ParaVOM: Parallel-Execution-Aware Validation and Optimization for Multilayered Continuous-Flow Microfluidic Biochips

Meng Lian, *Student Member, IEEE,* Shucheng Yang, Mengchu Li, *Member, IEEE,*
Tsun-Ming Tseng, *Senior Member, IEEE,* and Ulf Schlichtmann, *Senior Member, IEEE,*

***Abstract*—Multilayered continuous-flow microfluidic biochips are rapidly advancing platforms for delicate bio-applications. The high complexity of biochip structures and application protocols drives the growing demand for design automation solutions. Current research enables the automatic synthesis of the physical layout and the scheduling and binding protocols of biochips, showcasing the significant potential of microfluidic design automation for improved resource utilization and reduced bioassay completion time. However, state-of-the-art synthesis methods primarily focus on device and operation levels, assuming flow paths are always available and neglecting interactions of flow and control channels. This creates a critical gap in the synthesis process, causing performance degradation, resource redundancy, or even infeasible designs. This work bridges this gap with a two-stage approach. Firstly, we perform a mathematical model to synthesize a high-level protocol that specifies the paths and execution orders of fluid transportation operations. Specifically, we construct flow paths based on the fluidic architecture of a given biochip design and optimize scheduling schemes to minimize the completion time of a given bioassay. Next, we perform a simulation-based synthesis of control channel pressurization sequences to realize the high-level protocol. Experimental results confirm that the proposed approach efficiently validates flow paths for feasible designs, identifies conflicting design features in infeasible designs, and improves the design efficiency and quality: compared to the original designs, it reduces the average number of control channels by 49%, and compared to the preliminary work, it reduces the average fluid transportation time by 19% and the average program run time by 38%.**

***Index Terms*—Multilayered Continuous-Flow Microfluidic Biochip, Integer Linear Programming, Validation, Scheduling, Control Sequence Synthesis.**

I. Introduction

Multilayered continuous-flow microfluidic biochips [1] (mCFMB) are a promising lab-on-a-chip platform for high-throughput biological applications. On these microscale platforms, complex bioassays such as DNA purification [2], COVID-19 serology testing [3], adipose tissue-on-chip analysis [4], and highly parallelized cell culture [5] can be executed automatically.

A preliminary version of this paper was published in Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2019.

This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Project Number 515003344.

Meng Lian, Shucheng Yang, Mengchu Li, Tsun-Ming Tseng, and Ulf Schlichtmann are with the Chair of Electronic Design Automation, Technical University of Munich (TUM), Arcisstr. 21, 80333 Munich, Germany (e-mail: m.lian@tum.de; shucheng.yang@tum.de; mengchu.li@tum.de; tsun-ming.tseng@tum.de; ulf.schlichtmann@tum.de).

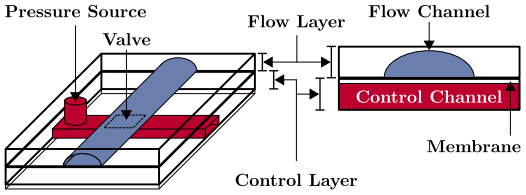


Fig. 1. Illustration of the structure of a push-up valve.

The typical configuration of mCFMBs features a *flow layer* and a *control layer*. Soft lithography technique is applied to bond these patterned elastomer layers, each with dedicated channels that allow gas or fluids to pass through. This multi-layered structure enables the construction of *valves*, which are composite functional units formed by channel segments from different layers with a flexible membrane at the layer interface. Fig. 1 shows a two-layer *polydimethylsiloxane* (PDMS) push-up valve [6] with a flow layer above a control layer. When gas or oil from the pressure source infuses the bottom control channel, the control channel becomes pressurized, pushing the membrane upwards. Since the flow channel segment of the valve has a rounded profile that perfectly fits the expanded membrane, the channel will be sealed, and the fluid movement will be blocked [7].

The integration of channels and valves allows bioengineers to create delicate microfluidic systems on coin-sized chips. However, as application protocols become more complex and the integration scale of mCFMBs grows, manual chip design becomes increasingly time-consuming and error-prone, which results in a strong demand for automated synthesis tools. Microfluidic design automation research works on analyzing and addressing this challenge. The ultimate goal is to develop a fully automated synthesis flow, which can transform a high-level abstraction of a given application into a feasible and optimized chip design with an explicit protocol for executing the application.

Over the past decade, researchers have achieved significant progress in *high-level synthesis* and *physical design*. Li *et al.* [8]–[10] proposed high-level modeling methods to optimize resource utilization according to specified application protocols. Tseng *et al.* [11]–[13] proposed place-and-route tools capable of generating manufacturing-ready physical designs that support applications of varying scales. Minhass *et al.* [14], [15] proposed scheduling and fluid routing approaches that

map operations to given physical topologies. There have also been works that focus on specific optimization criteria, such as reliability [16]–[18], fluid storage [19]–[21], control pin reduction [22]–[25], channel intersection minimization [26], [27], and testing [28]–[30].

Despite these advances, gaps remain between existing methods and a fully automated synthesis flow that seamlessly connects chip designs with application protocols.

Firstly, state-of-the-art high-level synthesis methods primarily focus on operation and device levels, assuming flow paths are always available and neglecting the interaction between the control and flow channels. Although existing approaches can synthesize the device-level scheduling and binding protocols, they fail to generate *channel-level protocols* that specify how to pressurize the control channels to construct the required flow path for each fluid transportation operation. The lack of a channel-level protocol not only makes it difficult to operate the chip but may also result in redundant or even incorrect physical design that fails to support the target application.

Secondly, state-of-the-art high-level synthesis methods often fail to correctly identify and optimally schedule fluid transportation operations that can be executed in parallel. It is typical, e.g., as outlined in [14], to consider different fluid transportation operations as "suffering cross-contamination when being executed in parallel" if their fluids pass through the same on-chip components and as "safe to be executed in parallel" otherwise. However, shared components do not necessarily cause cross-contamination, nor does their absence always guarantee safe parallel execution.

This work bridges the gap between high-level and physical synthesis methods with an approach named ParaVOM, which operates in two stages: In the first stage, ParaVOM adopts an integer linear programming (ILP) model to synthesize the *high-level protocol*, which specifies a conflict-free execution sequence and flow paths of fluid transportation operations, such that the bioassay completion time is minimized. In the second stage, ParaVOM adopts a simulation-based approach to synthesize the corresponding channel-level protocol under adjustable optimization criteria. The key contributions of ParaVOM are summarized as follows:

- It mathematically models the construction of flow paths for reaction products and unintended residues of reactants, as well as the identification of parallel-executable fluid transportation operations.
- It proposes an *execution model* that minimizes bioassay completion time by grouping parallel-executable operations into the same batch for parallel execution.
- It proposes an *event-driven mechanism* to automatically update the protocols to support parallel-executed fluid transportation operations that asynchronously reach their destinations.

The rest of this paper is organized as follows: Section II details the limitations of the state-of-the-art approaches and the motivation of this work. Section III presents an overview of ParaVOM. Sections IV and V describe the two stages of ParaVOM. Experimental results are shown in Section VI, followed by our conclusion in Section VII.

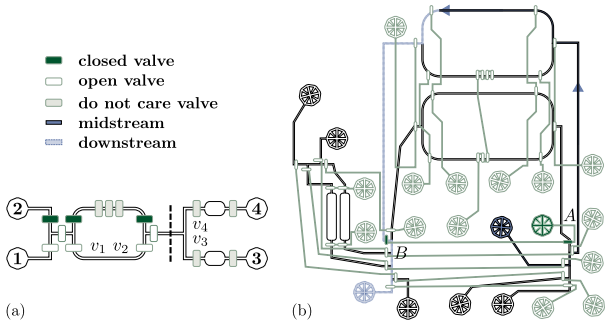


Fig. 2. (a) A manually derived valve actuation protocol from [22] to realize a flow path from inlet 1 along valve v_1 to valve v_2. (b) An mCFMB from [12], where valves A and B share a pressure source.

II. Background

A. Flow Path Validation

The construction of flow paths in mCFMBs is a dynamic process that requires more than just static physical connections of flow channels: as pointed out in [15], a valid flow path for transporting fluids from location a to location b must contain an *upstream* sub-path from inlets to a, a *midstream* sub-path from a to b, and a *downstream* sub-path from b to outlets. As mentioned in Section I, existing approaches omit flow path validation, which is an essential step for two key reasons:

- Current high-level protocols of mCFMBs focus on the midstream sub-path but usually neglect the up- and downstream sub-paths, which results in incorrect prediction of the pressure states of valves that may hinder the feasibility of the flow paths. An example from [22] is shown in Fig. 2(a), which intends to transport fluids from an inlet 1 to the bottom half-ring between valves v_1 and v_2 but mistakenly sets valves right to the dash-line to "do not care" status. However, if valves v_3 and v_4 are pressurized, no downstream sub-path remains from v_2 to an outlet. Due to the initial presence of air in the channel, forming the intended flow path is nearly impossible[1].
- Current physical designs of mCFMBs usually allow multiple valves to be connected by the same control channel to the same pressure source, referred to as "pressure-sharing valves", to reduce the chip-to-world interface. In this context, pressurizing one control channel will pressurize all valves along it, potentially causing unexpected blockages in the flow paths. For example, Fig. 2(b) shows a chip design [12] with two ring-shaped mixers and two reaction chambers. The blue line represents a flow path from an inlet to the upper mixer. To prevent contaminating the bottom mixer during fluid transportation, it is intuitive to pressurize (close) valve A. However, since valves A and B are pressure-sharing valves connected by the same control channel, pressurizing the control channel will close both valves A and B, blocking the downstream sub-path from the target ring to an outlet.

[1]PDMS is gas-permeable [31], so applying sufficient pressure from inlet 1 may gradually expel the air, but this significantly prolongs transportation time and risks fluid mixing with air bubbles.

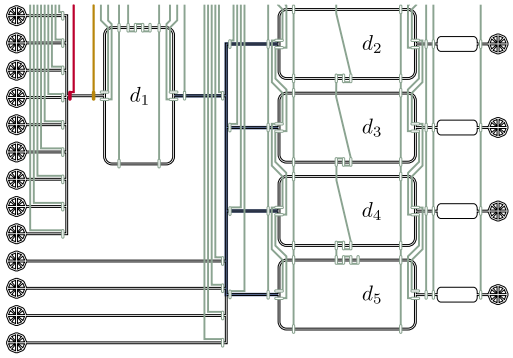


Fig. 3. A partial mCFMB synthesized by Columba S [13] using the netlist description for a ChIP 4-IP application [32]. The design supports fluid-multiplexing from d_1 to d_2, d_3, d_4, and d_5. Upon completing transportation to the nearest mixer d_3, the protocol dynamically adjusts to block flow in d_3 while continuing transport to the remaining mixers. The illustration also showcases control redundancy: the valve (red) at the channel branch overlaps in functionality with the right separation valve (brown).

