:** The list of components, the nets to be routed, and the design rules.

**Find:** Enhanced placement and routing solution for both flow layer and control layer.

**Objective:** Minimize the weighted cost of: (1) chip area, (2) number of control pins, (3) total flow-channel length, and (4) total control-channel length.

**Subject to:** (1) Minimum component spacing design rule, (2) minimum flow/control channel width design rules, and (3) minimum flow/control channel spacing design rules.

## III. Overall Flow

Figure 5 presents the overall flow of the proposed co-design method, which consists of two major stages, i.e., flow-layer design and control-layer design. In the proposed co-design flow, placement adjustment is the major co-design module, which integrates the two design stages. In the flow-layer design stage, device/component placement and flow-channel routing are performed, where flow-channel crossings are allowed. However, as flow-channel crossings introduce new valves and control pins, which increase the control-layer design complexity, flow-channel crossings need to be minimized. In the control-layer design stage, a direct addressing scheme is adopted, and control-layer routing is performed to obtain the overall design solution.

The placement adjustment module accepts routing feedbacks and iteratively adjusts component placement for further

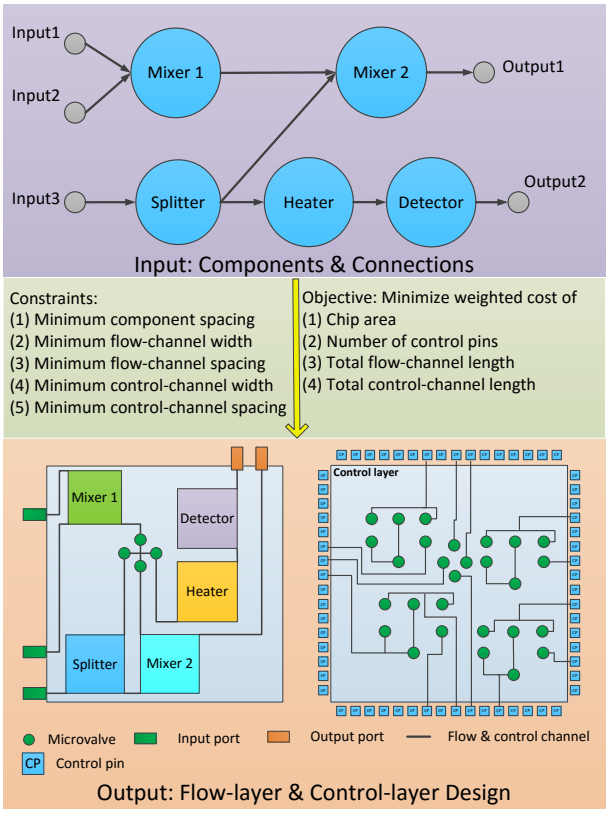



Fig. 4: Problem formulation of flow-control design automation.

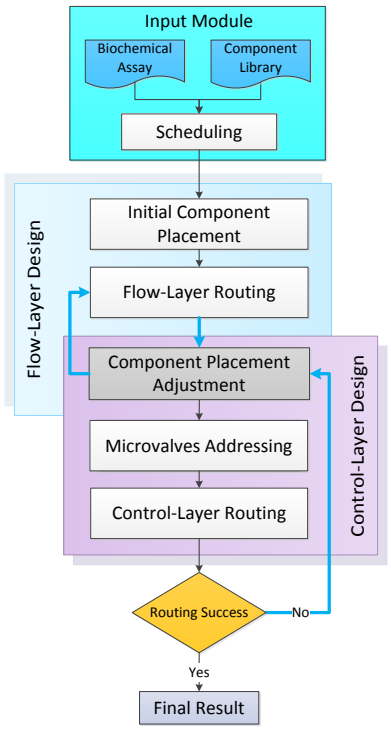



Fig. 5: Overall flow of the proposed co-design method.

improvement. There are two parts of routing feedbacks: (1) the feedback from flow-layer routing, such as the flow-channel crossings and the total flow-channel length, as well as (2) the feedback from control-layer routing, such as the number of failed valves[2] in routing and the total control-channel length. Using the proposed iterative adjustment mechanism, the overall solution quality can be significantly improved.

In the overall co-design flow, the component placement adjustment module is responsible for the co-design functionality between the flow-layer design and the control-layer design. The valves act as the bridge between the two layers. When the control-layer design fails (i.e., at least one failed valve occurs), placement adjustment will be carried out to incrementally shift the components' positions such that more routing resources surrounding the failed valves are granted for a successful routing. The iterated procedure of placement adjustment and control-layer routing effectively eliminates the failed valves, and results in an overall successful design.

## IV. Flow-Layer Design

Figure 6 provides an illustration of the problem to be addressed during the flow-layer design stage. The objective is to minimize the weighted cost of chip area, number of flow-channel crossings, and total flow-channel length. Besides that, there is another implicit optimization objective, i.e., to improve the control-layer design quality, which is considered during the placement adjustment. Figure 7 gives the design flow of our corresponding placement and routing method for the flow-layer design stage. There are three major steps, i.e., component placement, flow-layer routing, and placement adjustment. The input for the flow-layer design is the scheduling result [38], including the list of components, the nets to be routed, and the design rules. With these inputs, the placement process computes positions of the components by a simulated annealing algorithm, where the placement solution is interpreted by state-of-the-art sequence-pair representation (see Figure 8). To enforce the minimum spacing design rule while routing the flow-channels on the same layer, necessary spacings are assigned to each component. More precisely, each component is expanded to make more room for a valid routing. In the placement method, the objective is to minimize the weighted sum of total chip area, the number of estimated line segment crossings corresponding to the nets, and the total Manhattan length of the nets.[3]

After placement, the flow-channels are computed for each net during the routing stage. As flow-layer routing is performed on a single layer, the routing resources are very limited. Therefore, the routing order of the nets is a critical factor in determining the routing quality. To address this issue, a negotiation-based routing method is adopted to obtain enhanced routing solutions with minimized channel crossings and total channel length.

The placement and routing stages determine the number of valves and even the overall design quality. Inferior placement and routing results will increase the number of flow-channel crossings and unnecessary valves. The increased number of valves, however, will increase the total control-channel length,

[2]A failed valve is a valve that, during the control-layer routing process, fails to get connected to its corresponding control pin.

[3]During the placement process, the nets (i.e., the set of terminals to be interconnected by the flow-channels) are not routed yet. Therefore, line segments between the pins of placed modules are used for predicting the potential crossings and wirelength.

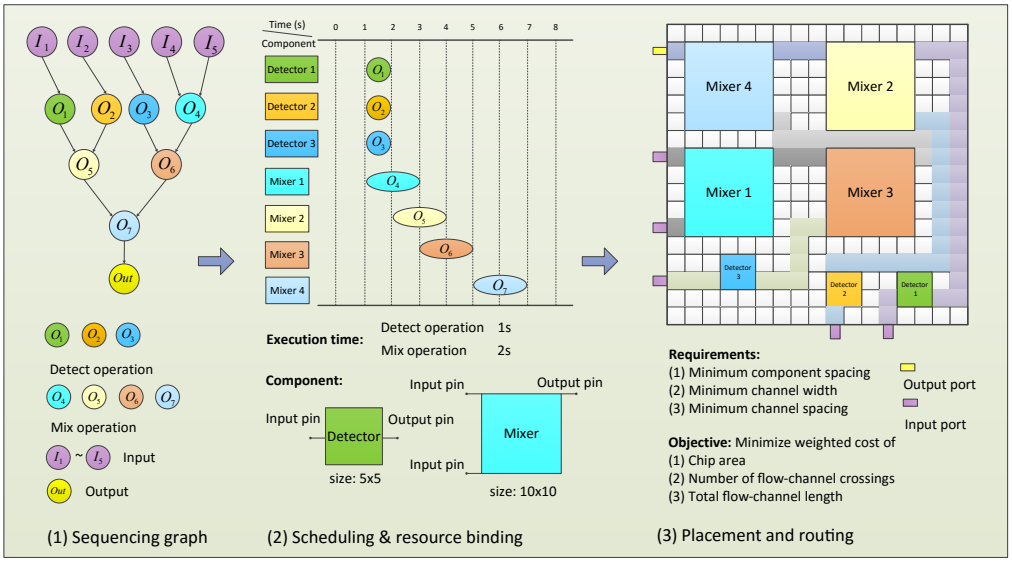



Fig. 6: Problem formulation of flow-layer design.

