

# Synthesis of a Cyberphysical Hybrid Microfluidic Platform for Single-Cell Analysis

Mohamed Ibrahim, *Student Member, IEEE*, Krishnendu Chakrabarty, *Fellow, IEEE*,  
and Ulf Schlichtmann, *Member, IEEE*

**Abstract**—Single-cell genomics is used to advance our understanding of diseases such as cancer. Microfluidic solutions have recently been developed to classify cell types or perform single-cell biochemical analysis on pre-isolated types of cells. However, new techniques are needed to efficiently classify cells and conduct biochemical experiments on multiple cell types concurrently. Nondeterministic cell-type identification, system integration, and design automation are major challenges in this context. To overcome these challenges, we present a hybrid microfluidic platform that enables complete single-cell analysis on a heterogeneous pool of cells. We combine this architecture with an associated design-automation and optimization framework, referred to as CoSynthesis (CoSyn). The proposed framework employs real-time resource allocation to coordinate the progression of concurrent cell analysis. Besides this framework, a probabilistic model based on a discrete-time Markov chain (DTMC) is also deployed to investigate protocol settings where experimental conditions, such as sonication time, vary probabilistically among cell types. Simulation results show that CoSyn efficiently utilizes platform resources and outperforms baseline techniques.

**Keywords**—Cyberphysical integration, design automation, graph search, hybrid system, Markov chains, microfluidics, synthesis, single-cell analysis.

## I. INTRODUCTION

Single-cell analysis using affordable microfluidic technologies has now become a reality [1], [2]. Thousands of heterogeneous cells can be explored in a high-throughput manner to investigate the link between gene expression and cell types, thereby providing insights into diseases such as cancer [3]. Microfluidic techniques have recently been developed to conduct each step of the following single-cell experimental flow. **(1) Cell Encapsulation and Differentiation:** Heterogeneous cells are isolated, encapsulated inside droplets, and differentiated according to their identity (type); e.g., their shape, size, cell-cycle stage, or lineage [4].

**(2) Droplet Indexing (Barcoding):** Each droplet is manipulated through a sequence of biochemical procedures such as cell lysis and mRNA analysis. At the end of these steps, the *in-situ* type of the encapsulated cell may no longer be available for down-stream analysis [5]. Therefore, indexing of droplets using barcodes is needed to keep track of their identity [6].

**(3) Type-Driven Cell Analysis:** Single-cell studies are increasingly being used to measure cell properties that are not directly

observable in a cell population. Single-cell bioassays such as chromatin immunoprecipitation (ChIP) are carried out using microfluidics, where the selection of a bioassay relies on the cell type that is identified in Step 1 [1]. To draw meaningful conclusions, the experimental outcomes are associated with droplet barcodes injected in Step 2 [7].

To tackle the myriad complexities associated with the above flow, microfluidics design-automation (“synthesis”) is essential. Independent multiple sample pathways need to be supported for concurrent manipulation of cells. Current synthesis techniques are not able to cross the formidable barrier that separates biochip design from practical single-cell studies. The following discussion highlights the main challenges in integrated single-cell studies:

**Heterogeneity of Single-Cell Methods:** Not all the above steps can be efficiently miniaturized using a single microfluidics technology. Valve-based techniques are used to rapidly separate and isolate biomolecules with high resolution, making them suitable for cell encapsulation (Step 1) [1]. On the other hand, digital-microfluidic biochips (DMFBs) enable real-time decision making for sample processing and genomic-analysis protocols, such as quantitative polymerase chain reaction (qPCR) [8] (Step 3). However, DMFBs are not as effective for interfacing to the external world [9]. Hence, there is a need for a hybrid microfluidic system that combines the advantages of the two domains, and a synthesis method that controls single-cell experiments in a dual-domain microfluidic setting. **Scalable Droplet Indexing:** A single-cell analysis flow may involve hundreds of cell types, each of which requires a distinct barcode for down-stream analysis using digital microfluidics. Therefore, droplet indexing on a DMFB requires either the use of pre-stored droplets that host individual barcoding hydrogels [7] (not feasible when a large number of cells are being investigated) or a specific input reservoir for each cell type. The latter solution increases the fabrication cost dramatically. Furthermore, since reservoir control is not readily automated [10], it is unrealistic to assume that each dispensed droplet contains only one barcoding particle.