B. Channel-Level Protocol Update

When parallel-executed fluid transportation operations reach their destinations asynchronously, the completion of one operation may alter the (sub-) flow paths of others, requiring corresponding updates to the channel-level protocols. For example, Fig. 3 shows a partial design of an mCFMB synthesized by a state-of-the-art physical design tool [13] containing five ring-shaped mixers supporting fluid-multiplexing from d_1 to d_2, d_3, d_4, and d_5. When the transportation from d_1 to its nearest mixer, d_3, is completed, the transportation to other mixers is still in progress. Therefore, the control channel pressurization sequence must be updated to block fluid movement in d_3 without disrupting ongoing flows in other sub-paths. Another example is to consider that f_1 and f_2 in Fig. 4 are executed in parallel. After the completion of f_1, i.e., sufficient fluid is collected at outlet 4, the control channel pressurization protocol must be updated to block the path between branch s_1 and outlet 4 such that f_2 can be ensured, i.e., enough fluid can be collected at outlet 3.

C. Design Redundancy

The lack of channel-level protocols results in inaccurate estimation of control resource usage and, consequently, design redundancies. For example, in the partial mCFMB shown in Fig. 3, a valve is placed at every branch of flow channels to control fluid direction. However, the valve (red) at the channel branch overlaps in functionality with the right separation valve (brown) of mixer d_1 and can be removed without affecting chip performance. Removing redundant control components, including valves, channels, and pins, results in a cleaner control-layer structure, reduced chip size, and improved robustness, as control channels are much thinner than flow channels [33] and more prone to damage [34].

D. Identification of Parallel-Executable Operations

Fluid transportation operations are often considered to suffer cross-contamination and thus not parallel-executable if their midstream sub-paths share components and are considered

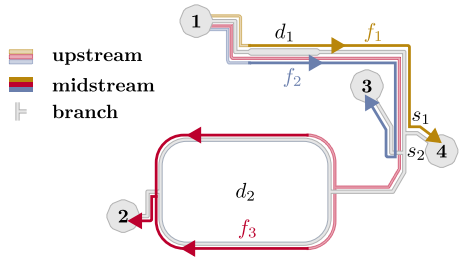


Fig. 4. Three fluid transportation operations f_1, f_2, and f_3 on an mCFMB design synthesized by Columba 2.0 [12]. f_1 and f_2 are operations to collect the reaction products in device d_1 at two different outlets: 3 and 4. f_3 is an operation to collect the reaction products in device d_2 at outlet 2. The upstream sub-paths of all three operations start at inlet 1.

parallel-executable otherwise. However, neither criterion always holds. We illustrate this with an example in Fig. 4, which shows three flow paths: f_1, f_2, and f_3. Although the midstream sub-paths of f_1 and f_2 share the same on-chip components, including device d_1 and branches s_1, there is no risk of cross-contamination as they carry the same fluid. Thus, f_1 and f_2 can be grouped into the same *batch*, where we refer to a batch as a set of operations that can be safely executed in parallel. On the other hand, fluids within f_1 and f_3 are transported along distinct midstream sub-paths: $d_1 \rightarrow s_1 \rightarrow 4$ and $d_2 \rightarrow 2$, respectively. Executing f_1 requires blocking the channel between s_2 and d_2 to prevent the fluid in f_1 from mistakenly flowing to d_2 to mix with the fluid in f_3. This manipulation, however, would disconnect the upstream sub-path of f_3 from the inlet. Thus, f_1 and f_3 must be assigned to different batches for separate execution.

III. Overview of ParaVOM

We illustrate the overall flow of ParaVOM in Fig. 5.

Input. ParaVOM requires the following inputs: a chip design and a bioassay with a device-level binding function.

- The chip design specifies the physical features of a given mCFMB. The flow-layer structure is interpreted as a weighted undirected graph $\mathcal{G}(V, E)$. Here, the vertex set V includes a set P of flow ports, a set S of flow channel branches, and a set N of valves. The edge set E consists of flow channel segments between vertices in V, with weight coefficients describing channel dimensions. We further introduce a set D of devices, where a ring-shaped mixer is defined by the pair of branch vertices at its both ends, and a chamber is defined by the pair of valve vertices at its both ends. The control-layer structure is interpreted as a set C of control channels, each consisting of a set of valves addressed by the control channel.
- The bioassay is modeled using a *sequencing graph* [35] $\mathcal{A}(X, F)$ and a device-level binding function $b : X \rightarrow V$. Here, the vertex set X includes a set P_{in} of inlets for importing reaction products, a set P_{out} of outlets for exporting reaction products and waste, and a set O of bio-operations. The edge set F is the set of fluid transportation operations, where each f_i in F starts from a *source* vertex x_i to a *destination* vertex $\bar{x}_i$. The binding

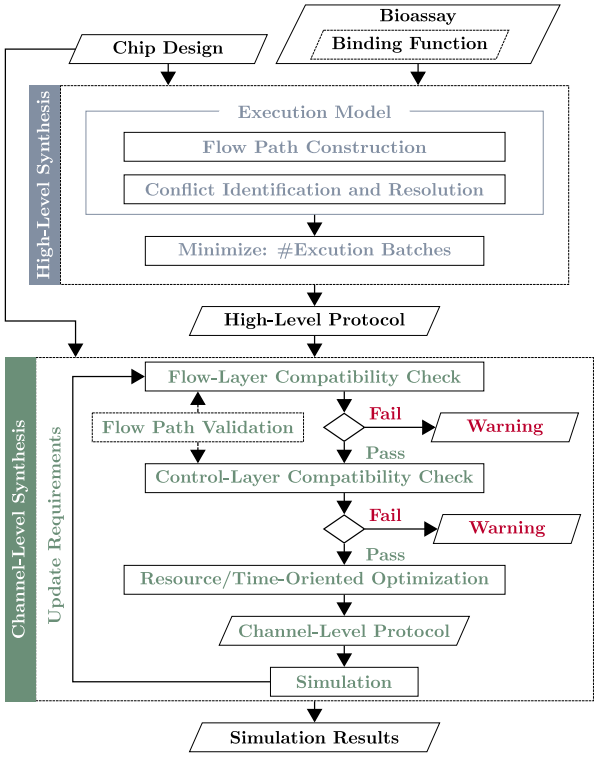


Fig. 5. The overall flow of ParaVOM.

function b maps operations in O to devices in D, and inlets/outlets in $P_{\text{in}}/P_{\text{out}}$ to flow ports in P.

High-Level Synthesis. Using the flow-layer structure, we mathematically model the flow path construction and the identification of parallel-executable operations. ParaVOM assigns conflicting fluid transportation operations to different batches for separate execution and minimizes the bioassay completion time by minimizing the number of batches required for conflict-free execution. The output is a high-level protocol specifying a sequence of execution batches and the target flow paths for fluid transportation operations.

Channel-Level Synthesis. After high-level synthesis, ParaVOM first performs a *flow-layer compatibility check* to verify whether the physical connection of flow channels supports the target flow paths with a *flow path validation method*, which identifies available flow channel segments based on the flow-layer structure and the execution restrictions of a fluid transportation operation. If the check fails, a warning indicates an incompatibility between the flow-layer structure and the high-level protocol; otherwise, ParaVOM proceeds to *control-layer compatibility check*, which models channel-level protocols as execution restrictions and employs the flow path validation method to collect all channel-level protocols capable of constructing the target flow paths. If the construction fails, ParaVOM warns that the control-layer structure is incompatible with the high-level protocol. Otherwise, ParaVOM finalizes a channel-level protocol based on user-defined optimization criteria. Finally, ParaVOM conducts an event-driven simulation to predict the demand for protocol updates triggered by asynchronous completion of parallel-executed fluid transportation operations. If such demand is detected, ParaVOM updates the high-level protocol and synthesizes a new channel-level protocol.

Output. If the chip design is compatible with the desired bioassay execution, ParaVOM produces three outputs: a channel-level protocol, a fluid transportation schedule, and a report detailing the resource usage, including an analysis of design redundancy.

IV. High-Level Synthesis: Execution Model

Given a flow-layer structure and a sequencing graph with a binding function, we avoid conflicts and minimize bioassay completion time by optimally assigning fluid transportation operations to different execution batches.

A. Flow Direction Determination

Given a flow-layer structure $\mathcal{G}(V, E)$, we omit valves and represent each device as a single vertex, resulting in a simplified undirected graph with edges adjusted accordingly. We then transform the simplified graph into a mixed graph $G(V_G, E_G)$, where $E_G = E_u \cup E_d$, with E_u and E_d denoting the sets of undirected and directed edges, respectively. In particular, the directions of edges represent the directions of flow paths and are determined as follows:

1) If an undirected edge e is incident to an inlet v_{in} or an outlet v_{out}, we transform e into a directed edge $e_d \in E_d$ that starts from v_{in}, i.e., $e_d = (v_{\text{in}}, \cdot)$ or ends at v_{out}, i.e., $e_d = (\cdot, v_{\text{out}})$, respectively.
2) If a vertex v is incident to exactly two undirected edges e_1 and e_2, we consider two direction assumptions: 1) e_1 is an incoming edge of v and e_2 is an outgoing edge of v; 2) e_1 is an outgoing edge of v and e_2 is an incoming edge of v. We then calculate the shortest path lengths from every inlet to v and from v to every outlet under these assumptions. If one assumption consistently results in shorter paths than the other for either all inlets or all outlets, we take that assumption to determine the directions of e_1 and e_2. If neither assumption meets this condition, e_1 and e_2 remain undirected.
3) If only one incident edge of a vertex v is undirected, and the other incident edges of v are all incoming (or outgoing) edges of v, then we transform the undirected edge to an outgoing (or incoming) edge of v.

Example. Fig. 6 illustrates how the fluid directions are determined for the flow-layer structure in Fig. 6(a), where vertices 1 and 4 are inlets, and vertices 2 and 3 are outlets. The flow-layer structure is initially represented as an undirected weighted graph in Fig. 6(b), with edge weights in parentheses indicating flow channel lengths.

The first criterion states that fluid always flows out of inlets and into outlets, which can be used to determine the directions of edges $(1, 5)$, $(6, 2)$, $(7, 3)$, and $(4, 8)$, as shown in Fig. 6(c).

The second criterion is based on the principle that fluid naturally follows paths of lower hydraulic resistance[2], often corresponding to shorter channel lengths. For example, under the assumption that the edge between vertices 5 and d_1 is an incoming edge of d_1, the length of the shortest path from inlet

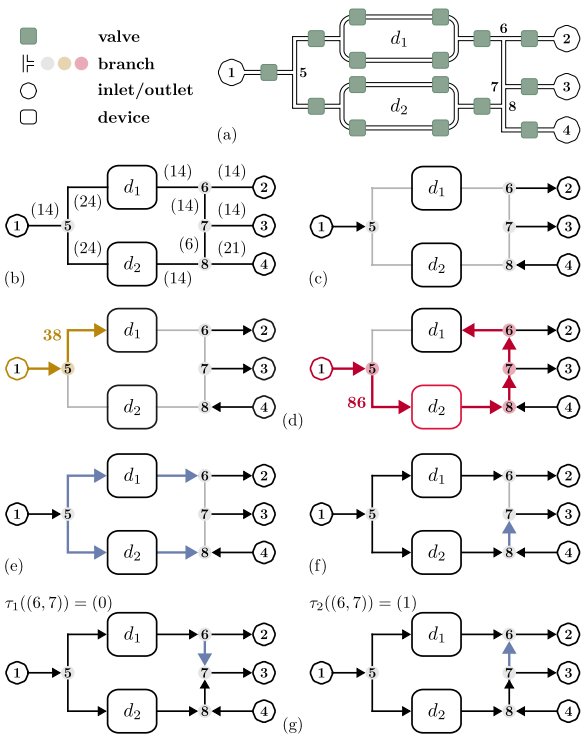


Fig. 6. Illustration of the graph transformation. (a) An exemplary flow-layer structure with (b) its representation as an undirected graph. (c)–(f) Criteria for determining edge directions. (g) The transformation from the mixed graph into two fully directed graphs using τ_1 and τ_2.