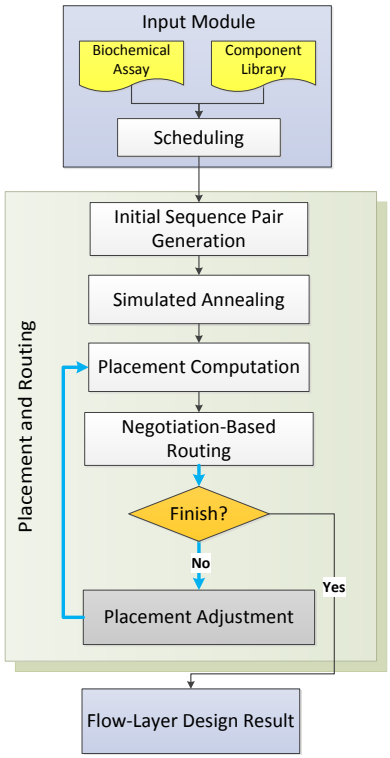



Fig. 7: Flow-layer design flow.

the number of needed control pins, and may even lead to design failures. Therefore, flow-layer placement and routing are critical for the overall design quality.

After routing, the routing solution is checked for over-congested regions. Then placement adjustment is performed to fine-tune the placement result for eliminating the congested regions. When a congested region is identified, the spacings between the corresponding components will be expanded in such a way that the corresponding sequence-pair keeps unchanged. Therefore, the new placement result can easily be obtained by evaluating the same sequence-pair once again. The iterative placement adjustment process continues until there is no over-congested region in the routing solution. Without over-congested regions, the routing process tends to find an optimized solution with less routing detours and channel crossings. Using the above sequence-pair-based placement and the iterative improvement mechanism, the overall solution quality is significantly improved. The next subsections provide the details of the proposed placement algorithm.

### A. *Placement Method*

In this subsection, we describe the proposed flow-layer component placement algorithm, where the sequence-pair representation is adopted for encoding and enumerating the solution space, and simulated annealing is used to search for an optimized placement solution.

*1) Sequence-Pair Representation:* We adopt the state-of-the-art sequence-pair (*SP*) representation for component placement [39]. The SP representation has a significant impact to the electronic design automation (EDA) field. A large body of works along this research thread has been published, which are mostly on evaluation of the sequence-pair representation [40], [41], and extending the representation with more constraints for different applications, such as the alignment, abutment, and symmetry constraints [42]. However, the intrinsic sequence-pair representation remains unchanged. The SP representation provides a coding scheme for the placement solutions of the given components, which essentially is a pair of permutations of the components. SP enables a solution space called *P-admissible* [39], which satisfies the following requirements:

1) The solution space is finite.
2) Every solution is feasible.
3) The realization of an SP code is possible in polynomial time.
4) There exists an SP code that corresponds to one of the optimal solutions.

Figure 8 shows an example illustrating the sequence-pair representation. Assume that there is a given set of components, e.g., $M = \{a,b,c,d,e,f\}$, and assume the SP code is $(s_1,s_2)$. As shown in Figure 8, we assume $(s_1, s_2)$ = (*ecadfb*, *fcbead*). The *Rule* for interpreting the SP code is as follows: (1) if symbol $a$ is in front of symbol $b$ in both the two sequences $s_1$ and $s_2$, then component $a$ is to the left side of component

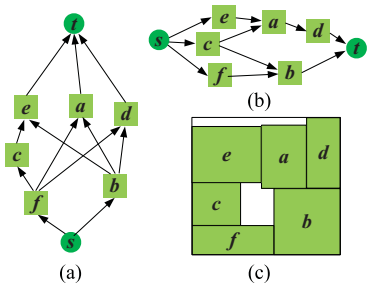



Fig. 8: Sequence-pair representation example with given SP code (*ecadfb*, *fcbead*): (a) vertical constraint graph, (b) horizontal constraint graph, and (c) placement result.

$b$ in the placement result, and (2) if symbol $a$ is in front of symbol $b$ in sequence $s_1$, and in back of symbol $b$ in sequence $s_2$, then component $a$ is to the top side of component $b$ in the placement result. Given the rules for interpreting the SP code, two directed acyclic graphs (DAGs) can be constructed according to the relative positions of the components, namely a *horizontal constraint graph* for relative left/right positions and a *vertical constraint graph* for relative top/down positions. The graphs are constructed as follows: (1) represent each component as a node in the graphs, (2) if component $a$ is to the left side of component $b$, then add a directed edge between the corresponding nodes from $n_a$ to $n_b$ into the horizontal constraint graph, (3) if component $a$ is to the bottom side of component $b$, then add a directed edge between the corresponding nodes from $n_a$ to $n_b$ into the vertical constraint graph, (4) pseudo source and target nodes are added to the two graphs denoting the placement boundaries, and (5) remove the redundant edges that can be computed from other directed edges by transitivity. Figure 8(a) and Figure 8(b) give the horizontal and vertical constraint graphs corresponding to the SP code.

Figure 8(c) give the corresponding placement result of the SP code ($s_1$ = *ecadfb*, $s_2$ = *fcbead*). For further explaining the interpretation of the SP code, since symbol $f(c)$ is behind symbol $e(c)$ in $s_1$ and in front of $e(c)$ in $s_2$, according to Rule (2), component $f(c)$ is to the bottom side of component $e(c)$. The vertical constraint graph in Figure 8(a) shows the relative positions of $f(c)$ and $e(c)$. Moreover, symbol $e(a)$ is in front of symbol $d(a)$ in both sequences $s_1$ and $s_2$. Therefore, according to Rule (1), component $e(a)$ is to the left side of component $d(a)$ as denoted in the horizontal constraint graph in Figure 8(b). The relative positions of other components in either horizontal or vertical directions can be derived similarly. Thus, based on Rule (1) and Rule (2), the horizontal and vertical constraint graphs can be constructed from the SP code. On the constructed constraint graphs, we assign edge weights to be 0 and node weights to be the sizes of the corresponding components (i.e., the component width as node weight for the horizontal constraint graph, and the component height as node weight for the vertical constraint graph). Then the locations of the placed components and the bounding box of the placement area can be obtained from the two constraint graphs, i.e., by computing the longest paths from the pseudo source node $s$ to the specific components in the constraint graphs. Further details of the sequence-pair representation can be found in [39].

**Input:** The sizes of the components and the nets for interconnection.
**Output:** Placement results of all the components.

Let $ex_i$ and $ey_i$ be the expanded spacings on the left side and bottom side of component $m_i$, respectively;
Let $EX$ and $EY$ be the vector of expanded spacings;
Randomly generate initial $EX$ and $EY$, where $e_{min} \leq ex_i, ey_i \leq e_{max}$;
Randomly generate initial $SP = (s_1, s_2)$;
Let placement state $ST = (s_1, s_2, EX, EY)$;
Set initial temperature $T \leftarrow T_0$;
**while** $T > T_{min}$ **do**
  **for** $round = 1$ *to* $R$ **do**
    Set $ST' = (s_1', s_2', EX', EY') \leftarrow ST$;
    Randomly generate binary (0/1) value $b$;
    **if** $b = 0$ **then**
      Randomly choose component $m_i$;
      Randomly choose $ex_i'$ or $ey_i'$, and randomly increase or decrease it by one unit;
      **if** $ex_i' < e_{min}$ **then**
        Set $ex_i' \leftarrow e_{min}$;
      **if** $ex_i' > e_{max}$ **then**
        Set $ex_i' \leftarrow e_{max}$;
      Set $ey_i'$ to be within $[e_{min}, e_{max}]$ similarly as $ex_i'$;
    **else**
      Randomly choose $s = s_1'$ or $s_2'$;
      Randomly swap two components $m_i$ and $m_j$ in $s$;
    **if** $E(ST') < E(ST)$ **then**
      Set $ST \leftarrow ST'$;
    **else**
      Set $p_0 \leftarrow e^{\frac{E(ST)-E(ST')}{T}}$;
      Randomly generate float value $p \in [0, 1]$;
      **if** $p < p_0$ **then**
        Set $ST \leftarrow ST'$;
  Set $T \leftarrow T \times R_r$;

**Algorithm 1:** Simulated annealing-based placement algorithm.

*2) Simulated Annealing:* Based on the sequence-pair representation, we propose a placement method using classic simulated annealing. Algorithm 1 gives the proposed component placement algorithm. After the scheduling process, the nets for component interconnections and sizes of the components can be obtained as input for the placement algorithm. As routing is performed on a single layer, necessary routing resources need to be reserved between the placed components in order to successfully route the channels. Therefore, during the placement process, we expand each component $m_i$ by a reserved spacing $ex_i$ and $ey_i$, whereby $ex_i$ is the expanded spacing on the left side of $m_i$, and $ey_i$ is the expanded spacing on the bottom side of $m_i$. Then two spacing vectors $EX$ and $EY$ are constructed for all the components. In Algorithm 1, we first randomly generate an SP code $(s_1, s_2)$ and spacing vectors ($EX$, $EY$). Each value in the spacing vectors ($EX$, $EY$) is constrained by $[e_{min}, e_{max}]$. Then $s_1$, $s_2$, $EX$, and $EY$ are combined as the initial placement state $ST$. Next, the temperature loop is entered with the initial temperature $T_0$ and temperature reduction rate $R_r$. For each temperature loop, the placement solution is iteratively optimized for $R$ rounds. In each round, a new placement state $ST'$ is obtained by randomly swapping a pair of components in one sequence ($s_1$ or $s_2$), or by fine-tuning the spacing vectors ($EX$ and $EY$) of the components. In the experiments, the following parameters have been employed: $e_{min} = 3$, $e_{max} = 5$, $T_0 = 10000$, $T_{min} = 10^{-4}$, $R_r = 0.95$, and $R = 200$. To guarantee the efficiency of the placement procedure, flow-layer routing is not actually performed in the simulated annealing algorithm.