**Dynamic synthesis:** Due to the inherent uncertainty about cell types, cyberphysical integration can play a key role in streamlining microfluidic cell-type identification and single-cell analysis. However, employing cyberphysical integration for processing every cell requires a dynamic synthesis capability, which can effectively explore resource space and also provide a prompt solution. As a result, the need for such a capability introduces a tradeoff between synthesis performance, e.g., protocol completion time, and system responsiveness—this tradeoff has yet to be investigated.

---

M. Ibrahim and K. Chakrabarty are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA (e-mail: {mohamed.s.ibrahim, krishnendu.chakrabarty}@duke.edu). Ulf Schlichtmann is with the Institute for Electronic Design Automation, Technical University of Munich, 80333 Munich, Germany (e-mail: ulf.schlichtmann@tum.de).

**Stochasticity of Protocol Conditions:** Despite the efforts made towards standardization of single-cell protocols, many protocol steps, such as the duration of sonication in ChIP, are deemed to be cell-type-specific. For example, CD4+ white-blood cells require significantly longer sonication time compared to other red blood cells [11]. Failing to characterize the variation in sonication time and other parameters among cell types may lead to degradation in down-stream immunoprecipitation and thus the overall ChIP performance [12]. Hence, it is important to take into consideration such stochasticity when defining the protocol guidelines.

In this paper, we address the above challenges by introducing the first hybrid microfluidic platform for integrated single-cell analysis. We present a synthesis method, referred to as Co-Synthesis (CoSyn), to control the dual-domain platform. We also develop a probabilistic model that employs a discrete-time Markov chain (DTMC) to capture protocol settings where experimental parameters vary among cell types. The main contributions of this paper are as follows:

- We present an architecture of a hybrid microfluidic platform that integrates digital-microfluidic and flow-based domains (using valves) for large-scale single-cell analysis.
- We describe CoSyn, which enables coordinated control of the microfluidic components, and allows dual-domain synthesis for concurrent sample pathways.
- We propose two schemes for valve-based routing (graph-theoretic and incremental methods), which enable dynamic routing of concurrent samples within a reconfigurable valve-based system.
- We construct a DTMC model, which utilizes probabilistic information related to protocol steps and experiment budget to investigate the efficiency of probabilistic protocol decisions.
- We evaluate system performance and reconfigurability while exploring various configurations of the valve-based system.

The rest of the paper is organized as follows. Section II presents an overview of related prior work and probabilistic formal methods that are relevant to this work. An overview of the hybrid microfluidic platform and its use for single-cell analysis are presented in Section III. Next, we formalize the single-cell analysis flow in Section IV and describe the proposed synthesis framework (CoSyn) in Section V. Subsequently, details of the valve-based synthesizer are introduced in Section VI, and probabilistic protocol modeling is presented in Section VII. Our experimental evaluation is presented in Section VIII and conclusions are drawn in Section IX.

## II. PRELIMINARIES

In this section, we review synthesis methods for microfluidic biochips and relevant modeling techniques related to stochastic processes.

### A. Synthesis of Microfluidic Platforms

A DMFB manipulates picoliter droplets, and consists of a two-dimensional array of electrodes and a set of on-chip resources [13]. Valve-based biochips, on the other hand, rely

on special-purpose components (e.g., microvalves and micro-pumps) to manipulate liquid flow [14].

Considerable research efforts have been devoted to the synthesis of biochemical applications for a specific microfluidic technology, e.g., either digital-microfluidic systems or valve-based systems. Early synthesis methods for both technologies focused on scheduling, droplet routing, chip-level routing, and sharing of control pins (or pressure sources) [15]–[26]. However, these methods are inadequate for single-cell analysis since they can only be applied to single biochemical assays. Moreover, real-time coordination between different single-cell microfluidic techniques is not possible using these early methods.

Cyberphysical synthesis techniques have enabled online error recovery [27]–[31], volume precision [32], termination of biochemical applications such as qPCR [33], and protocols for multiple samples [34], [35]. However, a key limitation of the above methods is that they fail to support heterogeneous single-cell analysis, and considerable manual effort is required to coordinate biochemical procedures.

Therefore, to close the gap between microfluidics and single-cell genomics, there is a need for a synthesis framework that can coordinate single-cell analysis techniques.