1 to d_1 is 38; and under the assumption that the edge between vertices 6 and d_1 is an incoming edge of d_1, the length of the shortest path from inlet 1 to d_1 is 86, as shown in Fig. 6(d). Similarly, we calculate the shortest paths for other inlets and outlets under both assumptions. With $(5, d_1)/(6, d_1)$ as the incoming edge of d_1, the shortest path lengths from inlet 4 to d_1, d_1 to outlet 2, and d_1 to outlet 3 is 83/55, 28/96, and 42/82, respectively. Since the paths from d_1 to outlet 2 (28 vs. 96) and to outlet 3 (42 vs. 82) are shorter under the first assumption, we set $(5, d_1)$ as an incoming edge of d_1. Following the same procedure, we determine the directions of edges $(d_1, 6)$, $(5, d_2)$, and $(d_2, 8)$, as shown in Fig. 6(e).

The third criterion ensures that each vertex has at least one incoming flow and one outgoing flow. Since edges $(d_2, 8)$ and $(4, 8)$ flow into vertex 8, the remaining incident edge $(8, 7)$ must flow outward, as shown in Fig. 6(f).

After applying the three criteria, the only undirected edge is between vertices 6 and 7, i.e., $E_u = \{(6, 7)\}$.

Further, to represent the possible directions of edges in E_u, we assign each edge a boolean value using a function $\tau : E_u \to \{0, 1\}$. Without loss of generality, zero indicates the direction from the endpoint with the smaller index to the larger one and vice versa. For the undirected edge set E_u, $\tau_{E_u} \equiv \tau^{|E_u|}(E_u)$ represents a specific combination of boolean values assigned to the edges in E_u, where $|E_u|$ denotes the cardinality of E_u. The set of all direction assignment functions is denoted as T with $|T| = 2^{|E_u|}$. Given an arbitrary $\tau \in T$, the mixed graph can be transformed into a fully directed graph, with each undirected edge e in E_u assigned a direction based on $\tau(e)$. For example, by assigning a direction to the edge between vertices 6 and 7, the mixed graph in Fig. 6(f) can be transformed into either of the directed graphs in Fig. 6(g).

We introduce the following notations to clarify the graph components. We define E_v as the set of incident edges of a vertex $v \in V_G$. Meanwhile, we define G^τ as the directed graph obtained by applying τ to G. In this directed graph, we define A_v^τ and D_v^τ as the sets of *ancestors* and *descendants* of $v \in V_G$. Specifically, an ancestor of v is any vertex from which fluid can flow to v, and a descendant of v is any vertex to which fluid can flow from v. Further, we define I_v^τ and O_v^τ as the sets of incoming and outgoing edges of v, respectively.

Based on the directions assigned by τ, the flow path for a fluid transportation operation f_i, denoted by p_i, can only include vertices and edges connected to its source or destination vertices, either directly or via a directed path. We introduce U_i^τ, M_i^τ, and W_i^τ as the sets of candidate vertices that can form the upstream, midstream, and downstream sub-paths of p_i in G^τ, respectively, which are defined as: $U_i^\tau \equiv A_{b(x_i)}^\tau$, $M_i^\tau \equiv D_{b(x_i)}^\tau \cap A_{b(\bar{x}_i)}^\tau$, $W_i^\tau \equiv D_{b(\bar{x}_i)}^\tau$. Similarly, we introduce $\bar{U}_i^\tau$, $\bar{M}_i^\tau$, and $\bar{W}_i^\tau$ as the sets of candidate edges corresponding to the upstream, midstream, and downstream sub-paths of p_i in G^τ, respectively, which are defined as: $\bar{U}_i^\tau \equiv \cup_{v \in A_{b(x_i)}^\tau} I_v^\tau$, $\bar{M}_i^\tau \equiv \left(\cup_{v \in D_{b(x_i)}^\tau} O_v^\tau\right) \cap \left(\cup_{v \in A_{b(\bar{x}_i)}^\tau} I_v^\tau\right)$, and $\bar{W}_i^\tau \equiv \cup_{v \in D_{b(\bar{x}_i)}^\tau} O_v^\tau$.

B. Flow Path Construction

For a given fluid transportation operation, the following constraints must be satisfied to construct its flow path based on the flow-layer structure.

Each flow path starts from inlets connected to an external pressure source and ends with outlets that release air to create the necessary pressure drop for fluid movement, which can be formulated as:

$$\underset{f_i \in F}{\forall}: \quad \sum_{b^{-1}(v) \in P_{\text{in}}} q_i^v \geq 1, \quad \sum_{b^{-1}(v) \in P_{\text{out}}} q_i^v \geq 1, \tag{1}$$

where binary variable q_i^v indicates whether vertex $v \in V_G$ is part of p_i. Meanwhile, the source and destination vertices of f_i must be included within p_i, which can be formulated as:

$$\underset{f_i \in F}{\forall}: \quad q_i^{b(x_i)} = 1, \quad q_i^{b(\bar{x}_i)} = 1. \tag{2}$$

To identify whether an edge is part of p_i, we introduce a binary variable q_i^e for each edge $e \in E_G$. Then, the following constraints ensure that e is part of p_i if and only if both its endpoints are part of p_i.

$$\underset{f_i \in F,}{\forall} \underset{e \in E_G}{\forall}: q_i^e \leq q_i^{v_e}, \; q_i^e \leq q_i^{\bar{v}_e}, \; q_i^e \geq q_i^{v_e} + q_i^{\bar{v}_e} - 1, \tag{3}$$

where v_e and $\bar{v}_e$ denote the endpoints of e, with v_e having the smaller index and $\bar{v}_e$ the larger index. Further, if the direction of e is undetermined, i.e., $e \in E_u$, we introduce two binary variables $q_i^{(v_e, \bar{v}_e)}$ and $q_i^{(\bar{v}_e, v_e)}$ to represent the two possible

[2]The analogy between hydraulic and electric circuits underpins the use of circuit methods in microfluidics. A more detailed discussion will be introduced in Section V-D2

directions of e. Then, the following constraints ensure that every edge must have exactly one direction:

$$\underset{f_i\in F,}{\forall}\ \underset{e\in E_u}{\forall}: \quad q_i^{(v_e,\bar{v}_e)} + q_i^{(\bar{v}_e,v_e)} \le 1. \tag{4}$$

To determine which directed graph f_i is constructed on, we define a binary variable q_i^τ to indicate whether the undirected edges in p_i follow the directions assigned by a given $\tau \in T$:

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F}{\forall}: \quad q_i^\tau \ge \Sigma_i^\tau(E_u) - \sum_{e\in E_u} q_i^e + 1, \tag{5}$$

where $\Sigma_i^\tau(E_u)$ counts the number of undirected edges in E_u that follow τ with

$$\Sigma_i^\tau(E_u) = \sum_{e\in E_u} \big(1-\tau(e)\big)\cdot q_i^{(v_e,\bar{v}_e)} + \tau(e)\cdot q_i^{(\bar{v}_e,v_e)}. \tag{6}$$

Specifically, if all undirected edges in E_u that are part of p_i follow τ, i.e., $\Sigma_i^\tau(E_u) = \sum_{e\in E_u} q_i^e$, (5) limits $q_i^\tau = 1$. Using the *big M method* [36], the flow continuity can then be constrained as:

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{v\in D\cup S,}{\forall}\ \underset{b^{-1}(v)\in P_{\text{out}}}{\forall}: \quad \sum_{e\in I_v^\tau} q_i^e \ge 1 - (2 - q_i^v - q_i^\tau)\, M, \tag{7a}$$

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{v\in D\cup S,}{\forall}\ \underset{b^{-1}(v)\in P_{\text{in}},}{\forall}: \quad \sum_{e\in O_v^\tau} q_i^e \ge 1 - (2 - q_i^v - q_i^\tau)\, M, \tag{7b}$$

where M is an extremely large auxiliary constant. In other words, (7) ensures that each vertex in the flow path must have at least one incoming flow (except at inlets) and one outgoing flow (except at outlets). Specifically, if v is part of p_i in G^τ, i.e., $q_i^v = q_i^\tau = 1$, the right-hand sides of (7a) and (7b) become 1, ensuring the continuity of flow at v. Otherwise, the left-hand sides remain unconstrained.

Moreover, for every fluid transportation operation f_i, any device d that can receive fluids from $b(x_i)$ but does not have a directed path to $b(\bar{x}_i)$ must be excluded from p_i to prevent contamination. We define the set of such devices as the *exclusion set* $Z_i^\tau \equiv D \cap \big(D^\tau_{b(x_i)} \backslash A^\tau_{b(\bar{x}_i)} \backslash \{b(\bar{x}_i)\}\big)$ and introduce the following constraint to ensure the exclusion:

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{d\in Z_i^\tau}{\forall}: \quad q_i^d \le \big(1-q_i^\tau\big)M. \tag{8}$$

C. Residual Fluid Detection

Residual fluid refers to reaction products that inadvertently enter flow channels outside the midstream sub-path during a target operation, potentially contaminating subsequent operations. As introduced in Section II, valves control fluid movement by blocking flow along specific flow channels. However, valve-free flow channels outside the midstream sub-path are prone to residual fluid accumulation as they cannot be blocked by closing valves to prevent the ingress of reaction products. Consider the directed graph on the left in Fig. 6(g) as an example. When transporting fluid from mixer d_1 to outlet 2, the fluid traverses vertex 6, which also branches to vertex 7. As a result, a portion of fluid may flow toward 7; however, the unintended flow cannot be blocked by closing the valves. To address this issue, we introduce the following constraints to detect residual fluid and transport it to an outlet as waste.

In G^τ, an edge may retain residual fluid after f_i if it originates from a vertex capable of receiving reaction products and is neither part of the midstream sub-path nor equipped with a valve. Specifically, this occurs for valve-free edges connecting a vertex in M_i^τ to another vertex outside M_i^τ. We denote the set of such edges as R_i^τ and introduce the following constraint to formulate the residual fluid detection:

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{e\in R_i^\tau}{\forall}: \quad r_i^e \ge q_i^e + q_i^\tau - 1, \tag{9}$$

where binary variable r_i^e indicates whether e contains residual fluid after f_i. For the operation mentioned above, $M_i^{\tau_1}$ is $\{d_1, 6, 2\}$. Since edge $(6,7)$ is not equipped with a valve and, under τ_1, i.e., $q_i^{\tau_1} = 1$, its source vertex 6 belongs to $M_i^{\tau_1}$, while its endpoint 7 does not, we conclude that $(6,7) \in R_i^{\tau_1}$. As a result, if $(6,7)$ is part of p_i, i.e., $q_i^{(6,7)} = 1$, (9) limits $r_i^{(6,7)}$ to be 1, indicating the presence of residual fluid.