Instead, an estimation function (namely Equation (1), which is introduced next) is proposed to assess the placement quality in terms of area, the number of estimated channel crossings, and the estimated Manhattan wirelength of the nets. After the simulated annealing algorithm, flow-layer routing will be actually performed, and placement adjustment will be carried out to further improve the routing solution.

The objective of the proposed placement algorithm is to minimize the weighted sum of the total placement area ($A$), the estimated number of net crossings ($C$), and the estimated total channel length ($L$). Therefore, our energy function to evaluate a placement state $ST$ in Algorithm 1 is defined as

$$E(ST) = \alpha \cdot A + \beta \cdot C + \gamma \cdot L + \theta \cdot L_2 \quad (1)$$

where $A$ represents the area of the minimum bounding box of all placed components, and $C$ represents the total number of crossings between line segments corresponding to the nets. By minimizing $C$ during placement, we potentially minimize the number of channel crossings during routing. $L$ is the sum of Manhattan distances of the nets. $L_2$ is the sum of square of Manhattan distances corresponding to the nets, which is used to avoid extra-long paths. In the experiments, the following parameters have been employed: $\alpha = 1$, $\beta = 300$, $\gamma = 20$, and $\theta = 0.001$.

Now we analyze the time complexity of Algorithm 1. The temperature loop will execute for $log_{R_r}^{T_0/T_{min}}$ (i.e., $\frac{log^{T_0} - log^{T_{min}}}{log^{R_r}}$) times. For each temperature loop, the inner loop executes for $R$ times. The placement perturbation and spacing modification take $O(1)$ time. The computation of $E(ST)$ in Equation (1) takes $O(n^2+m^2)$, where $n$ is the number of components, and $m$ is the number of nets. Therefore, the overall time complexity is $O(\frac{log^{T_0} - log^{T_{min}}}{log^{R_r}} \times R \times (n^2+m^2))$.

### B. Placement Adjustment

In the flow-layer design stage, the spacing between the placed components needs to be carefully tuned in order to obtain an improved flow-layer routing solution. In this subsection, a polynomial-time placement adjustment method is proposed, which incrementally expands the components for increasing the routing resources between components. Placement adjustment is performed after flow-layer routing. After one-round of placement adjustment, flow-layer routing is performed once again. The iteration is finished when all the flow channels are successfully routed without an over-congested routing area.

Algorithm 2 provides in detail the placement adjustment algorithm, which is invoked after each round of routing. The intrinsic idea is that, whenever the routed channels are over-congested, more routing resources should be assigned and the corresponding components need to be pushed away from there. In this way, the congested regions are iteratively removed by placement adjustment. By this, the final routing solution is significantly improved. The time complexity of Algorithm 2 is $O(W \times H)$. Details of flow-layer routing method are presented in the following subsection.

### C. Flow-layer Routing and Routing Feedback

This subsection describes details of our negotiation-based routing method and the routing feedback for incremental placement adjustment. Note that channel crossings are allowed during the flow-layer routing.

**Input:** Flow-layer placement and routing results.
**Output:** Modified placement solution.
1 Assume $W$ and $H$ are width and height of the biochip, respectively;
2 **for** $y = 0$ *to* $H-1$ **do**
3  Set $flag \leftarrow true$;
4  **for** $x = 0$ *to* $W-1$ **do**
5  **if** *grid* $(x,y)$ *is empty* **then**
6  Set $flag \leftarrow false$;
7  **if** *grid* $(x,y)$ *is on the left boundary of component* $m_i$ **then**
8  **if** $flag = true$ **then**
9  Set $ex_i \leftarrow ex_i + 1$;
10  Set $flag \leftarrow true$;
11 Scan the routing grids in vertical direction and expand $ey_i$ for all the components similarly;
12 Re-compute the placement solution based on the same sequence-pair.

**Algorithm 2:** Placement adjustment algorithm.

*1) Negotiation-Based Flow-Layer Routing:* In the routing stage, the objective is to find feasible routing results with minimized channel crossings and total channel length. Due to the single routing layer, the routing order of the nets has great impact on the routing quality. To optimize the routing solution, we adopt the enhanced routing method using a negotiation strategy [43], [44].

Algorithm 3 shows the negotiation-based routing algorithm. In contrast to the original global routing method considering routing congestions [43], this routing method directly considers the routability of the routing grids. Therefore, a different cost function for history cost is defined for each routing grid $g$, namely

$$C_h(g)^{r+1} = C_b + \lambda \cdot C_h(g)^r \quad (2)$$

where $C_h(g)^{r+1}$ is the current history cost of the routing grid $g$ for iteration $r+1$, $C_b$ is the base history cost, $C_h(g)^r$ is the history cost in iteration $r$, and $\lambda$ is a user-define parameter. In the experiments, $C_b$ is set to be 1.0, and $\lambda$ is set to be 0.1.

In Algorithm 3, the history costs for the routing grids are first initialized to be 0. Then an obstacle map *ObsMap*, which is a two-dimensional array of Boolean values, is constructed for the routing grids. Next, a Boolean flag *done* and an integer counter $r$ are initialized for the following negotiated iterative routing process. In *Step* 5, $P_o$ is initialized for storing the best routed paths. Then a negotiation-based iterative routing loop is entered. In *Steps* 9-15, all the nets are routed one by one using a modified A* searching algorithm. If the routing is successful, the routed path will be inserted into $P$, and routing grids along the routed path will be set as obstacles in *ObsMap*. Otherwise, the iteration flag *done* will be reset as *false* for further iteration. Here, the history routing cost $C_h$ is critical in the negotiation-based routing method, where iterative optimization considering the routing history helps to avoid routing congestion. This relieves the well-known net-ordering issue in the one-by-one routing procedure. In *Steps* 16-17, the best routed paths are recorded. Then in *Steps* 18-20, the iteration counter $r$ is increased by 1. If the number of iterations exceeds the user-defined threshold $\tau$, the while-loop will be terminated. In the experiments, $\tau$ is set to be 10. In *Steps* 21-23, when the number of iterations does not exceed the threshold and there are failed nets, the history

**Input:** Placement result and set of nets $N$.
**Output:** Routing result.
1 Initialize history cost $C_h$ for the routing grids;
2 Construct *ObsMap* for routing obstacles on the routing grids;
3 Set flag *done* $\leftarrow$ *false*;
4 Set counter $r \leftarrow 0$;
5 Initialize $P_o$ to store the best routed paths;
6 **while** *done* $\neq$ *true* **do**
7  Set *done* $\leftarrow$ *true*;
8  Initialize $P$ to store the routed paths;
9  **for** $i = 1$ *to* $|N|$ **do**
10  Perform routing for net $i$ with $C_h$ and *ObsMap*;
11  **if** *routing successful* **then**
12  Insert the routed path into $P$;
13  Set the routing grids along routed path as obstacles in *ObsMap*;
14  **else**
15  Set *done* $\leftarrow$ *false*;
16  **if** $|P_o| < |P|$ **then**
17  Set $P_o \leftarrow P$;
18  Set counter $r \leftarrow r+1$;
19  **if** $r \geq \tau$ **then**
20  **break**;
21  **if** *done* $\neq$ *true* **then**
22  Update $C_h$ for all the routing grids along the paths in $P$;
23  Reset the obstacle flags in *ObsMap* as *false*;
24 Initialize congestion cost $C_c$ for routing grids along routed paths in $P_o$;
25 **for** $i = 1$ *to* $|N|$ **do**
26  **if** *net i is not routed* **then**
27  Perform routing for net $i$ with $C_c$;
28  **if** *routing successful* **then**
29  Insert the routed path into $P_o$;
30  Set congestion cost $C_c$ for routing grids along the routed path;
31  **else**
32  Report failed net;

**Algorithm 3:** Negotiation-based routing method.

cost for all the routing grids along the routed paths will be updated to a larger value using Equation (2). Furthermore, all the flags in *ObsMap* are reset to *false*. In the next iteration, those routing grids with larger history cost are less likely to be occupied by the computed routing paths unless there are no alternative routing solutions. Therefore, the negotiation-based routing approach greatly improves the routing solution while avoiding path crossings.