### B. Discrete-Time Markov Chains and Stochastic Systems

DTMCs constitute a formal method to model stochastic systems, such as in biology [36], that exhibit a discrete state space [37]. DTMCs are similar to finite-state machines, but enriched with probabilistic transitions to model random phenomena. A transition from state  $s_i$  to state  $s_j$  is associated with a one-step transition probability that must depend only on the two states and not on the previous transitions—this property is referred to as Markov property.

Formally, a DTMC  $\mathcal{D}$  is specified as a tuple  $\mathcal{D} = \langle \mathcal{S}, s_{init}, \mathcal{U}, \mathcal{J} \rangle$ , where: (i)  $\mathcal{S}$  is a finite set of states (state space) and  $s_{init} \in \mathcal{S}$  is the initial state; (ii)  $\mathcal{U} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$  is a transition probability matrix such that  $\sum_{s' \in \mathcal{S}} \mathcal{U}(s, s') = 1 \forall s \in \mathcal{S}$ ; (iii)  $\mathcal{J} : \mathcal{S} \rightarrow 2^{AP}$  is a labeling function that assigns each state to a set of atomic propositions from a denumerable proposition set  $AP$  [37]. These labels can be utilized by a probabilistic model-checking tool to express DTMC model properties with the help of temporal logic.

Properties of a DTMC model can be described using Probabilistic Computation Tree Logic (PCTL) [38]. PCTL formulae are interpreted over the DTMC states. For instance, consider a predicate  $\phi$  that is always satisfied in state  $s$  (i.e.,  $s \models \phi$ ), we can use the following PCTL formulae to describe path or numerical queries:

- 1)  $Q_1 := P_{\geq 0.95}[\diamond \phi]$ , which inquires if the modeled system can eventually reach a state that satisfies  $\phi$  with a probability that is greater than or equal to 0.95. The operator  $\diamond$  means “eventually”.
- 2)  $Q_2 := P_{max=?}[\diamond^{<10} \phi]$ , which seeks the maximum probability that the system will eventually reach a state  $s$  satisfying  $\phi$  in less than 10 steps.

The query  $Q_1$  is considered to be a reachability query, whereas  $Q_2$  is a numerical query. These queries and others can be used to investigate the evolution of a stochastic system that

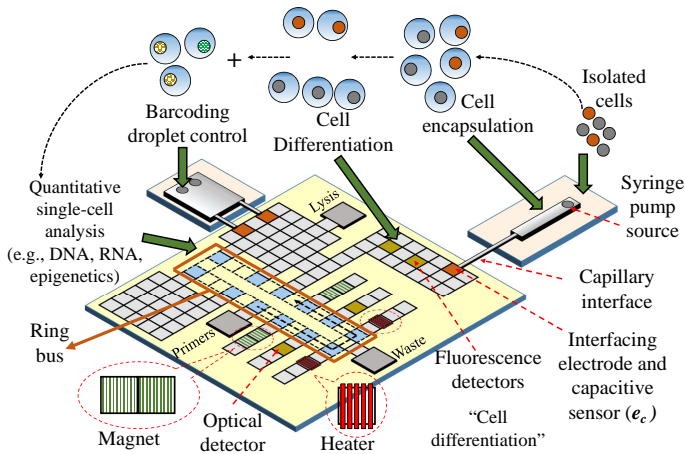


Fig. 1: The hybrid platform for single-cell analysis.

is modeled using DTMC. More details about DTMC model checking and PCTL syntax can be found in [38].

### III. HYBRID PLATFORM AND SINGLE-CELL ANALYSIS

Single-cell analysis relies on the concurrent manipulation of sample droplets, where each sample cell is run through the protocol flow discussed in Section I. An efficient on-chip implementation of the single-cell analysis protocol is accomplished using a hybrid platform. Fig. 1 shows the platform components matched with different protocol stages. The two domains are connected through a capillary interface; this technique has been successfully adopted in practice [10].

#### A. Cell Encapsulation and Flow Control

As shown in Fig. 1, on-chip operation starts with the encapsulation of single cells in droplets, which is efficiently accomplished using flow-based microfluidics [10]. The droplet generator uses a syringe pump such that the flow rate of pressure-driven droplets can be automatically controlled via feedback. A capacitive sensor is placed at the interfacing electrode ( $e_c$  in Fig. 1) on the digital side to sense a droplet [39]. When the digital array is unable to accommodate additional droplets, it stops the flow by switching off the pump.