To export residual fluid, we construct flow path $p_{i\langle e\rangle}$ for transporting the residual fluid accumulated in edge e after f_i, denoted as $f_{i\langle e\rangle}$, using constraints (1) and (3)–(7). Since the destination (an outlet) of $f_{i\langle e\rangle}$ is not specified in $\mathcal{A}$, we define a binary variable $q_{i\langle e\rangle|o}$ for every outlet o to indicate whether it is selected to export the residual fluid. Then, the following constraint ensures that at least one outlet is selected:

$$\underset{f_i\in F,}{\forall}\ \underset{e\in R_i^\tau,}{\forall}\ \underset{b^{-1}(o)\in P_{\text{out}}}{\forall}: \quad \sum_{b^{-1}(o)\in P_{\text{out}}} q_{i\langle e\rangle|o} \ge r_i^e. \tag{10}$$

Meanwhile, instead of applying (2), we introduce the following constraint to ensure that e and the selected outlet o are in $p_{i\langle e\rangle}$:

$$\underset{f_i\in F,}{\forall}\ \underset{e\in R_i^\tau,}{\forall}\ \underset{b^{-1}(o)\in P_{\text{out}}}{\forall}: \quad q_{i\langle e\rangle}^e = 1, \quad q_{i\langle e\rangle}^o \ge q_{i\langle e\rangle|o}. \tag{11}$$

Moreover, the exclusion set in (8) for preventing devices from contamination is modified as $Z^\tau_{i\langle e\rangle|o} \equiv D \cap \big(D^\tau_{b(x_i)} \backslash A^\tau_o\big)$.

D. Conflict Identification

As introduced in Section II, parallel-executed fluid transportation operations may suffer two types of conflicts: *cross-contamination* and *sub-path blockage*. For any two operations f_i and f_j that are independent in the sequencing graph $\mathcal{A}$, we define a binary variable $c_{i,j}$ to indicate whether f_i conflicts with f_j. We then constrain the conditions under which $c_{i,j}$ is set to 1, indicating the existence of cross-contamination or sub-path blockage conflicts.

A cross-contamination conflict occurs when different fluids unintentionally mix at common vertices. In G^τ, since the fluid carried by each operation flows through its midstream sub-path, for any two operations f_i and f_j, the set of vertices at which their fluids may meet is $M_i^\tau \cap M_j^\tau$. Thus, the constraints for identifying cross-contamination conflicts can be formulated as:

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{f_j\in F_i^c,}{\forall}\ \underset{v\in M_i^\tau\cap M_j^\tau}{\forall}: c_{i,j} \ge q_i^v + q_i^\tau + q_j^v + q_j^\tau - 3, \tag{12}$$

where F_i^c is the set of operations that are independent from f_i according to $\mathcal{A}$, excluding the cases that f_i and f_j transport to-be-mixed fluids.

Besides, we introduce the following constraints to identify cross-contamination conflicts between an operation $f_i \in F$ and a residual fluid transportation operation $f_{j\langle e\rangle} \in F_i^c$.

$$\begin{aligned}\underset{\tau\in T,}{\forall}\ \underset{f_i,f_j\in F,}{\forall}\ \underset{e\in R_j^\tau,}{\forall}\ \underset{f_{j\langle e\rangle}\in F_i^c,}{\forall}\ \underset{b^{-1}(o)\in P_{\text{out}},}{\forall}\ \underset{v\in M_i^\tau\cap M_{j\langle e\rangle|o}^\tau}{\forall}:\\ c_{i,j\langle e\rangle} \geq q_i^v + q_i^\tau + q_{j\langle e\rangle}^v + q_{j\langle e\rangle}^\tau + q_{j\langle e\rangle|o} - 4.\end{aligned} \tag{13}$$

It is noteworthy that for two operations both transporting residual fluids, mixing their carried fluids does not lead to cross-contamination.

A sub-path blockage conflict occurs when the execution of one fluid transportation operation blocks the up- or downstream sub-paths of another operation, isolating it from the inlets or outlets. Specifically, a sub-path blockage conflict occurs when the flow paths of two parallel-executed operations share at least one common vertex in one operation's midstream sub-path and the other operation's up- or downstream sub-paths. To identify whether the execution of an operation f_i blocks the up- or downstream sub-paths of another operation f_j, we modify (12) by changing the vertex set to:

$$\Big\{v \in M_i^\tau \cap \big(U_j^\tau \cup W_j^\tau\big) \mid\ E_v \cap \big(\bar{M}_i^\tau \backslash \bar{U}_j^\tau \backslash \bar{W}_j^\tau\big) \neq \emptyset\Big\}.$$

Note that the midstream sub-path of a residual fluid transportation operation $f_{j\langle e\rangle}$ terminates at an outlet, leaving $f_{j\langle e\rangle}$ with no downstream sub-path. Thus, another fluid transportation operation can block only the upstream sub-path of $f_{j\langle e\rangle}$. Thus, the disruption caused by executing $f_i \in F$ to the execution of $f_{j\langle e\rangle}$, or vice versa, can be characterized using (13) with the vertex set adjusted to:

$$\begin{aligned}\Big\{v \in \big(M_i^\tau \cap U_{j\langle e\rangle|o}^\tau\big) \cup \big(M_{j\langle e\rangle|o}^\tau \cap (U_i^\tau \cup W_i^\tau)\big) \mid\\ E_v \cap \big(\bar{M}_i^\tau \backslash \bar{U}_{j\langle e\rangle|o}^\tau \cup \bar{M}_{j\langle e\rangle|o}^\tau \backslash \bar{U}_i^\tau \backslash \bar{W}_i^\tau\big) \neq \emptyset\Big\}.\end{aligned}$$

As previously mentioned, mixing residual fluids does not result in cross-contamination. Thus, when the midstream sub-path of one residual fluid transportation operation overlaps with the upstream sub-path of another, blocking the latter to prevent residual fluid from entering the other's midstream sub-path is unnecessary. Consequently, sub-path blockage conflicts do not arise between operations transporting residual fluids.

E. Bioassay Completion Time Minimization

Our execution model minimizes the completion time of a given bioassay by optimally assigning fluid transportation operations into batches, enabling non-conflicting operations to be executed in parallel.

To determine the batch indices of operations, we define binary variables $q_{i,k}$ to indicate whether an operation f_i is assigned to the k^{th} batch. The following constraints guarantee that every operation is executed exactly once:

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{e\in R_i^\tau}{\forall}:\quad \sum_{1\leq k\leq u_\ell} q_{i,k} = 1,\quad \sum_{1\leq k\leq u_\ell} q_{i\langle e\rangle,k} = r_i^e, \tag{14}$$

where u_ℓ denotes the upper bound of the batch indices with $u_\ell = |F| + \sum_{\tau\in T, f_i\in F} |R_i^\tau|$. Then, we introduce an integer variable ℓ_{f_i} to represent the batch index of f_i with:

$$\ell_{f_i} = \sum_{1\leq k\leq u_\ell} k \cdot q_{i,k}. \tag{15}$$

Without loss of generality, we let batches be executed in ascending order of their indices. Then, to match the execution order with the sequencing graph, we introduce the following constraints to ensure that the batch index of f_i must be smaller than any of its successors in the sequencing graph:

$$\underset{f_i\in F,}{\forall}\ \underset{f\in F_i^s}{\forall}:\quad \ell_{f_i} + 1 \leq \ell_f, \tag{16}$$

where F_i^s is the set of successors of f_i. Meanwhile, if an edge e contains residual fluid after f_i, i.e., $r_i^e = 1$, $f_{i\langle e\rangle}$ will be executed in the batch immediately following f_i:

$$\begin{aligned}\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{e\in R_i^\tau}{\forall}:\quad \ell_{f_{i\langle e\rangle}} &\leq \ell_{f_i} + 1 + (1 - r_i^e)\, M,\\ \ell_{f_{i\langle e\rangle}} &\geq \ell_{f_i} + 1 - (1 - r_i^e)\, M.\end{aligned} \tag{17}$$

As discussed in Section IV-D, conflicting operations must be executed in different batches, the constraints for which can be formulated as:

$$\underset{f_i\in\mathcal{F},}{\forall}\ \underset{f_j\in F_i^c,}{\forall}\ \underset{1\leq k\leq u_\ell}{\forall}:\ q_{i,k} + q_{j,k} \leq 1 + \big(1 - c_{i,j}\big)M. \tag{18}$$

Finally, to minimize the completion time of a given bioassay, the objective function is set to minimize the total number of batches. To this end, we introduce an integer variable n_ℓ and the following constraints for all operations to ensure that n_ℓ represents the largest batch index:

$$\underset{\tau\in T,}{\forall}\ \underset{f_i\in F,}{\forall}\ \underset{e\in R_i^\tau}{\forall}:\quad \ell_{f_i} \leq n_\ell,\quad \ell_{f_{i\langle e\rangle}} \leq n_\ell. \tag{19}$$

Thus, the overall optimization problem is modeled as

$$\begin{aligned}&\textit{Minimize}\quad n_\ell,\\ &\textit{subject to}\quad (1)\text{–}(19).\end{aligned}$$

After optimization, ParaVOM outputs a high-level protocol specifying a sequence of execution batches, denoted by $(\ell_n)_{1\leq n\leq n_\ell}$ with n_ℓ fixed to its minimum value. Each batch consists of parallel-executable operations, with flow paths specified. Within a batch ℓ_n, the vertices in the midstream sub-paths of the operations, as well as any valves on the edges in these sub-paths, are added to a target set $T_{\ell_n} \subseteq V$. To prevent contamination, we define an exclusion set

$$Z_{\ell_n} = \bigcup_{f_i\in\ell_n} Z_i^{\tau_{\ell_n}} \cup N_i^{\tau_{\ell_n}},$$

where τ_{ℓ_n} is the direction assignment function governing the operations in ℓ_n. Here, $N_i^{\tau_{\ell_n}} \subseteq N$ denotes the set of valves on the edges incident to the vertices in the midstream sub-paths of the operations in ℓ_n.

V. Channel-Level Synthesis

Using the high-level protocol, ParaVOM synthesizes and optimizes channel-level protocols to dynamically open and close valves to execute the operations. To support parallel-executed operations that asynchronously reach their destinations, ParaVOM performs an event-driven simulation to automatically update the high- and channel-level protocols.

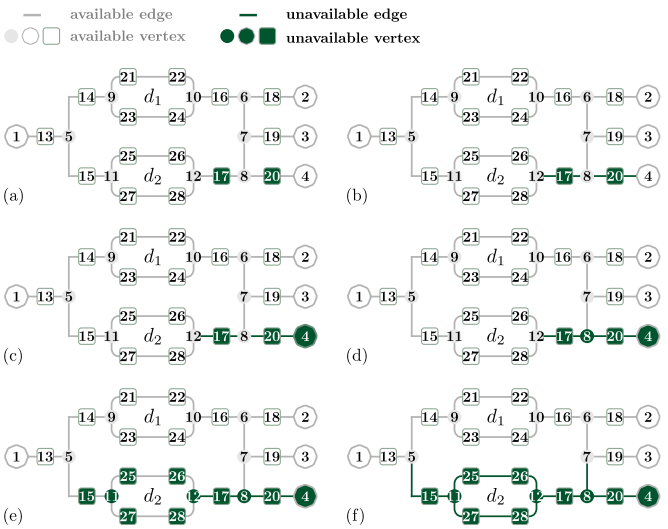


Fig. 7. Illustration of the recursive algorithm to modify the availability of the vertices and edges.