If there are failed nets after the previous iterative routing process, we start to route the failed nets with path crossings allowed (see *Steps* 24-32). At this point, the modified A* searching algorithm considers an additional routing congestion cost $C_c$, which is used to minimize the number of path crossings. In the experiments, $C_c$ is set to be 100 routing grids.

The proposed A* searching algorithm incorporates the additional routing costs on the routing grids, i.e., history cost $C_h$ for routability improvement and congestion cost $C_c$ for minimizing number of path crossings. In the modified A* searching algorithm, the routing cost of the current searching grid $g$ is computed as

$$F(g) = G(g) + C(g) + H(g) \tag{3}$$

where $G(g)$ denotes the path length from the source grid to $g$, $H(g)$ denotes the estimated path length from $g$ to the target grid, and $C(g)$ is the additional routing cost ($C_h$ or $C_c$) for choosing grid $g$. The modified A* searching algorithm is able to compute routing paths with minimized total cost.

*2) Flow-Layer Routing Feedback:* After the negotiation-based routing process, a fairly good routing result is obtained

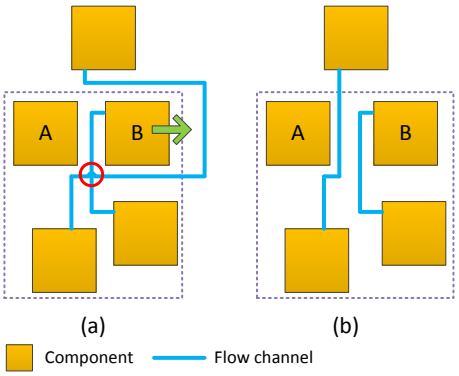



Fig. 9: Flow-layer routing feedback for placement adjustment.

to be used for the current placement solution. However, if the placement solution is not good enough, the routing quality could be further improved. For example, Figure 9(a) illustrates a placement solution with a degraded routing result. In Figure 9(a), the routing resource between components $A$ and $B$ is limited, and thus only one channel can be routed through this region. By placement adjustment, the spacing between $A$ and $B$ can be slightly increased such that the flow channel crossing could be eliminated as shown in Figure 9(b).

We propose the placement adjustment process (see Section IV-B) to remove the above mentioned over-congested regions. After routing, we sweep the whole routing area horizontally and vertically (see Algorithm 2) to find the over-congested regions. Given the over-congested regions, the placement adjustment algorithm refines the placement solution by increasing the left/bottom spacing of each component. For example, for the given region denoted by the congestion window in Figure 9, the placement adjustment algorithm will increase component $B$'s left-side spacing, and then re-compute the placement result by the same SP code. By using the same SP code, the relative positions between the components are preserved, i.e., the optimized placement solution from simulated annealing process is not lost. After iterations of these placement adjustments and flow-layer routings, the design result is significantly improved.

## V. Control-Layer Design

During the control-layer design stage, the valves need to be routed to the control pins. To this end, we adopt a minimum-cost flow formulation for escape routing and obtain the solutions using a linear programming solver [45]. A network flow formulation is constructed considering the minimum width and minimum spacing design rules, where an ingoing/outgoing flow of a routing grid refers to the flow going into/out of the node corresponding to the routing grid. This routing method can compute the control channels from the valves to the control pins in a concurrent way and route the valves with maximized routability. The placement adjustment module will be invoked when there are failed valves, and control-layer routing will be performed on the updated valve positions once again. The co-design flow will terminate when there are no failed valves or after a pre-specified number of iterations.

Figure 10 illustrates the routing feedback during the control-layer design stage. The position of a failed valve during routing along with the surrounding valves is reported. To make the

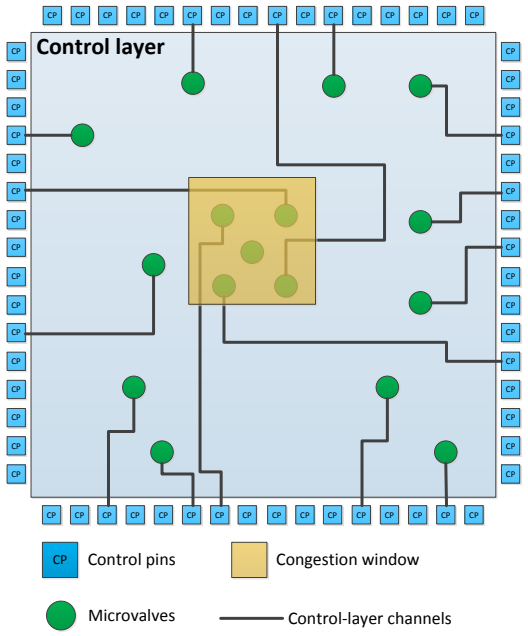



Fig. 10: Control-layer routing feedback.

routing feedback mechanism more effective, especially for the case with large number of failed valves, we identify clusters of failed valves according to the distance between them. Then the congestion areas are detected and reported. The placement adjustment algorithm only adjusts the components that overlap with the congestion areas. In the control-layer design stage, the number of control pins and the total chip area are the major objectives. However, it is hard to simultaneously optimize these two objectives. Thus, we propose two placement adjustment strategies to optimize these two objectives sequentially: (1) A bounding-box-based placement adjustment strategy is proposed to adjust the placement of components for enhanced control-layer routability while controlling the increase in the total chip area; (2) A sequence-pair-based placement adjustment strategy is proposed to reduce the number of control pins needed in the control-layer design stage.

Figure 11 illustrates the general idea of the first method. First, the bounding box of failed valves are constructed. Then, the whole chip is partitioned into eight regions according to the bounding box. Finally, the components within different regions are shifted in different directions by a user-defined distance. By this, the failed valves in the bounding box will be granted more routing resources in the next iteration of the control-layer rerouting. After several iterations, all of the failed valves will successfully be routed to the control pins. As Figure 11 shows, the adjustments have only a small influence to the flow-layer design, i.e., the chip size is slightly increased. Another advantage of this adjustment strategy is that the flow-channels can be reused as much as possible. Thus, the runtime consumption of flow-channel rerouting is notably reduced, and hence the whole design flow becomes more efficient.

Figure 12 illustrates the general idea of the flow-channel rerouting after the bounding-box-based placement adjustment. From the figure, the majority of channel segments can be reused, and we only need to connect pseudo crosspoints along the boundary of the regions. In each placement adjustment iteration, the gaps between the partitioned regions are increased by

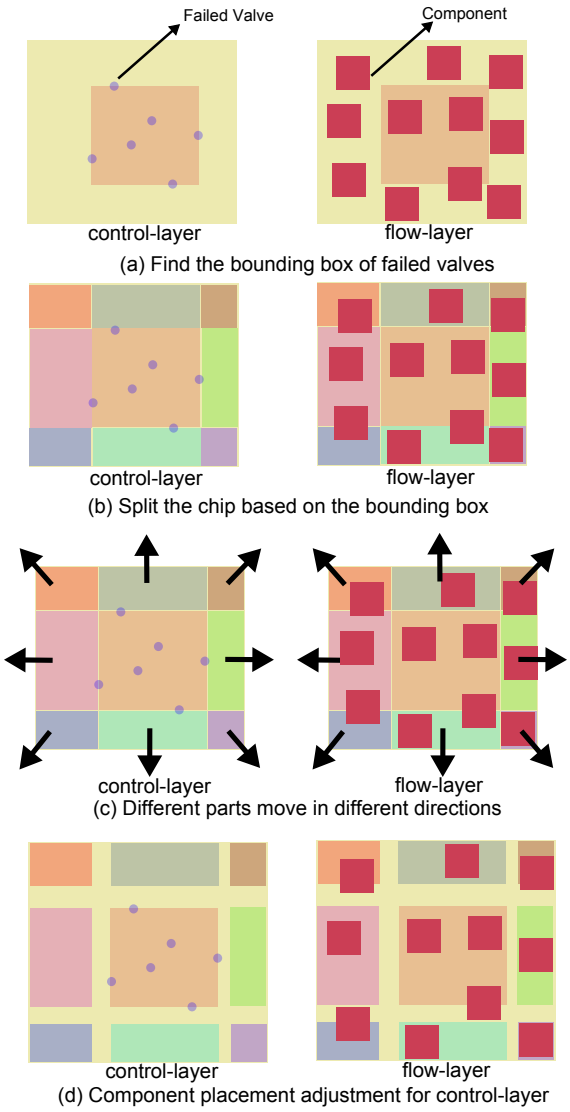
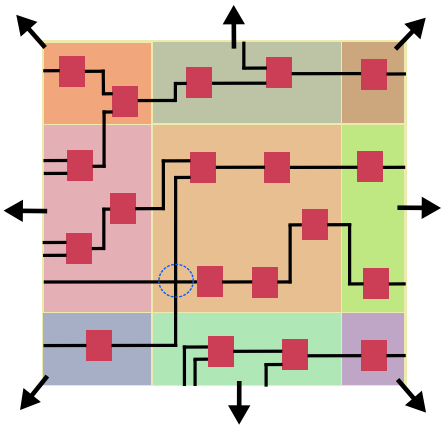



Fig. 11: Placement adjustment for control-layer design.