Note that an actuator is used in the flow-based component, whereas a sensor is placed on the digital side. To synchronize the two domains, the flow-control procedure (capacitive sensing and pump control) is invoked at the same frequency as droplet actuation in the digital domain (1 Hz to 10 Hz).

#### B. Cell Differentiation

Automated cell-type identification can be achieved by analyzing signaling events in single cells *in situ*. Similar to the miniaturization of gene-expression analysis [40], a green fluorescent protein (GFP) reporter is used for cell differentiation. In each cell, the fluorescence intensity from the GFP (detected in real-time using an on-chip fluorescence detector or imaging apparatus) is used to account for differences in expression level among cells; this is equivalent to classifying cells into functional clusters that represent cell types.

Although a valve-based biochip can also be used for cell differentiation, we consider a DMFB for this purpose due to its

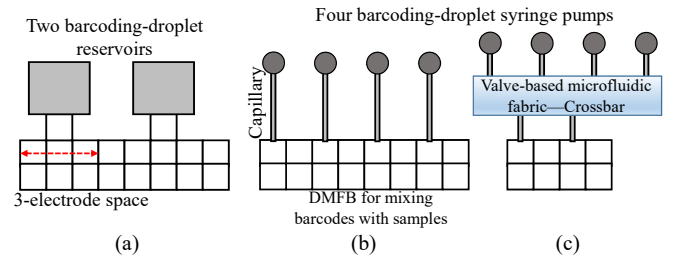


Fig. 2: Droplet barcoding using: (a) a DMFB; (b) a valve-based biochip with one-to-one mapping between syringe pumps and DMFB ports; (c) a valve-based biochip with many-to-many mapping.

demonstrated ability to carry out high-throughput fluorescence detection and distinguish between hundreds of cell types; this feature is not supported by valve-based mechanisms.

#### C. Droplet Barcoding

Since thousands of cells (and hundreds of cell types) can be involved in an analysis protocol, a barcoding droplet must be dispensed *on demand*, and mixed with a sample droplet and other reagents according to the cell type [7]. If we consider a population with  $n$  cell types, droplet barcoding on a digital-microfluidic array requires  $n$  reservoirs, each of which typically covers a 3-electrode space. An additional electrode is needed for separation; see Fig. 2(a). In this case, to accommodate  $n$  reservoirs, a lower bound on the array perimeter is  $4n + k$  electrodes, where  $k$  is a constant that represents the number of electrodes covered by other reservoirs. This approach is therefore impractical because of the significant increase in chip size with the number of target cell types. It also requires the dispensing of a single hydrogel particle per droplet; this feature has not been implemented yet using reservoir control.

To overcome the above limitations, a valve-based biochip is connected to the DMFB to exploit its pressure-driven ports that have smaller footprints than reservoirs; see Fig. 2(b).

This biochip is used to generate barcoding droplets via a syringe pump; the droplets are routed to appropriate locations on the digital-microfluidic array through a capillary interface [7]. With this hybrid configuration, the lower bound on the digital-array perimeter is reduced to  $2n + k$  electrodes for  $n$  cell types. A one-electrode gap is needed to prevent accidental mixing of droplets. This hybrid configuration, however, overprovisions the number of concurrently utilized ports; it is unlikely that all cell types will simultaneously request barcoding.

As a cost-effective solution, we utilize a reconfigurable valve-based fabric [41]; see Fig. 2(c). This routing fabric acts as a crossbar since it allows routing of barcoding droplets from any of the  $n$  input ports to any of the  $m$  output ports, where  $n \gg m$ . The full connectivity offered by this crossbar allows droplets to be easily re-routed and therefore it intrinsically supports fault tolerance. The  $m$ -output valve-based fabric is then stitched to the DMFB; hence the lower bound on the perimeter is decreased to  $2m + k$ . However, in this situation,  $p$  valves ( $p > 0$ ) are necessary for routing barcoding droplets across the fabric. By unlocking this capability of valve-based