A. Flow Path Validation

The following introduces our flow path validation method, which is used for flow- and control-layer compatibility checks. In general, given a flow-layer structure $\mathcal{G}(V, E)$, we interpret the execution restrictions[3] for each fluid transportation operation as a set of initially unavailable vertices $U \subset V$, which are supposed to be blocked during the operation. We further develop a recursive algorithm to model the consequences of the blockage on the availability of other vertices and edges of the graph, and check whether the edges along the flow path required by the operation can be formed at the steady state. Specifically, our algorithm modifies the availability based on the following criteria:

1) An edge becomes unavailable if it contains an unavailable vertex.
2) A vertex becomes unavailable if it belongs to an unavailable edge and is not a flow channel branch.
3) A vertex becomes unavailable if it is connected to only one available edge and is neither an inlet nor an outlet.
4) A vertex becomes unavailable if it cannot be reached from any inlet or cannot reach any outlet.

Fig. 7 illustrates our algorithm using the flow-layer structure in Fig. 6(a), where vertices 17 and 20 are initially unavailable, as shown in Fig. 7(a). The first criterion states that a blocked flow channel segment cannot initiate or receive fluid flow due to either blocked endpoints or a valve closure. Consequently, edges $(12, 17)$, $(17, 8)$, $(20, 8)$, and $(4, 20)$ are specified unavailable, as shown in Fig. 7(b). The second criterion indicates that any chip component lacking a valid flow channel connection becomes unavailable. Thus, vertex 4, as an endpoint of the unavailable edge $(4, 20)$, is specified as unavailable, as shown in Fig. 7(c). The third criterion requires chip components to be connected to at least two available channels, except at inlets and outlets. Since vertex 8 has only one available neighbor, neither an inlet nor an outlet, it is unavailable, as shown in Fig. 7(d). The fourth criterion ensures that a valid flow path includes at least one inlet and one outlet to maintain a proper pressure difference. As a result, vertices 11, 12, 15, and 25–28 are unavailable, as shown in Fig. 7(e). By recursively applying these criteria until no further vertices or edges can be specified as unavailable, we find all vertices and edges available for forming the required flow path in Fig. 7(f).

Complexity Analysis. The method initializes all vertices and edges in $\mathcal{O}(|V|+|E|)$ time. The first three criteria propagate unavailability based on local connectivity, each requiring at most $\mathcal{O}(|E|)$ per iteration. The fourth criterion conducts two depth-first searches per available vertex to determine reachability from inlets and to outlets, yielding a worst-case cost of $\mathcal{O}(|V|^2 \cdot (|V| + |E|))$ per iteration. As the process may iterate up to $\mathcal{O}(|V|)$ times before convergence, the overall worst-case time complexity is $\mathcal{O}(|V|^3 \cdot (|V| + |E|))$.

B. Flow-Layer Compatibility Check

For each execution batch ℓ_n, we first check whether the flow-layer structure can support the target flow paths with the flow path validation method introduced in Section V-A by setting the initially unavaible vertices set $U_{\ell_n} = Z_{\ell_n} \cup T_{l_{(1,\ldots,n-1)}}$, where Z_{ℓ_n} denotes the exclusion set as introduced in Section IV-E and $T_{l_{(1,\ldots,n-1)}}$ denotes the set of destination vertices of fluid transportation operations from previous batches if the fluids in the associated devices are not transported during ℓ_n. Without loss of generality, when filling a mixer, the upper/left half-ring is filled first, followed by the lower/right half-ring. Consider the flow-layer structure in Fig. 7(a) as an example. Suppose that in batch ℓ_n, we want to transport fluid from inlet 1 to the lower half-ring of d_1 without contaminating d_2; besides, the upper half ring of d_1 has been filled in a previous batch and the fluids in it will not be transported in this batch, we have $Z_{\ell_n} = \{11, 12, 15\}$ and $T_{l_{(1,\ldots,n-1)}} = \{22\}$, and thus $U_{\ell_n} = \{11, 12, 15, 22\}$. Using $\mathcal{G}(V, E)$ and U_{ℓ_n}, ParaVOM performs flow path validation to identify available vertices and edges. If every vertex in T_{ℓ_n} remains available after validation, the flow-layer structure is concluded to be capable of supporting the target operations, and ParaVOM proceeds. Otherwise, ParaVOM outputs a warning message.

Complexity Analysis. Since each check involves a single invocation of the flow path validation method, its complexity is equivalent to that of the validation method.

C. Control-Layer Compatibility Check

Upon passing the flow-layer compatibility check, ParaVOM constructs a search tree to collect all channel-level protocols supporting the target flow paths, inspired by the *branch and cut* [37] approach commonly used in solving ILP problems.

Apart from a virtual root (level-0), each node in the search tree represents a pressurization option, where a level-1 node represents a single pressurized control channel, a level-2 node represents a pair of pressurized control channels formed by combining two level-1 nodes, and so forth. We then perform a *depth-first search* [38] to traverse the tree. Each node $c \subseteq C$ undergoes flow path validation, where the initially unavailable

[3]The restrictions are derived from the input sequencing graph $\mathcal{A}$ and the high-level protocol synthesized in the first stage, as detailed in Section V-B and Section V-C.

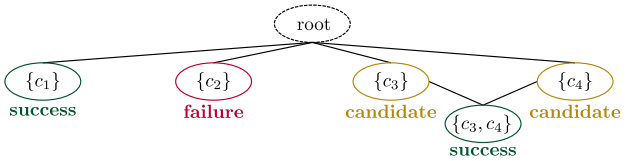


Fig. 8. Illustration of synthesizing valid channel-level protocols.

vertex set U_{ℓ_n} contains all valves actuated by the control channels in c. Namely, all channels in c are assumed to be pressurized, and the corresponding valves are treated as closed. Based on validation outcomes, nodes are classified into three categories:

- If any vertex in the target set T_{ℓ_n} is unavailable, classify the node as a *failure*.
- If all vertices in T_{ℓ_n} are available and all vertices in Z_{ℓ_n} are unavailable, classify the node as a *success*.
- If all vertices in T_{ℓ_n} and at least one vertex in Z_{ℓ_n} are available, classify the node as a *candidate*.

After processing nodes at a given level, we prune the search tree by removing descendants of nodes classified as failures or successes. Specifically, a failure node indicates an unexpected blockage, implying its descendants will also lead to the same blockage and are thus eliminated. On the other hand, a success node is sufficient to support the target flow paths, so additional control channels are unnecessary, and its descendants are also pruned. Once all nodes are traversed, the success nodes are collected as potential channel-level protocols.

Example. Fig. 8 illustrates our synthesis method for a chip design with four control channels. Firstly, a search tree is constructed, followed by flow path validation for each level-1 node. Based on validation results, the set $\{c_1\}$ is classified as a success and $\{c_2\}$ as a failure. Thus, we eliminate all descendants of $\{c_1\}$ and $\{c_2\}$ from the tree, leaving $\{c_3, c_4\}$ as the only remaining level-2 node. After subsequent validation, $\{c_3, c_4\}$ is also classified as a success. Finally, we collect $\{c_1\}$ and $\{c_3, c_4\}$ as potential channel-level protocols.

Complexity Analysis. Each node requires a flow path validation, incurring a per-call cost of $\mathcal{O}(|V|^3 \cdot (|V|+|E|))$. To explore all nodes up to level-d, the algorithm considers at most $\mathcal{O}(|C|^d)$ combinations of control channels, each corresponding to a distinct validation call. Consequently, the worst-case time complexity is $\mathcal{O}(|C|^d \cdot |V|^3 \cdot (|V| + |E|))$.

D. Channel-Level Optimization

ParaVOM provides two optimization criteria for selecting a channel-level protocol. Users can choose one of these criteria based on specific application demands.

1) Resource-Oriented Optimization: The first criterion aims to minimize the number of pressurized control channels. Control channels never pressurized during execution are considered *redundant*. Eliminating a control channel also removes all associated valves and the control pin, thus effectively reducing redundancy in the control layer. The following describes our ILP method to solve this optimization problem.

Considering a batch ℓ_n, we define C_{ℓ_n} as the set of potential channel-level protocols to execute operations within ℓ_n. We then introduce the following constraints to ensure that exactly one option must be selected to execute ℓ_n:

$$\forall_{1 \le n \le n_\ell} : \quad \sum_{c \in C_{\ell_n}} q_{\ell_n, c} = 1, \tag{20}$$

where binary variable $q_{\ell_n,c}$ indicates whether channel-level protocol c is selected to execute ℓ_n. Further, all control channels within a selected channel-level protocol c must be pressurized.

$$\forall_{1 \le n \le n_\ell,\ c \in C_{\ell_n},\ c_k \in c} : \quad q_{c_k} \ge q_{\ell_n, c}, \tag{21}$$

where binary variable q_{c_k} indicates whether control channel c_k is pressurized. Finally, to minimize the control-layer redundancy, the objective function is set to minimize the number of pressurized control channels.

$$\textit{Minimize} \quad \sum_{c_k \in C} q_{c_k},$$
$$\textit{subject to} \quad (20),\ (21).$$

2) Time-Oriented Optimization: The second criterion aims to minimize the fluid transportation time. In micro- and nano-scale environments, fluid flows are generally viscous, incompressible, and laminar [39], allowing the fluid motion to be predicted using straightforward calculations. Specifically, the hydraulic behavior of pressure-driven flow follows *Hagen-Poiseuille's law*, analogous to *Ohm's law* in electric circuits, where pressure drop corresponds to voltage, volumetric flow rate to current, and hydraulic resistance to electrical resistance [40]. To minimize transportation time, the flow rate should be maximized by minimizing hydraulic resistance under a constant input pressure.

Given an input pressure Δp and the a fluid resistor with hydraulic resistance r, the volumetric flow rate Q $[\mathrm{m}^3\,\mathrm{s}^{-1}]$ can be calculated as:

$$Q = \frac{\Delta p}{r}. \tag{22}$$

In microfluidic networks, where most channels are rectangular, the hydraulic resistance r can be approximated as:

$$r = \frac{12 \mu l}{w h^3}, \tag{23}$$

where μ is the viscosity of the fluid, and l, h, and w are the length, height, and width of the channel. For fluidic resistors connected in serial, the effective hydraulic resistance is the sum of the resistance of each resistor. For fluidic resistors connected in parallel, the effective hydraulic resistance equals the reciprocal of the sum of the reciprocals of the resistance of each resistor.

Further, a microfluidic network forms a bridge configuration [41] when a cross-connection or "bridge" is placed between resistors, as shown in Fig. 9(a). In such cases, $Y\Delta$ and ΔY conversions [42], which convert between Y and Δ configurations, are used to calculate resistance.

$Y\Delta$ conversion: In Fig. 9(b), vertex 2 has neighbors 1, 3, and 4. The $Y\Delta$ conversion removes 2 and adds edges between vertices 1, 3, and 4, forming a triangle, as shown in Fig. 9(c). The resistances in the newly formed triangle are calculated as:

$$r_a = \frac{r_p}{r_3}, \quad r_b = \frac{r_p}{r_1}, \quad r_c = \frac{r_p}{r_2}, \tag{24}$$

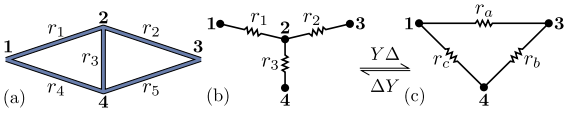


Fig. 9. (a) A microfluidic network connected in a bridge configuration. (b)(c) $Y\Delta$ and ΔY conversions used for simplifying network analysis.