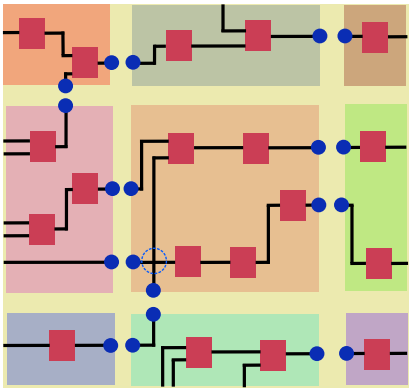
$\frac{n_f}{\alpha}+\beta$, where $n_f$ represents the total number of failed valves. Here, $\alpha$ and $\beta$ are user-defined parameters. In the experiments, $\alpha = 8$ and $\beta = 1$ are employed.

In the sequence-pair-based placement adjustment strategy, the expanded spacings $ex_i$ and $ey_i$ of component $m_i$ in Algorithm 1 are used to adjust the placement solution. For each component, we calculate the number of failed valves $n_x$ on its left side, and the number of failed valves $n_y$ on its bottom side. Then, the expanded spacing of $ex_i$ ($ey_i$) is $\alpha \times n_x + \beta$ ($\alpha \times n_y + \beta$), where $\alpha$ and $\beta$ are user-defined parameters. In the experiments, $\alpha = 0.1$ and $\beta = 1$ are employed. Then, $ex_i$ and $ey_i$ are used to update the placement results using the same SP code. In this strategy, the relative positions of the components keep unchanged. In most cases, the sequence-pair-based placement adjustment strategy obtains routing solutions with a smaller number of flow-channel crossings and control pins.

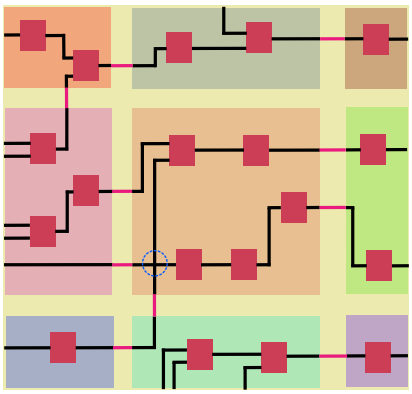
For the available benchmarks including biochips for real-life assays as well as artificially constructed benchmarks, the proposed co-design flow successfully finishes the control-layer routing with a 100% routing completion rate (see Section VI for details). We claim that, using the direct-addressing scheme,



(a) Split the chip and move the modules



(b) Flow-channels move with modules



(c) Connect the points on the boundries of different modules

Fig. 12: Flow-channel movement for placement adjustment.

most of the biochip benchmarks can be successfully routed by our proposed co-design flow. If there exists a biochip that cannot be successfully routed by our flow, the biochip architecture needs to be re-synthesized for a successful design.

## VI. Experimental Results

We have implemented the proposed physical co-design method for flow-based microfluidic biochips in C++ on a 2.60GHz 32-core Intel Xeon Linux workstation with 132GB memory. Only a single thread is used. The benchmarks are the same as used in [1] except four additionally synthesized benchmarks. Table I shows the details of the benchmarks along with the experimental results obtained by the proposed flow-layer design, where "#C" denotes the total number of components, "Width" and "Height" represent the sizes of the resulting chip in terms of routing grids obtained by the component placement method, "#Cross" gives the number of flow-channel crossings after routing, "Length" gives the flow channel length, and "CPU" gives total runtime in seconds. The columns under "SA" give the routing results obtained directly after the proposed simulated annealing-based placement algorithm, i.e., without the placement adjustment stage. "PA" gives the routing results obtained after the proposed placement adjustment algorithm. "Imp." gives the improvement ratio. From the results, it can be concluded that the placement adjustment method further reduces the number of flow-channel crossings especially for large testcases with many components. As each crossing will cause four additional valves and an increased number of control pins, these improvements are significant and desirable. Due to the placement adjustment, the chip area and the flow-channel length are also increased. However, this is acceptable considering the reduction in the number of valves. The results show that the proposed placement adjustment method obtains 35.68% reduction in the number of crossings on average.

To validate the effectiveness of the proposed component placement method, we compared the results to the existing work from [36]. As mentioned in Section I, the placement method in [36] embeds the components into two-dimensional rectangular grids, which greatly reduces the solution space and degrades the placement quality. Table II shows the corresponding results. To allow for a fair comparison, the same routing algorithm is used in both methods. From the results, the proposed method obtains an average of 40.04% in the number of crossings, 31.95% in chip area, and 22.02% in flow-channel length.

Figures 13 to 19 illustrate the solution quality using three different placement adjustment strategies of the proposed co-design flow. "BB-1" shows the results obtained by the proposed bounding-box-based strategy, and "SP" shows the results obtained by the sequence-pair-based strategy. Besides that, we additionally implemented a modified bounding-box-based strategy (denoted by"BB") to evaluate the trade-off between the number of valves and the total chip area. In "BB-2", flow channels belonging to the same partitioned region will shift with components, while flow channels cut by region boundaries are rerouted using Algorithm 3. This way, the number of flow-channel crossings will be reduced by flow-channel rerouting. To control the chip area, we decrease the initial spacing between components in the component placement stage. To verify the performance of the three strategies, we synthesize four additional benchmarks ("Random-1" to "Random-4"), where the number of components of the synthesized benchmarks are 20, 70, 76, and 70, respectively. Note that placement adjustment for the control-layer design are not needed for benchmarks "PCR", "ProteinSplit-1", "InVitro-1", "InVitro-2", "Random-1", "kinase act-1", and "acid proc-1", i.e., the experimental results for these benchmarks are identical in the figures.

As shown in Figure 13, "BB-1" and "BB-2" have better performance than "SP" with respect to chip area. However, "SP" can get solutions with a smaller number of flow-channel crossings and control pins than "BB-1" and "BB-2" (see

TABLE I: Experimental results of flow-layer design with vs. without placement adjustment.

| Bench | #C | Width | | Height | | Area | | #Cross | | | Length | | CPU (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SA | PA | SA | PA | SA | PA | SA | PA | Imp. | SA | PA | SA | PA |
| PCR | 16 | 50 | 51 | 58 | 58 | 2900 | 2958 | 2 | 1 | 50.00% | 569 | 522 | 25.088 | 42.684 |
| ProteinSplit-1 | 30 | 63 | 63 | 78 | 78 | 4914 | 4914 | 5 | 5 | 0.00% | 1162 | 1162 | 85.174 | 113.973 |
| ProteinSplit-2 | 66 | 125 | 131 | 128 | 130 | 16000 | 17030 | 74 | 42 | 43.24% | 3139 | 3247 | 379.001 | 527.898 |
| InVitro-1 | 30 | 69 | 70 | 73 | 73 | 5037 | 5110 | 3 | 1 | 66.67% | 756 | 802 | 59.827 | 84.143 |
| InVitro-2 | 45 | 82 | 84 | 98 | 98 | 8036 | 8232 | 10 | 8 | 20.00% | 1508 | 1485 | 128.832 | 179.69 |
| InVitro-3 | 60 | 97 | 99 | 113 | 113 | 10961 | 11187 | 14 | 6 | 57.14% | 2002 | 1846 | 230.414 | 301.372 |
| kinase act-1. [46] | 11 | 121 | 126 | 89 | 90 | 11214 | 11340 | 20 | 12 | 40.00% | 1021 | 884 | 32.24 | 41.11 |
| kinase act-2. [46] | 33 | 232 | 239 | 167 | 169 | 38744 | 40391 | 56 | 45 | 19.64% | 3555 | 2745 | 248.03 | 389.29 |
| acid proc-1. [47] | 17 | 148 | 152 | 82 | 86 | 12136 | 13072 | 42 | 30 | 28.57% | 2577 | 1900 | 89.82 | 112.07 |
| acid proc-2. [47] | 34 | 164 | 170 | 166 | 174 | 23904 | 29580 | 76 | 52 | 35.68% | 6123 | 4685 | 331.37 | 443.11 |

TABLE II: Experimental results in flow-layer compared with [36].