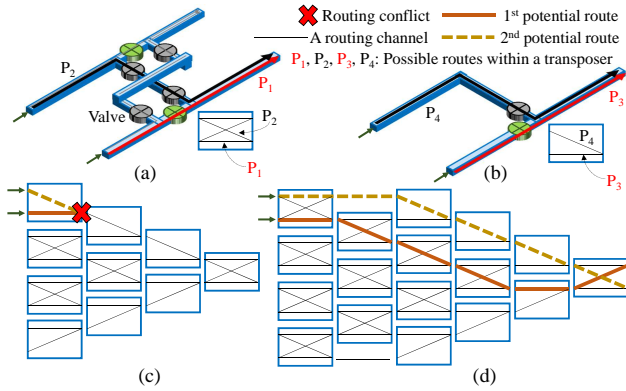


Fig. 3: A valve-based routing fabric for droplet barcoding: (a) a 2-input full transposer; (b) a 2-input half transposer; (c) a 4-level, 8-to-2 routing fabric; (d) a 6-level, 8-to-2 fabric.

TABLE I: Design problems for assembling and integrating an  $n$ -to- $m$  valve-based crossbar.

Problem	Objective
Architecture	Analyze the tradeoff between routing flexibility and operation timing and cost to obtain the best transposer configuration.
Modeling	Map the architecture into algorithmic semantics that support real-time routing.
Synthesis	Solve the valve-based routing problem considering real-time reconfiguration and in coordination with resource allocation in DMFBs.

crossbars, we shift the scaling complexity from the digital domain to the flow-based domain, which is known to have a cost-effective fabrication process and efficient peripheral components.

We utilize the “transposer” primitive introduced in [41]. As shown in Fig. 3(a)-(b), a valve-based transposer appears in two forms: (1) a two-input, two-output transposer, which is comprised of six valves, controlled via two pneumatic inputs (full transposer); (2) a two-input, one-output transposer, which consists of two valves controlled via two pneumatic inputs (half transposer). Note that only a full transposer allows simultaneous dispensing of two barcoding droplets, wherein the droplets can be driven “straight” or “crossed”.

The use of transposers to construct an  $n$ -to- $m$  valve-based crossbar leads to various design problems that must be tackled; see Table I. An architectural design challenge arises because various configurations of transposers can be exploited to achieve the required number of input and output ports. For example, an 8-to-2 crossbar can be constructed using four “vertical” levels, as shown in Fig. 3(c), or using six levels, as shown in Fig. 3(d). A six-level crossbar, while incurring higher cost, provides a higher degree of reconfigurability and flexibility in routing. In Section VI-C, we study valve-based routing using various configurations of the crossbar.

#### D. Type-Driven Single-Cell Protocol

After a droplet is barcoded, a single-cell analysis protocol is applied to the constituent cell in the DMFB, where protocol specifications are determined based on the cell type. For example, an investigator might be interested in identifying

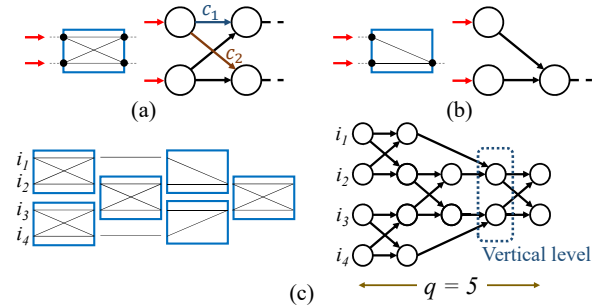


Fig. 4: Mapping a valve-based crossbar to a graph model: (a) a full transposer; (b) a half transposer; (c) a 4-to-2 crossbar.

the gene expression of a specific genomic loci for a certain cell type  $A$ . Another cell type  $B$  might show unexpected heterochromatic state at a certain loci, and the investigator might be interested in identifying the protein interactions (i.e., causative proteins) or chromatin modifications causing this behavior. For type  $A$ , it is sufficient to perform gene-expression analysis using qPCR [8], [40], whereas ChIP protocol followed by qPCR must be used for type  $B$  to reveal the DNA strains contributing in the activity of the causative proteins.