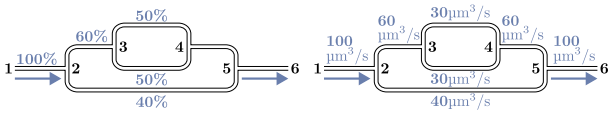


Fig. 10. An example of the flow rate distribution model.

where $r_p = r_1r_2 + r_1r_3 + r_2r_3$.

ΔY conversion: In Fig. 9(c), vertices 1, 3, and 4 form a triangle Δ. The ΔY conversion removes the original edges and adds a new vertex 2 connected to vertices 1, 3, and 4. The resistances at vertex 2 are calculated as:

$$r_1 = \frac{r_a r_c}{r_s}, \quad r_2 = \frac{r_a r_b}{r_s}, \quad r_3 = \frac{r_b r_c}{r_s}, \tag{25}$$

where $r_s = r_a + r_b + r_c$.

Using $Y\Delta$ and ΔY conversions, the microfluidic network can be simplified to a structure involving only series and parallel configurations, enabling easy resistance calculation. Finally, the potential channel-level protocol with the lowest hydraulic resistance is selected.

E. Simulation: Event-driven Protocol Update

ParaVOM simulates and updates the high-level and channel-level protocols for parallel-executed transportation operations that asynchronously reach their destinations.

As fluid velocity varies across different flow channels, we determine the flow rate distribution to accurately predict transportation delays. To this end, we build a hierarchical model of the flow paths with the validated graph. For example, Fig. 10 shows a flow path $1 \to 6$ consisting of three sequentially connected sub-paths $1 \to 2$, $2 \to 5$, and $5 \to 6$, among which the sub-path $2 \to 5$ again consists of two parallel sub-paths. Using Hagen-Poiseuille's law, we calculate the hydraulic resistance of the sub-paths based on the resistance of the underlying edges and thus derive the flow rate distribution among the sub-paths. Specifically, the upper sub-path $2 \to 3 \to 4 \to 5$ has a lower resistance than the bottom sub-path $2 \to 5$. As a result, the upper sub-path will inherit 60% of the volumetric flow rate from its predecessor path, while the bottom sub-path will only inherit 40%. And since $2 \to 3 \to 4 \to 5$ contains a pair of parallel edges $(3, 4)$ with balanced resistance, the volumetric flow rate is further evenly divided. Thus, for an input flow rate of $100\,\mu\text{m}^3/\text{s}$, the volumetric flow rate at each parallel edges $(3, 4)$ will be calculated as $100\,\mu\text{m}^3/\text{s} \cdot 60\% \cdot 50\% = 30\,\mu\text{m}^3/\text{s}$.

After computing the flow rate distribution, we perform a modified *list-scheduling algorithm* [43] to simulate the fluid status over time by processing edges sequentially, such that an edge is processed only if at least one of its predecessor edges is filled with fluids. After processing each edge, the simulation pauses and generates an event signal indicating the vertex v_r that has just been reached by fluids. We then determine the next steps based on three different scenarios:

1) If v_r is not a destination, the simulation continues without changing any configuration.
2) If v_r is a destination and all destination vertices are reached, the simulation stops.
3) If v_r is a destination and at least one other destination has not been reached by fluids, the simulation pauses to update the high-level protocol for ℓ_n by moving v_r from T_{ℓ_n} to U_{ℓ_n} and removing the exclusion set of the completed operation from U_{ℓ_n}. The updated protocol is then fed into the channel-level synthesis.

The fluid status from the previous process is inherited whenever a new simulation starts, and the event-driven protocol update mechanism is repeated.

Complexity Analysis. The simulation processes each edge at most once, leading to a time complexity of $\mathcal{O}(|E|)$ per simulation phase. As the number of destination vertices is typically small, the number of event-driven protocol updates is bounded by a constant in practice. Hence, the total complexity is dominated by the control-layer compatibility check.

VI. Experimental Results

We investigate the performance of ParaVOM using four chip designs synthesized by a state-of-the-art physical design tool Columba 2.0 [12]. The features of the test cases are summarized in Table I.

Our work was implemented using C++, and optimizations were performed on a computer with an Apple M1 8-core CPU. The ILP model is solved with the Gurobi Optimizer [44]. In our experimental setup, the input pressure Δp was set to $10\,\text{Pa}$, the fluid viscosity μ was $0.000\,89\,\text{Pa} \cdot \text{s}$, and the flow channel dimensions were $50\,\mu\text{m}$ in height and $100\,\mu\text{m}$ in width.

A. Test Cases

The first chip design is shown in Fig. 11(a). Two binding functions were applied, denoted by cases 1^a and 1^b, resulting in different numbers of undirected edges. This chip design was tested using a sequencing graph involving five fluid transportation operations, two of which can be combined as a fluid-multiplexing operation.

TABLE I
Test Cases Used in the Experiments

Case	$\|P\|$	$\|D\|$	$\|S\|$	$\|C\|$	$\|N\|$	$\|E\|$	$\|E_u\|$	$O(\|V\|^3 \cdot (\|V\| + \|E\|))$
1^a	4	2	4	13	16	24	1	10^5
1^b							0	
2	5	3	9	12	34	51	0	10^7
3^a	7	4	13	15	38	60	1	10^7
3^b				17				
4	23	21	43	31	145	222	0	10^9

$|P|$: number of flow ports; $|D|$: number of devices; $|S|$: number of flow channel branches; $|C|$: number of control channels; $|N|$: number of valves; $|E|$: number of edges; $|E_u|$: number of undirected edges; $O(|V|^3 \cdot (|V| + |E|))$: worst-case time complexity of flow path validation; shown values are estimated magnitudes.

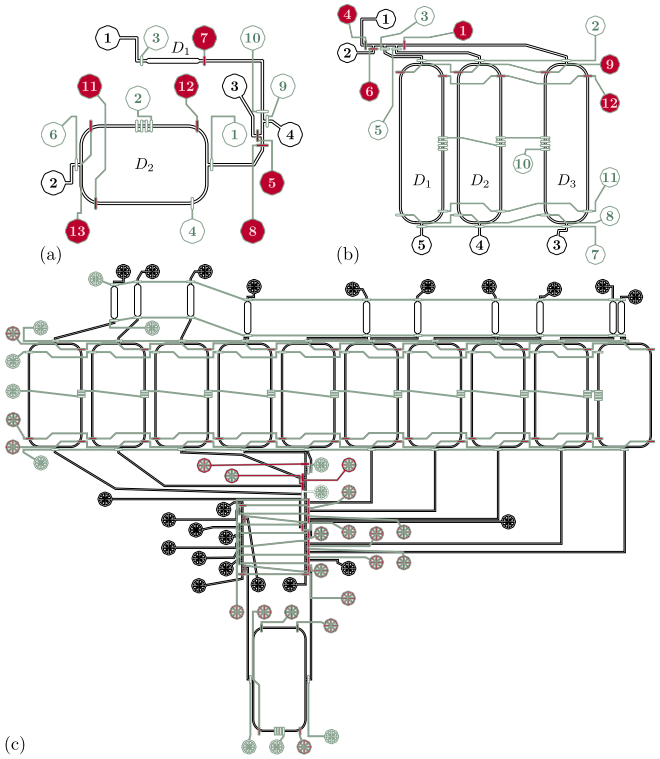


Fig. 11. Synthesis results using ParaVOM for cases (a) 1^a, (b) 2, and (c) 4, where flow channels are shown in black, control channels in green, and redundant control channels in red.

The second chip design is shown in Fig. 11(b). This chip design was tested using a sequencing graph with nine fluid transportation operations. In particular, six of these operations, which involve fluid flows from two inlets to the half-ring-shaped channels in the three mixers, can be combined as two fluid-multiplexing operations.

The third chip design, case 3^a, is shown in Fig. 2(b). This application-specific design, proposed in [12], uses an aggressive pressure-sharing strategy to minimize the number of control pins, resulting in a relatively larger number of sequentially connected valves. A modified version, case 3^b, was created by adding two additional control channels to independently pressurize the conflicting valves identified using the preliminary method VOM [45]. This design was tested with two sequencing graphs: the first, denoted by cases 3^a_1 and 3^b_1, contains four fluid transportation operations, none of which can be combined, while the second, denoted by cases 3^a_2 and 3^b_2, contains eight operations, two of which can be combined as a fluid-multiplexing operation.

The fourth chip design is shown in Fig. 11(c). This design was tested with a sequencing graph involving 42 fluid transportation operations. In particular, 20 operations involving fluid flows from the lower mixer and an inlet to the ten upper mixers can be combined as three fluid-multiplexing operations.

B. Performance Comparison

The preliminary work [45] VOM does not support the parallel execution of non-conflicting operations. Besides, VOM neglects the bridge configuration when calculating the hydraulic resistance. For comparison, we incorporate the new resistance calculation model into VOM. Table II presents the synthesis results comparing VOM and ParaVOM.

Feasibility Check. VOM successfully validates flow paths for cases 1, 2, and 3^b. For case 3^a, VOM fails to validate the flow path for transporting fluid from an inlet to the half-ring-shaped channels in one of the mixers. For case 4, VOM does not identify any feasible channel-level protocol within a 24-hour runtime. ParaVOM successfully validates flow paths for all cases. For case 3^a, ParaVOM avoids the incompatibility encountered by VOM by identifying that operations transporting different fluids to the same mixer for mixing are conflict-free and can thus be grouped into the same batch.

Channel-Level Optimization. As shown in Table II, compared to VOM, ParaVOM requires fewer control channels to execute the same fluid transportation operations, resulting in a cleaner control layer, reduced chip size, and improved robustness. In particular, compared to the original designs, ParaVOM reduces the average number of control channels by 49%; and compared to VOM, ParaVOM identifies one more redundant control channel for test cases 1^a and 3^a_2. We illustrate the synthesis results of ParaVOM using the resource-oriented criterion for cases 1^a, 2, 3^a_1, and 4 in Figs. 11(a), 11(b), 12(a), and 11(c), respectively.

Parallel execution of non-conflicting operations also significantly reduces transportation time. On average, ParaVOM achieves a 19% reduction compared to VOM, and the benefit becomes more pronounced as the number of fluid-multiplexing operations decreases, with a maximum improvement of 39%.