| Bench | #C | Width | | Height | | Area | | | #Cross | | | Length | | | CPU (s) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | [36] | PA | [36] | PA | [36] | PA | Imp. | [36] | PA | Imp. | [36] | PA | Imp. | [36] | PA |
| PCR | 16 | 57 | 51 | 50 | 58 | 2850 | 2958 | -3.79% | 4 | 1 | 75.00% | 484 | 522 | -7.85% | 13.83 | 42.68 |
| ProteinSplit-1 | 30 | 81 | 63 | 100 | 78 | 8100 | 4914 | 39.33% | 22 | 5 | 77.27% | 1159 | 1162 | -0.26% | 48.06 | 113.97 |
| ProteinSplit-2 | 66 | 237 | 131 | 231 | 130 | 54747 | 17030 | 68.89% | 75 | 42 | 44.00% | 6502 | 3247 | 50.06% | 252.92 | 527.9 |
| InVitro-1 | 30 | 83 | 70 | 81 | 73 | 6723 | 5110 | 23.99% | 4 | 1 | 75.00% | 924 | 802 | 13.20% | 28.9 | 84.14 |
| InVitro-2 | 45 | 107 | 84 | 174 | 98 | 18618 | 8232 | 55.78% | 10 | 8 | 20.00% | 2067 | 1485 | 28.16% | 65.63 | 179.69 |
| InVitro-3 | 60 | 209 | 99 | 171 | 113 | 35739 | 11187 | 68.70% | 9 | 6 | 33.33% | 4317 | 1846 | 57.24% | 127.96 | 301.34 |
| kinase act-1. [46] | 11 | 153 | 126 | 116 | 90 | 17748 | 11340 | 36.11% | 16 | 12 | 25.00% | 1266 | 884 | 30.17% | 30.54 | 41.11 |
| kinase act-2. [46] | 33 | 253 | 239 | 187 | 169 | 47311 | 40391 | 14.63% | 55 | 45 | 18.18% | 3245 | 2745 | 15.4% | 265.55 | 389.29 |
| acid proc-1. [47] | 17 | 161 | 152 | 90 | 86 | 14490 | 13072 | 9.79% | 37 | 30 | 23.33% | 2497 | 1900 | 23.91% | 92.31 | 112.07 |
| acid proc-2. [47] | 34 | 175 | 170 | 180 | 174 | 31500 | 29580 | 6.10% | 60 | 52 | 13.33% | 5213 | 4685 | 10.13% | 393.31 | 443.11 |
| Avg. | | | | | | | | 31.95% | | | 40.44% | | | 22.02% | | |

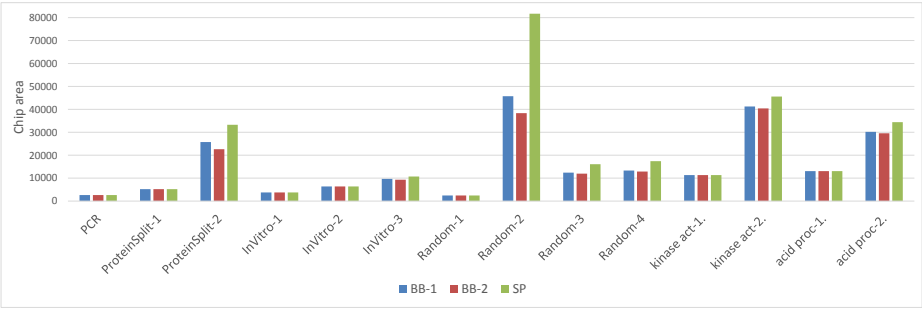



Fig. 13: Comparison in chip area using the three placement adjustment strategies.

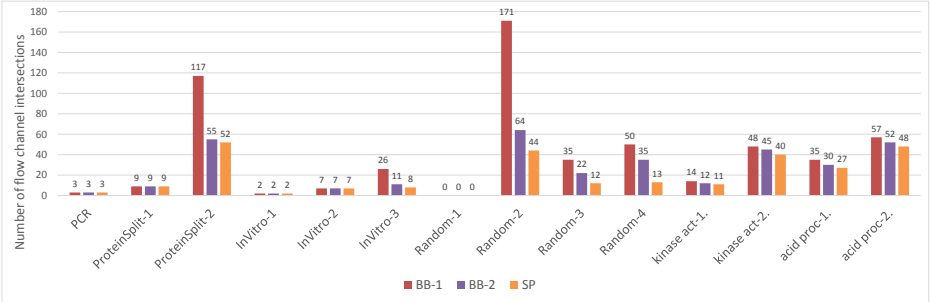



Fig. 14: Comparison in number of flow-channel crossings using the three placement adjustment strategies.

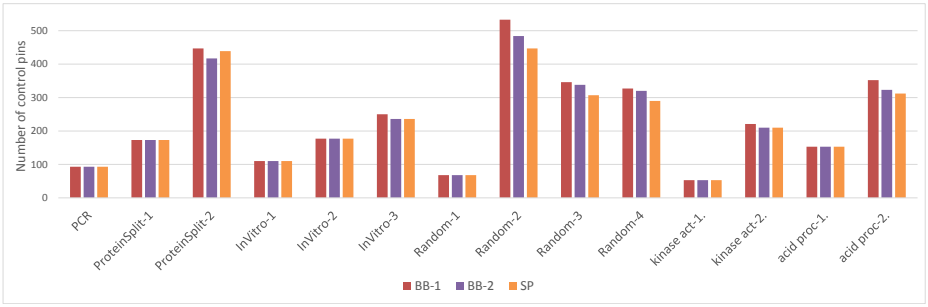



Fig. 15: Comparison in number of control pins using the three placement adjustment strategies.

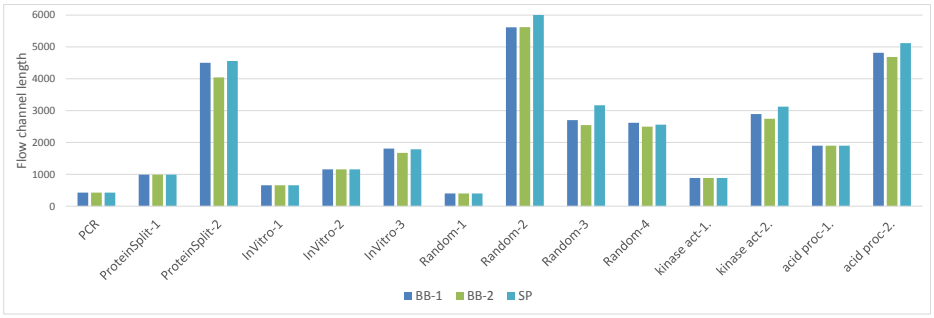



Fig. 16: Comparison in flow-channel length using the three placement adjustment strategies.

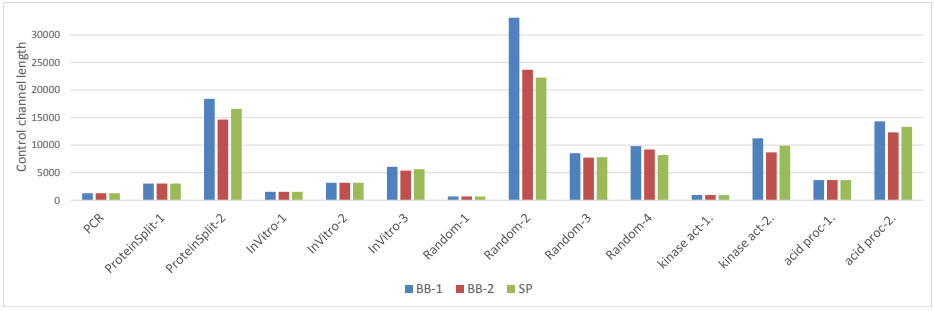



Fig. 17: Comparison in control-channel length using the three placement adjustment strategies.

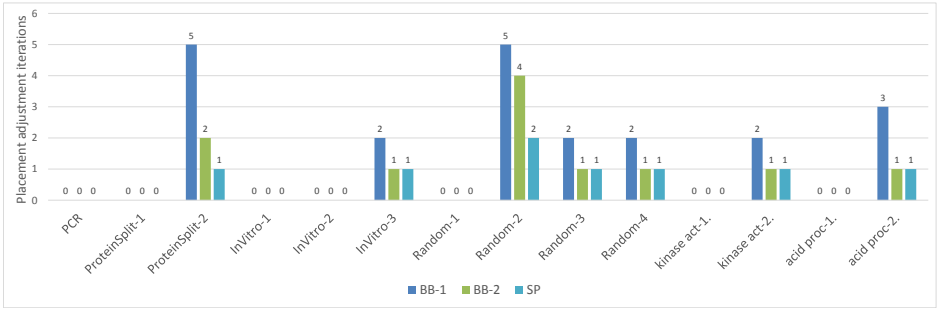



Fig. 18: Comparison in the number of adjustment iterations using the three placement adjustment strategies.