## IV. MAPPING TO ALGORITHMIC MODELS

### A. Modeling of a Valve-Based Crossbar

We represent the set of transposers and their interconnections as a directed acyclic graph (DAG)  $\mathcal{T} = (\mathcal{X}, \mathcal{Z})$ , where a vertex  $x_i \in \mathcal{X}$  is a transposer node, and an edge  $z_i \in \mathcal{Z}$  represents a connection between two transposers. Within a transposer, the point at which a droplet can be routed either straight or crossed is defined as a decision point. We map an  $n$ -to- $m$  valve-based crossbar (with a transposer network  $\mathcal{T}$ ) into a DAG  $\mathcal{F}_{n \times m} = (\mathcal{D}_{n \times m}, \mathcal{S}_{n \times m})$ , where a vertex  $d_i \in \mathcal{D}_{n \times m}$  is a flow-decision node, and an edge  $s_i \in \mathcal{S}_{n \times m}$  represents a channel that connects two decision nodes. To simplify the discussion, we do not include  $\mathcal{T}$  in the notation for the crossbar DAG. We can view a full (half) transposer as a 2-to-2 (2-to-1) valve-based crossbar; thus, we represent fluid-flow control in a full (half) transposer as a DAG  $\mathcal{F}_{2 \times 2}$  ( $\mathcal{F}_{2 \times 1}$ ); see Fig. 4(a)-(b). The cost  $c_i$  of  $s_i$  represents the time needed to transport fluid between the two connected nodes, measured in flow time steps ( $T_f$ ). We assume that the routing time of a droplet on a straight channel between two decision nodes is a unit of  $T_f$ . For example, as shown in Fig. 4(a),  $c_1$  is equal to  $T_f$ , whereas  $c_2$  is equal to  $2 T_f$ , since even though a diagonal is shown in Fig. 3 as a fluidic path, routing of such paths in a transposer is implemented only along the  $x$ - and  $y$ -directions and the distances along these dimensions are equal [41]. Fig. 4(c) depicts the graph  $\mathcal{F}_{4 \times 2}$  for a 4-to-2 crossbar with 4 levels of transposers and 5 levels of nodes (denoted henceforth by  $q$ ;  $q = 5$  in this case).

### B. Modeling of a Digital-Microfluidic Biochip

While DMFBs are highly reconfigurable and can support a diverse set of transport paths, we reduce the burden of managing droplet transport in real-time by considering a unidirectional ring-based architecture, as shown in Fig. 1. Connected

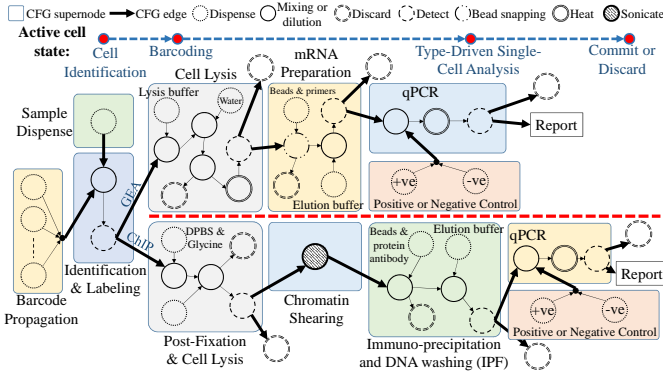


Fig. 5: CFG of type-driven analysis for a single cell pathway.

to this ring are on-chip resources. Since there is always a route between any pair of on-chip resources, a ring-based DMFB is modeled as a strongly connected DAG  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where a vertex  $v_i \in \mathcal{V}$  represents the fluid-handling operation offered by an on-chip resource, and a directed edge  $e_i \in \mathcal{E}$  represents a path (over the ring) that connects two resources. The cost  $ce_i$  of  $e_i$  indicates the number of digital time steps ( $T_d$ ) needed to transport a droplet. We assume that the durations corresponding to  $T_f$  and  $T_d$  are equal, which can be achieved in practice by tuning the actuation frequency (Hz) and the flow rate (mL/min).