Moreover, we assess the trade-off between resource usage and transportation time by comparing synthesis outcomes under different optimization criteria. When resource minimization is prioritized, compared to the time-oriented setting, VOM incurs an average transportation time increase of 13%, whereas ParaVOM limits this overhead to 1%. On the other hand, when transportation time minimization is prioritized, compared to

TABLE II
SYNTHESIS RESULT BASED ON DIFFERENT OPTIMIZATION CRITERIA

VOM							
Case	$\|\mathcal{F}_M\|$	n_u	Resource-Oriented		Time-Oriented		
			$\|\mathcal{C}_p\|$	T (s)	$\|\mathcal{C}_p\|$	T (s)	
1^a	1	1	**8**	1.2	8	**1.1**	
1^b			**7**	1.6	7	**1.1**	
2	2	4	**7**	3.1	7	**2.9**	
3^b_1	1	1	**8**	2657.4	9	**2624.6**	
3^b_2	1	1	**9**	4717.1	13	**4557.8**	
ParaVOM							
Case	$\|\mathcal{F}_\ell\|$	n_u	Resource-Oriented		Time-Oriented		
			$\|\mathcal{C}_p\|$	T (s)	$\|\mathcal{C}_p\|$	T (s)	r_T (%)
1^a	2-2-1	2	**7**	1.0	7	**1.0**	11.29
1^b			**7**	0.9	7	**0.9**	14.59
2	3-3-3	6	**7**	2.7	7	**2.5**	13.63
3^a_1	2-1-2	3	**8**	1592.3	8	**1592.3**	39.33
3^a_2	2-1-3-1-1	3	**8**	3802.3	11	**3798.0**	15.96
4	12-10-10-10	2	**11**	1588.01	11	**1587.37**	–

$|\mathcal{F}_M|$: number of fluid-multiplexing transportation operations; n_u: number of event-driven protocol updates; $|\mathcal{C}_p|$: number of pressurized control channels during the execution of the bioassay; T: transportation time in seconds; $|\mathcal{F}_\ell|$: number of fluid transportation operations in a batch, listed in execution order.; r_T: percentage reduction in the transportation time as compared to VOM.

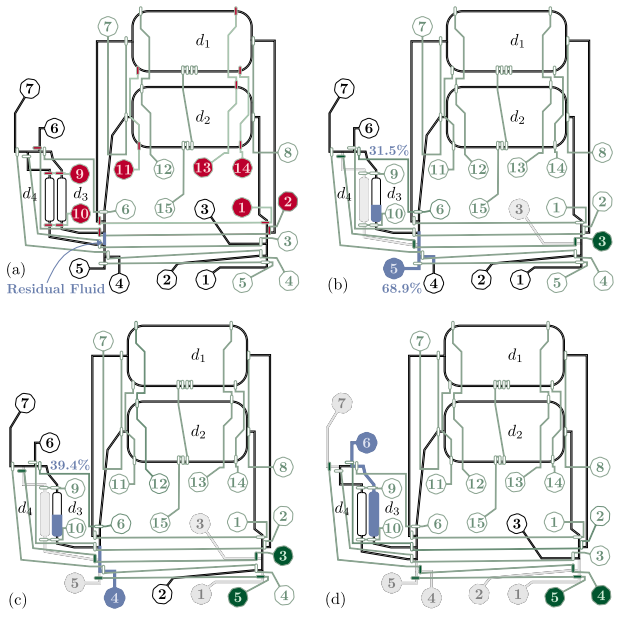


Fig. 12. Synthesis results using ParaVOM for case 3_1^a. Flow channels are shown in black, control channels in light green, redundant control channels in red, blocked flow channels in grey, pressurized control channels in dark green, and fluids in blue. (a) A chip design from [12] with seven redundant control channels. (b)–(d) Event-driven updates of the channel-level protocol during the third batch. The numbers represent the percentage of the volume of the corresponding flow channel segment that fluids have occupied.

the resource-oriented setting, the number of pressurized control channels increases by 11% in VOM and 6% in ParaVOM. These results highlight the ability of ParaVOM to balance conflicting objectives, achieving substantial improvement in one aspect with minimal compromise in the other.

Residual Fluid Transportation and Event-driven Protocol Update. In case 3_1^a, ParaVOM detected residual fluids on an edge, highlighted in blue in Fig. 12(a). Specifically, transporting fluids from d_1 to d_3 caused residual fluids to accumulate on that edge. In the next batch, the accumulated fluids are exported to outlets 4 and 5, along with the operation from d_3 to outlet 6. The event-driven protocol updates are shown in Figs. 12(b)–12(d). At the start of execution, control channel c_3 is pressurized to establish the flow paths while preventing contamination of non-target device d_4, as shown in Fig. 12(b). When the residual fluids reach outlet 5, ParaVOM detects two remaining sub-paths still in progress. Thus, it pressurizes control channel c_5, stopping fluid transportation in the completed sub-path without disturbing the others, as shown in Fig. 12(c). Similarly, after the residual fluids reach outlet 4, ParaVOM pressurizes control channel c_4, as shown in Fig. 12(d). Meanwhile, as the remaining operation poses no contamination risk to d_4, ParaVOM depressurizes c_3, allowing d_3 to remain connected to an inlet 3.

VOM cannot detect the residual fluid accumulation in flow channels. Thus, when using VOM to synthesize the channel-level protocol for case 3_1^b, we first attempted to add the start point of the residual-fluid-accumulated edge to the exclusive set for transporting fluids from d_1 to d_3 to prevent the accumulation. However, this led to no feasible channel-level protocol. To resolve this, we added a new operation into the high-level protocol (before the operation from d_3 to outlet 6) to transport the residual fluid to outlets 4 and 5. Since VOM does not support parallel execution, the total transportation time significantly increases compared to ParaVOM.

C. Runtime Analysis

Table III summarizes the runtime. The dominant contributors are flow- and control-layer compatibility checks, both relying on the flow path validation method as a core subroutine; their invocation counts are listed in parentheses. In general, designs with more control channels result in deeper search trees and, thus, more validation calls during the control-layer compatibility check. It is worth mentioning that although the ILP optimization problems are NP-hard, both ILP models in our high- and channel-level synthesis methods can be solved within a few seconds for all test cases. Table III also presents the average runtime per validation under different design complexities. As discussed in Section V-A, the worst-case time complexity of flow path validation is $\mathcal{O}(|V|^3 \cdot (|V|+|E|))$, as summarized in Table I. The two tables reveal a clear positive correlation between the flow-layer complexity and the runtime.

ParaVOM completes all test cases in under 90 seconds, except for case 4, where the high structural complexity of the flow layer results in a longer runtime per validation. On average, ParaVOM achieves a 38% reduction in runtime compared to VOM for the first three designs. This improvement is

TABLE III
RUNTIME COMPARISON

Case		1^a	1^b	2	3_1^b/3_1^a	3_2^b/3_2^a	4
VOM: Channel-level Synthesis							
Resource-Oriented	runtime (s)	6.22	6.69	46.20	568.57	207.89	–
	FLCC (s)	0.26	0.22	1.64	3.24	3.73	–
	(#FPV)	(13)	(12)	(26)	(14)	(16)	–
	CLCC (s)	4.69	4.90	41.71	562.76	201.32	–
	(#FPV)	(301)	(282)	(705)	(5285)	(1522)	–
	ILP (s)	0.01	0.00	0.01	0.01	0.00	–
	others (s)	1.26	1.57	2.84	2.56	2.83	–
Time-Oriented	runtime (s)	4.20	4.28	21.41	283.45	204.09	–
	FLCC (s)	0.08	0.09	0.67	1.43	1.91	–
	(#FPV)	(5)	(5)	(9)	(6)	(8)	–
	CLCC (s)	3.10	3.13	18.82	279.63	197.56	–
	(#FPV)	(220)	(208)	(328)	(2507)	(1522)	–
	others (s)	1.03	1.06	1.92	2.39	4.62	–
Avg. (s)		0.02		0.06	0.11	0.13	–
ParaVOM: High-Level Synthesis							
ILP (s)		0.16	0.03	0.52	0.61	5.34	688.29
ParaVOM: Channel-Level Synthesis							
Resource-Oriented	runtime (s)	5.35	5.42	51.70	48.08	75.34	8791.80
	FLCC (s)	0.19	0.20	1.94	2.69	3.76	185.87
	(#FPV)	(13)	(14)	(30)	(12)	(17)	(14)
	CLCC (s)	3.96	3.95	47.94	44.54	69.04	8354.22
	(#FPV)	(308)	(301)	(802)	(351)	(532)	(501)
	ILP (s)	0.01	0.01	0.01	0.01	0.01	0.02
	other (s)	1.19	1.26	1.81	0.84	2.52	251.70
Time-Oriented	runtime (s)	3.25	3.04	20.89	47.01	62.80	6404.23
	FLCC (s)	0.12	0.10	0.78	1.35	1.88	78.93
	(#FPV)	(5)	(5)	(9)	(6)	(8)	(6)
	CLCC (s)	2.40	2.46	18.75	44.82	59.00	6087.59
	(#FPV)	(220)	(208)	(328)	(351)	(472)	(353)
	other (s)	0.72	0.49	1.36	0.84	2.10	237.72
Avg. (s)		0.01		0.06	0.13		16.83

FLCC: runtime of flow-layer compatibility check; #FPV: number of flow path validation method invocations; CLCC: runtime of control-layer compatibility check; ILP: runtime for solving the ILP model; other: remaining runtime excluding key algorithms; Avg.: average runtime of each validation.

primarily due to the reduced number of channel-level synthesis invocations enabled by parallel execution. Whereas VOM conducts channel-level synthesis individually for each fluid transportation operation, ParaVOM performs it once per batch; since the number of batches is substantially smaller, the overall runtime is significantly reduced. However, for chip designs with highly complex flow-layer structures and sequencing graphs, ParaVOM may exhibit noticeably increased runtime, which, while acceptable in practice, suggests potential for future improvements in scalability and efficiency.

VII. Conclusion

This work proposes ParaVOM, a two-stage synthesis method, to bridge the gap between state-of-the-art high-level synthesis and physical design approaches based on different optimization criteria. In general, ParaVOM identifies parallel-executable fluid transportation operations and allows them to be executed in batches to improve fluid transportation efficiency; synthesizes channel-level protocols to construct the flow paths; and detects mismatches and redundancy usage in the input design. Experimental results confirm that, compared with the preliminary work, ParaVOM enables the execution of given bioassays that were previously unachievable on given chip designs, significantly reduces fluid transportation times, lowers the number of required pressurized control channels, and requires much less runtime.

References

[1] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake, "Monolithic microfabricated valves and pumps by multilayer soft lithography," *Science*, vol. 288, no. 5463, pp. 113–116, 2000. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.288.5463.113

[2] J. W. Hong, V. Studer, G. Hang, W. F. Anderson, and S. R. Quake, "A nanoliter-scale nucleic acid processor with parallel architecture," *Nature Biotechnology*, vol. 22, no. 4, pp. 435–439, 2004. [Online]. Available: https://doi.org/10.1038/nbt951

[3] N. Compera, S. Atwell, J. Wirth, C. von Törne, S. M. Hauck, and M. Meier, "Adipose microtissue-on-chip: a 3d cell culture platform for differentiation, stimulation, and proteomic analysis of human adipocytes," *Lab Chip*, vol. 22, pp. 3172–3186, 2022. [Online]. Available: http://dx.doi.org/10.1039/D2LC00245K

[4] R. Rodriguez-Moncayo, D. F. Cedillo-Alcantar, P. E. Guevara-Pantoja, O. G. Chavez-Pineda, J. A. Hernandez-Ortiz, J. U. Amador-Hernandez, G. Rojas-Velasco, F. Sanchez-Muñoz, D. Manzur-Sandoval, L. D. Patino-Lopez, D. A. May-Arrioja, R. Posadas-Sanchez, G. Vargas-Alarcon, and J. L. Garcia-Cordero, "A high-throughput multiplexed microfluidic device for covid-19 serology assays," *Lab Chip*, vol. 21, pp. 93–104, 2021. [Online]. Available: http://dx.doi.org/10.1039/D0LC01068E

[5] A. R. Vollertsen, D. de Boer, S. Dekker, B. A. M. Wesselink, R. Haverkate, H. S. Rho, R. J. Boom, M. Skolimowski, M. Blom, R. Passier, A. van den Berg, A. D. van der Meer, and M. Odijk, "Modular operation of microfluidic chips for highly parallelized cell culture and liquid dosing via a fluidic circuit board," *Microsystems & Nanoengineering*, vol. 6, no. 1, p. 107, 2020. [Online]. Available: https://doi.org/10.1038/s41378-020-00216-z

[6] J. Melin and S. R. Quake, "Microfluidic large-scale integration: the evolution of design rules for biological automation." *Annu Rev Biophys Biomol Struct*, vol. 36, pp. 213–231, 2007.