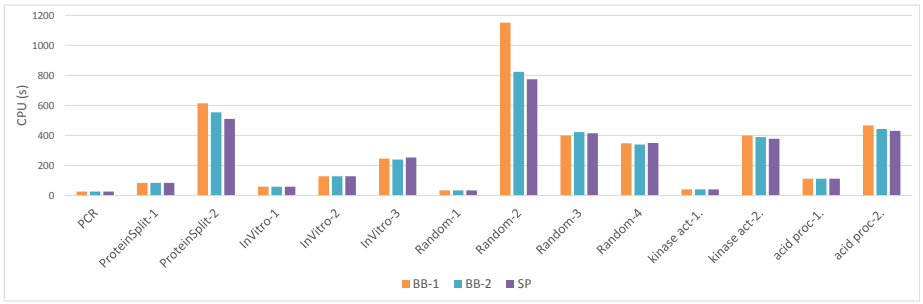



Fig. 19: Comparison in runtime using the three placement adjustment strategies.

Figure 14 and Figure 15). Figure 16 shows that the flow channel lengths of the results obtained by these three strategies are very close. Figure 17 gives the results of the control-channel length, where "BB-1" obtains solutions with much longer control-channel length because the solutions have more flow-channel crossings, and each flow-channel crossing causes four additional valves that need to be connected to the control pins. As shown in Figure 18 and Figure 19, "BB-1" needs more adjustment iterations to finish the whole flow, i.e., it consumes more runtime. According to the experimental results, it seems that "BB-1" has little advantage over "BB-2" and "SP". However, it will make the design flow more efficient when the flow-channel routing algorithm becomes too complex. Overall, the proposed bounding-box-based strategy obtains solutions with a smaller chip area, while the proposed sequence-pair-based strategy obtains solutions with smaller number of flow-channel crossings and control pins. All of the proposed placement adjustment strategies can finish the design flow successfully without any violations.

## VII. Conclusion

We proposed an effective integrated physical co-design methodology, which seamlessly integrates the flow-layer and control-layer design stages. In the flow-layer design stage, we proposed an integrated placement and routing flow, which features sequence-pair-based iterative placement adjustment and flow-layer routing feedbacks. In the control-layer design stage, we proposed different placement adjustment strategies with control-layer routing feedback. Compared with the existing work, our flow-layer design method obtains significantly improved design solutions for flow-based biochips, with an improved chip area, number of flow-channel crossings, and flow-channel length. Moreover, our control-layer design method can finish the overall flow successfully without failed valves. Benchmarks from real-life biochemical applications validate the effectiveness of the proposed co-design method. Future work will be to improve the efficiency of this codesign flow and apply it to solve similar problems such as codesign for paper-based biochips [48].

## VIII. Acknowledgement

The work of H. Yao was supported in part by the National Natural Science Foundation of China (61674093) and Tsinghua University Initiative Scientific Research Program (20141081203). The work of T. Ho was supported in part by the Ministry of Science and Technology of Taiwan, under Grant MOST 105-2221-E-007-118-MY3 and 104-2220-E-007-021 and in part by the Technical University of Munich-Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Program under grant agreement no 291763. The work of Y. Cai was supported by the National Natural Science Foundation of China (61274031).

## References

[1] Q. Wang, Y. Ru, H. Yao, T.-Y. Ho, and Y. Cai, "Sequence-pair-based Placement and Routing for Flow-based Microfluidic Biochips," *Proc. of ASP-DAC*, 2016, pp. 587-592.

[2] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic Large-Scale Integration," *Science*, vol. 298, no. 5593, pp. 580-584, 2002.

[3] G. M. Whitesides, "The Origins and the Future of Microfluidics," *Nature*, vol. 442, no. 7101, pp. 368-373, 2006.

[4] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle, "Microfluidic Lab-on-a-Chip Platforms: Requirements, Characteristics and Applications," *Chemical Society Reviews*, vol. 39, no. 3, pp. 1153-1182, 2010.

[5] N. N. Watkins, U. Hassan, G. Damhorst, H. Ni, A. Vaid, W. Rodriguez, and R. Bashir, "Microfluidic Cd4+ And Cd8+ T Lymphocyte Counters for Point-of-care HIV Diagnostics Using Whole Blood," *Science Translational Medicine*, vol. 5, no. 214, p. 214ra170, 2013.

[6] Rice University. "Quick Nano-bio-chip Checks for Oral Cancer," *ScienceDaily*, www.sciencedaily.com/releases/2010/04/100405152753.htm.

[7] J. A. Rogers and R. G. Nuzzo, "Recent Progress in Soft Lithography," *Materials Today*, vol. 8, no. 2, pp. 50-56, 2005.

[8] T. M. Squires and S. R. Quake, "Microfluidics: Fluid Physics at the Nanoliter Scale," *Reviews of Modern Physics*, vol. 77, pp. 977-1026, 2005.

[9] V. Studer, G. Hang, A. Pandolfi, M. Ortiz, W. F. Anderson, and S. R. Quake, "Scaling Properties of a Low-Actuation Pressure Microfluidic Valve," *Journal of Applied Physics*, vol. 95, no. 1, pp. 393-398, 2004.

[10] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-Layer Optimization for Flow-Based mVLSI Microfluidic Biochips," *Proc. of CASES*, 2014, pp. 16:1-16:10.

[11] R. Wille, B. Li, U.Schlichtmann, R. Drechsler, "From biochips to quantum circuits: computer-aided design for emerging technologies," *Proc. of ICCAD*, 2016, pp. 1-6.

[12] M. Cho and D. Z. Pan, "A High-Performance Droplet Routing Algorithm for Digital Microfluidic Biochips," *IEEE Transactions on CAD*, vol. 27, no. 10, pp. 1714-1724, 2008.

[13] O. Keszocze, R. Wille, T.-Y. Ho, and R. Drechsler, "Exact One-pass Synthesis of Digital Microfluidic Biochips," *Proc. of DAC*, 2014, pp. 1-6.

[14] 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" *Proc. of DAC*, 2014, pp. 76:1-76:6.

[15] Q. Wang, W. He, H. Yao, T.-Y. Ho, Y. Cai, "SVM-Based Routability-Driven Chip-Level Design for Voltage-Aware Pin-Constrained EWOD Chips", *Proc. of ISPD*, 2015, pp. 49-56.

[16] H. Yao, Q. Wang, Y. Shen, T.-Y. Ho, and Y. Cai, "Integrated Functional and Washing Routing Optimization for Cross-Contamination Removal in Digital Microfluidic Biochips," *IEEE Transactions on CAD*, vol. 35, no. 8, pp. 1283-1296, 2016.

[17] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips" *Proc. of ASP-DAC*, 2017, pp. 530-535.

[18] Q. Wang, S. Zuo, H. Yao, T.-Y. Ho, B. Li, U. Schlichtmann, Y. Cai, "Hamming-distance-based valve-switching optimization for control-layer multiplexing in flow-based microfluidic biochips," *Proc. of ASP-DAC*, 2017, pp. 524-529.

[19] T. Tseng, B. Li, U. Schlichtmann, and T. Ho, "Storage and caching: Synthesis of flow-based microfluidic biochips," *IEEE Design & Test*, vol. 32, no. 6, pp. 69-75, 2015.

[20] C. Liu, B. Li, H. Yao, P. Pop, T.-Y. Ho, U. Schlichtmann, "Transport or Store?: Synthesizing Flow-based Microfluidic Biochips using Distributed Channel Storage," *Proc. of DAC*, 2017, pp. 49:1-49:6.

[21] T. Tseng, B. Li, T. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," *Proc. of DAC*, 2015, pp. 141:1-141:6.

[22] T. Tseng, B. Li, M. Li, T.-Y. Ho, U. Schlichtmann, "Reliability-Aware Synthesis With Dynamic Device Mapping and Fluid Routing for Flow-Based Microfluidic Biochips," *IEEE Transactions on CAD*, vol. 35, no. 12, pp. 1981-1994, 2016.

[23] C. Liu, B. Li, B. B. Bhattacharya, K. Chakrabarty, T. Ho, and U. Schlichtmann, "Testing microfluidic fully programmable valve arrays (FPVAs)," *Proc. of DATE*, 2017, pp. 91-96.

[24] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-Layer Routing and Control-Pin Minimization for Flow-Based Microfluidic Biochips," *IEEE Transactions on CAD* vol. 36, no. 1, pp. 55-68, 2016.

[25] I. E. Araci, P. Pop, and K. Chakrabarty, "Microfluidic very large-scale integration for biochips: Technology, testing and fault-tolerant design," *Proc. of IEEE European Test Symposium* 2015, pp. 1-8.

[26] T. Tseng, M. Li, B. Li, T.-Y. Ho, U. Schlichtmann, "Columba: Co-Layout Synthesis for Continuous-Flow Microfluidic Biochips," Proc. of DAC 2016, pp. 147:1-147:6.

[27] Paul Pop, Ismail Emre Araci, and Krishnendu Chakrabarty, "Continuous-Flow Biochips: Technology, Physical-Design Methods, and Testing," *IEEE Design & Test* vol. 32, no. 6, pp. 8-19, 2015.

[28] I. E. Araci,andStephen R. Quake, "Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves," *Lab Chip* vol. 12, no. 16, pp. 2803-2806, 2012.