[7] C.-C. Lee, G. Sui, A. Elizarov, C. J. Shu, Y.-S. Shin, A. N. Dooley, J. Huang, A. Daridon, P. Wyatt, D. Stout, H. C. Kolb, O. N. Witte, N. Satyamurthy, J. R. Heath, M. E. Phelps, S. R. Quake, and H.-R. Tseng, "Multistep synthesis of a radiolabeled imaging probe using integrated microfluidics." *Science*, vol. 310, no. 5755, pp. 1793–1796, Dec 2005.

[8] M. Li, T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 624–629.

[9] ——, "Component-oriented high-level synthesis for continuous-flow microfluidics considering hybrid-scheduling," in *Proceedings of the 54th Annual Design Automation Conference*, ser. DAC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3061639.3062213

[10] F. Zuo, M. Li, T.-M. Tseng, T.-Y. Ho, and U. Schlichtmann, "Relative-scheduling-based high-level synthesis for flow-based microfluidic biochips," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–9.

[11] T.-M. Tseng, M. Li, B. Li, T.-Y. Ho, and U. Schlichtmann, "Columba: co-layout synthesis for continuous-flow microfluidic biochips," in *Proceedings of the 53rd Annual Design Automation Conference*, ser. DAC '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2897937.2897997

[12] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1588–1601, 2018.

[13] T.-M. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T.-Y. Ho, and U. Schlichtmann, "Columba s: A scalable co-layout design automation tool for microfluidic large-scale integration," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[14] W. H. Minhass, P. Pop, and J. Madsen, "System-level modeling and synthesis of flow-based microfluidic biochips," in *2011 Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, 2011, pp. 225–233.

[15] W. H. Minhass, J. McDaniel, M. Raagaard, P. Brisk, P. Pop, and J. Madsen, "Scheduling and fluid routing for flow-based microfluidic laboratories-on-a-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 615–628, 2018.

[16] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proceedings of the 52th Annual Design Automation Conference*, 2015, pp. 1–6.

[17] T.-M. Tseng, B. Li, M. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis with dynamic device mapping and fluid routing for flow-based microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 1981–1994, 2016.

[18] S. Liang, M. Lian, M. Li, T.-M. Tseng, U. Schlichtmann, and T.-Y. Ho, "Armm: Adaptive reliability quantification model of microfluidic designs and its graph-transformer-based implementation," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[19] T.-M. Tseng, B. Li, U. Schlichtmann, and T.-Y. 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, and U. Schlichtmann, "Transport or store? synthesizing flow-based microfluidic biochips using distributed channel storage," in *Proceedings of the 54th Annual Design Automation Conference*, 2017, pp. 1–6.

[21] X. Huang, W. Guo, Z. Chen, B. Li, T.-Y. Ho, and U. Schlichtmann, "Flow-based microfluidic biochips with distributed channel storage: Synthesis, physical design, and wash optimization," *IEEE Transactions on Computers*, vol. 71, no. 2, pp. 464–478, 2022.

[22] 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 Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 55–68, 2017.

[23] Y. Zhu, B. Li, T.-Y. Ho, Q. Wang, H. Yao, R. Wille, and U. Schlichtmann, "Multi-channel and fault-tolerant control multiplexing for flow-based microfluidic biochips," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[24] X. Huang, T.-Y. Ho, W. Guo, B. Li, and U. Schlichtmann, "Minicontrol: Synthesis of continuous-flow microfluidics with strictly constrained control ports," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317864

[25] X. Huang, T.-Y. Ho, Z. Li, G. Liu, L. Wang, Q. Li, W. Guo, B. Li, and U. Schlichtmann, "Minicontrol 2.0: Co-synthesis of flow and control layers for microfluidic biochips with strictly constrained control ports,"

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 12, pp. 5449–5463, 2022.

[26] X. Huang, Y. Pan, G. L. Zhang, B. Li, W. Guo, T.-Y. Ho, and U. Schlichtmann, "Pathdriver: a path-driven architectural synthesis flow for continuous-flow microfluidic biochips," in *2020 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, ser. ICCAD '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3400302.3415725

[27] ——, "Pathdriver+: Enhanced path-driven architecture design for flow-based microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 7, pp. 2185–2198, 2022.

[28] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1463–1475, 2014.

[29] M. Li, Y. Zhang, J. Y. Lee, H. Gasvoda, I. E. Araci, T.-M. Tseng, and U. Schlichtmann, "Integrated test module design for microfluidic large-scale integration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 6, pp. 1939–1950, 2023.

[30] M. Li, H. Gu, Y. Zhang, S. Liang, H. Gasvoda, R. Altay, I. E. Araci, T.-M. Tseng, T.-Y. Ho, and U. Schlichtmann, "Late breaking results: Efficient built-in self-test for microfluidic large-scale integration (mlsi)," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, ser. DAC '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3649329.3663489

[31] G. Firpo, E. Angeli, L. Repetto, and U. Valbusa, "Permeability thickness dependence of polydimethylsiloxane (pdms) membranes," *Journal of Membrane Science*, vol. 481, pp. 1–8, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376738814009466

[32] A. R. Wu, J. B. Hiatt, R. Lu, J. L. Attema, N. A. Lobo, I. L. Weissman, M. F. Clarke, and S. R. Quake, "Automated microfluidic chromatin immunoprecipitation from 2,000 cells." *Lab Chip*, vol. 9, no. 10, pp. 1365–1370, May 2009.

[33] Stanford Microfluidics Foundry. Basic design tips. [Online]. Available: https://www.stanfordmicrofluidics.com/design-basics

[34] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1463–1475, 2014.

[35] K. Chakrabarty and J. Zeng, "Design automation for microfluidics-based biochips," *J. Emerg. Technol. Comput. Syst.*, vol. 1, no. 3, pp. 186–223, oct 2005. [Online]. Available: https://doi.org/10.1145/1116696.1116698

[36] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization: Second Edition*. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[37] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM Review*, vol. 33, no. 1, pp. 60–100, 1991. [Online]. Available: http://www.jstor.org/stable/2030652

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009, ch. 22.3 Depth-first search, pp. 603–612.

[39] H. A. Stone, *Introduction to Fluid Dynamics for Microfluidic Flows*. Boston, MA: Springer US, 2007, pp. 5–30. [Online]. Available: https://doi.org/10.1007/978-0-387-68913-5_2

[40] K. W. Oh, K. Lee, B. Ahn, and E. P. Furlani, "Design of pressure-driven microfluidic networks using electric circuit analogy," *Lab Chip*, vol. 12, pp. 515–545, 2012. [Online]. Available: http://dx.doi.org/10.1039/C2LC20799K

[41] C. W. De Silva, *Vibration monitoring, testing, and instrumentation*. CRC Press, 2007.

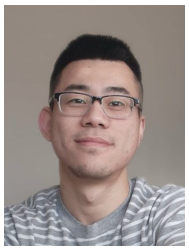
[42] A. E. Kennelly, "The equivalence of triangles and three-pointed stars in conducting networks," *Electrical World and Engineer*, vol. 34, no. 12, pp. 413–414, 1899.

[43] R. L. Graham, "Bounds for certain multiprocessing anomalies," *The Bell System Technical Journal*, vol. 45, no. 9, pp. 1563–1581, 1966.

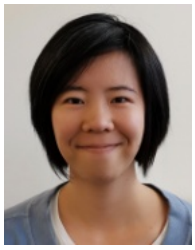
[44] Gurobi Optimization LLC, *Gurobi Optimizer Reference Manual*, 2022. [Online]. Available: https://www.gurobi.com

[45] M. Li, T.-M. Tseng, Y. Ma, T.-Y. Ho, and U. Schlichtmann, "Vom: Flow-path validation and control-sequence optimization for multilayered continuous-flow microfluidic biochips," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

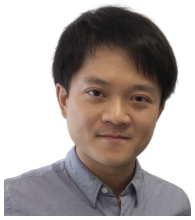
Meng Lian received a bachelor's degree in business mathematics and a master's degree in mathematics from Ludwig Maximilian University of Munich (LMU), Munich, Germany, in 2017 and 2020, respectively. She is currently a doctoral researcher with the Chair of Electronic Design Automation at Technical University of Munich (TUM). Her research interests cover design automation for electronic circuits and emerging technologies, including inkjet-printed electronics and continuous-flow microfluidics.



Shucheng Yang received his B.Sc. degree in Computer Science (Informatics) from the Technical University of Munich (TUM), Germany, in 2022. He is currently pursuing a Master's degree at the Chair of Data Engineering and Analytics within TUM's School of Computation, Information and Technology, where he previously served as a Research Assistant (Hiwi) at the Chair of Electronic Design Automation. His research interests include design automation for continuous-flow microfluidics, generative AI, multimodal reasoning, and language agents.



Mengchu Li received a B.A. degree in German Studies from Tongji University in China. She then came to Germany and received the B.Sc. and M.Sc. degrees in Computer Science from Ludwig Maximilian University of Munich, and the Dr.-Ing. degree from Technical University of Munich. She is currently a postdoctoral researcher in the Chair of Electronic Design Automation at the Technical University of Munich. Her research interests include design automation for continuous-flow microfluidics and optical networks-on-chips.



Tsun-Ming Tseng received a bachelor's degree in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2010, the M.Sc. and the Dr.-Ing. degrees from Technical University of Munich (TUM), Munich, Germany, in 2013 and 2017, respectively. He currently leads the Emerging Technology Group in the Chair of Electronic Design Automation, TUM. His research interests focus on design automation for emerging technologies, such as microfluidic biochips and optical networks-on-chips.

Ulf Schlichtmann received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering and information technology from Technical University of Munich (TUM), Munich, Germany, in 1990 and 1995, respectively. He was with Siemens AG, Munich, and Infineon Technologies AG, Munich, from 1994 to 2003, where he held various technical and management positions in design automation, design libraries, IP reuse, and product development. He has been a Professor and the Head of the Chair of Electronic Design Automation, TUM, since 2003, where he also served as Dean of the Department of Electrical and Computer Engineering, from 2008 to 2011, and as Associate Dean of Studies of International Studies from 2013 to 2022. Since 2022, he serves as vice-chairman of TUM's Academic Senate. He is a member of the German National Academy of Science and Engineering and also serves on a number of advisory boards. Ulf's current research interests include computer-aided design of electronic circuits and systems, with an emphasis on designing reliable and robust systems. In recent years, he has increasingly worked on emerging technologies, such as microfluidic biochips and optical interconnects.