[29] N. Amin, W. Thies, and S. Amarasinghe, "Computer-Aided Design for Microfluidic Chips Based on Multilayer Soft Lithography," *Proc. of ICCD*, 2009, pp. 2-9.

[30] W. H. Minhass, P. Pop, J. Madsen, and T.-Y. Ho, "Control Synthesis for the Flow-based Microfluidic Large-Scale Integration Biochips," *Proc. of ASP-DAC*, 2013, pp. 205-212.

[31] W. H. Minhass, P. Pop, and J. Madsen, "System-Level Modeling and Synthesis of Flow-Based Microfluidic Biochips," *Proc. of CASES*, 2011, pp. 225-233.

[32] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga, "Architectural Synthesis of Flow-Based Microfluidic Large-Scale Integration Biochips," *Proc. of CASES*, 2012, pp. 181-190.

[33] W. H. Minhass, P. Pop, and J. Madsen, "Synthesis of Biochemical Applications on Flow-Based Microfluidic Biochips Using Constraint Programming," *Proc. of 2012 DTIP*, 2012 , pp. 37-41.

[34] C.-X. Lin, C.-H. Liu, I.-C. Chen, D. T. Lee, and T.-Y. Ho, "An Efficient Bi-criteria Flow Channel Routing Algorithm for Flow-Based Microfluidic Biochips," *Proc. of DAC*, 2014, pp. 1-6.

[35] K.-H. Tseng, S.-C. You, J.-Y. Liou, and T.-Y. Ho, "A Top-Down Synthesis Methodology for Flow-Based Microfluidic Biochips Considering Valve-Switching Minimization," *Proc. of ISPD*, 2013, pp. 123-129.

[36] J. McDaniel, B. Parker, and P. Brisk, "Simulated Annealing-Based Placement for Microfluidic Large Scale Integration (mLSI) Chips," *Proc. of VLSI-SoC*, 2014, pp. 1-6.

[37] H. Yao, Q. Wang, Y. Ru, and T.-Y. Ho, "Integrated Flow-Control Co-Design Methodology for Flow-Based Microfluidic Biochips" *IEEE Design & Test*, vol. 32, no. 6, pp. 60-68, 2015

[38] D. Grissom, K. O'Neal, B. Preciado, H. Patel, R. Doherty, N. Liao, and P. Brisk, "A Digital Microfluidic Biochip Synthesis Framework," *Proc. of VLSI-SoC*, 2012, pp. 177-182.

[39] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair," *IEEE Transactions on CAD*, vol. 15, no. 12, pp. 1518-1524, 1996.

[40] Xiaoping Tang, Ruiqi Tian, and D. F. Wong, "Fast Evaluation of Sequence Pair in Block Placement by Longest Common Subsequence Computation," *IEEE Transactions on CAD*, vol. 20, no. 12, pp. 1406-1413, 2001.

[41] Andrzej Kozik, "Fully Dynamic Evaluation of Sequence Pair," *IEEE Transactions on CAD*, vol. 32, no. 6, pp. 894-904, 2013.

[42] Qiang Ma, Linfu Xiao, Yiu-Cheong Tam, Evangeline F. Y. Young, "Simultaneous Handling of Symmetry, Common Centroid, and General Placement Constraints," *IEEE Transactions on CAD*, vol. 30, no. 1, pp. 85-95, 2011.

[43] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," *Proc. of FPGA*, 1995, pp. 111-117.

[44] H. Yao, T.-Y. Ho, and Y. Cai, "PACOR: Practical Control-layer Routing Flow with Length-matching Constraint for Flow-based Microfluidic Biochips," *Proc. of DAC*, 2015, no. 142.

[45] Gurobi Optimizer. http://www.gurobi.com/.

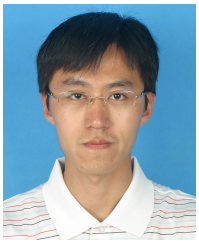
[46] C. Fang, Y. Wang, N. T. Vu, W.-Y. Lin, Y.-T. Hsieh, L. Rubbi, M. E. Phelps, M. Muschen, Y.-M. Kim, A. F. Chatziioannou, H.-R. Tseng, T. G. Graeber, "Integrated microfluidic and imaging platform for a kinase activity radioassay to analyze minute patient cancer samples," *Cancer research*, vol. 70, no. 21, pp. 8299-8308, 2010.

[47] J. W. Hong, V. Studer, G. Hang, W. F. Anderson, and Stephen R. Quake, "A nanoliter-scale nucleic acid processor with parallel architecture," *Nature Biotechnology*, vol. 22, no. 4, pp. 435-439, 2004.

[48] Q. Wang, Z. Li, H. Cheong, O.-S. Kwon, H. Yao, T.-Y. Ho, K. Shin, B. Li, U. Schlichtmann, Y. Cai, "Control-fluidic CoDesign for paper-based digital microfluidic biochips," *Proc. of ICCAD*, 2016, no. 103.

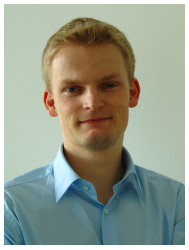

**Qin Wang** received his B.S. degree in Software Engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently a Ph.D. student in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include physical design and design automation for microfluidic biochips.

**Hao Zou** is currently an undergraduate student in the Department of Computer Science and Technology, Tsinghua University, Beijing, China.



**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 postdoctoral research scholar in the Department of Computer Science and Engineering, University of California at San Diego, La Jolla. He joined the Department of Computer Science and Technology in Tsinghua University as an assistant professor in September 2009. His research interests include computer-aided design for microfluidic biochips and very large scale integration (VLSI) physical design. Dr. Yao received two Best Paper Award Nominations at ICCAD in 2006 and 2008, respectively. He received the ISQED Best Paper Award Nomination in 2011, and received the SASIMI Best Paper Award in 2016.

**Tsung-Yi Ho (SM'12)** received his Ph.D. in Electrical Engineering from National Taiwan University in 2005. He is a Professor with the Department of Computer Science of National Tsing Hua University, Hsinchu, Taiwan. His research interests include design automation and test for microfluidic biochips and nanometer integrated circuits. He has presented 10 tutorials and contributed 11 special sessions in ACM/IEEE conferences, all in design automation for microfluidic biochips. He has been the recipient of the Invitational Fellowship of the Japan Society for the Promotion of Science (JSPS), the Humboldt Research Fellowship by the Alexander von Humboldt Foundation, and the Hans Fischer Fellow by the Institute of Advanced Study of the Technical University of Munich. He was a recipient of the Best Paper Awards at the VLSI Test Symposium (VTS) in 2013 and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems in 2015. He served as a Distinguished Visitor of the IEEE Computer Society for 2013-2015, the Chair of the IEEE Computer Society Tainan Chapter for 2013-2015, and the Chair of the ACM SIGDA Taiwan Chapter for 2014-2015. Currently he serves as an ACM Distinguished Speaker, a Distinguished Lecturer of the IEEE Circuits and Systems Society, and Associate Editor of the ACM Journal on Emerging Technologies in Computing Systems, ACM Transactions on Design Automation of Electronic Systems, ACM Transactions on Embedded Computing Systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, and IEEE Transactions on Very Large Scale Integration Systems, and the Technical Program Committees of major conferences, including DAC, ICCAD, DATE, ASP-DAC, ISPD, ICCD, etc.



**Robert Wille** received the Diploma and Dr.-Ing. degrees in computer science from the University of Bremen, Germany, in 2006 and 2009, respectively. He is a Full Professor at the Johannes Kepler University Linz. Before, he was with the Group of Computer Architecture at the University of Bremen and with the German Research Center for Artificial Intelligence (DFKI). Additionally, he worked as Lecturer at the University of Applied Science in Bremen, Germany, and as visiting professor at the University of Potsdam, Germany, and the Technical University Dresden, Germany. His research interests are in the design of circuits and systems for both conventional and emerging technologies. Robert Wille was repeatedly awarded e.g. with a Best Paper Award at ICCAD in 2013 and FDL in 2010. Besides that, he served Guest Editor for the ACM's JETC, Springer's LNCS, and the MVLSC. Additionally, he was General Chair and PC Chair for several conferences such as ISMVL, FDL, and others as well as is frequent PC member and track chair for conferences such as ASP-DAC, DAC, DATE, ICCAD, and more.

**Yici Cai** received B.S degree in Electronic Engineering from Tsinghua University, Beijing, China in 1983, received M.S. degree in Computer Science and Technology from Tsinghua University, Beijing, China in 1986, and received Ph.D in Computer Science, University of Science & Technology of China, Hefei, China, in 2007. She has been a professor with the Department of Computer Science & Technology, Tsinghua University. Her research interests include design automation for VLSI integrated circuits algorithms and theory, power/ground distribution network analysis and optimization, high performance clock synthesis, low power physical design.