### C. Protocol Model and Cell State Machine

To solve the synthesis problem for single-cell analysis, we take into account the complexity imposed by the barcoding mechanism. Similar to the design methodology in [40], we represent the protocol as a control flow graph (CFG)  $\mathcal{A} = (\mathcal{H}, \mathcal{L})$ , in which every node  $h_i \in \mathcal{H}$  (referred to as a *supernode*) models a bioassay such as qPCR; see Fig. 5. A directed edge  $l_i \in \mathcal{L}$  linking two supernodes  $\{h_j, h_k\}$  indicates that a potential decision can be made at runtime to direct the protocol flow to execute the bioassay  $h_k$  after  $h_j$ . A supernode  $h_i$ , in turn, encapsulates the sequencing graph that describes the fluid-handling operations of a bioassay and the interdependencies among them. Since there is inherent uncertainty about the type of barcoding droplets for a sample cell at design time, we extend the basic CFG model by incorporating an internal supernode (barcode propagation) that describes all possible dispensing options of barcoding droplets. Note that this model is agnostic about the type of the microfluidic technology used for implementing the protocol. Yet, the synthesis of each supernode is accomplished in a technology-aware manner using CoSyn.

In addition to the CFG model, a state machine is utilized to model the progression of each cell along the single-cell pipeline. Typically, the hybrid platform can iteratively process thousands of cells; such cells might be scattered across the platform domains at any given point in time. Therefore, this state machine (Fig. 5) is necessary to keep track of the cells that are being processed simultaneously.

## V. CO-SYNTHESIS METHODOLOGY

In this section, we describe the synthesis problem and present an overview of the proposed co-synthesis methodology.

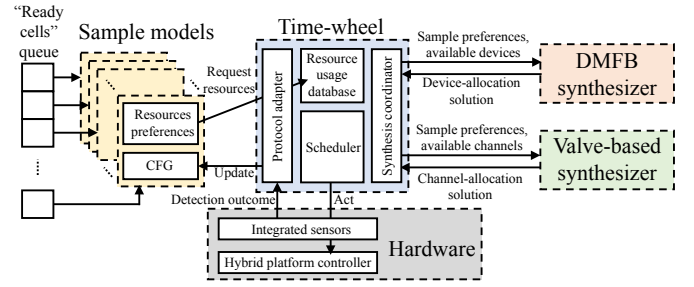


Fig. 6: Overview of the proposed co-synthesis methodology.

### A. Problem Formulation

Our optimization problem is as follows:

**Inputs:** (i) The protocol CFG  $\mathcal{A}$ . (ii) A matrix  $C$ ; each vector  $C_i \in C$  corresponds to a cell, and consists of integers that encode cell state machine, cell type, and the assigned bioassays in  $\mathcal{A}$ . (iii) The configuration of the valve-based system; this information includes the graph  $\mathcal{F}_{n \times m}$ , the number of inputs  $n$ , and the number of outputs  $m$ . (iv) The types of resources corresponding to the DMFB, their operation time, and the routing distance between each pair of resources.

**Output:** Allocation of chip modules to the individual cells, protocol completion time  $T_{comp}$ , and average response (computational) time  $T_{resp}$  of the cyberphysical system.

**Objective:** Minimize  $T_{comp}$  to provide high throughput and minimize  $T_{resp}$  to improve system responsiveness.

### B. Proposed Solution

The proposed co-synthesis scheme, depicted in Fig. 6, consists of four components: (1) valve-based synthesizer, which is used to route barcoding droplets through the valve-based crossbar; (2) DMFB synthesizer, which is utilized for allocating DMFB resources, e.g., mixers and heaters, to sample pathways; (3) biology-sample model, which records the progress of a sample (cell) within the protocol CFG and also provides updated resource preferences; (4) time-wheel engine, which seamlessly coordinates real-time interactions between the individual sample models and the synthesizers of the hybrid system. Note that the stages of the single-cell pipeline, simulated by this scheme, match the states of the cell state machine (Fig. 5).

The time-wheel interacts with other components through APIs. For example, whenever the time-wheel locates an available fluorescence detector at the DMFB, a new sample model is initialized (Fig. 6) and the associated cell is allocated to the detector in order to perform type identification. Next, when the cell type is identified and there are available valve-based routes to route the associated barcoding droplet, the time-wheel triggers the valve-based synthesizer to start the pipelined routing process of the barcoding droplet; valve-based routing is performed through iterations until the droplet reaches the electrode interface at the digital-microfluidic side and mixed with the cell. We discuss two methods for valve-based routing in Section VI.

When a DMFB resource is available to further process the cell, the previously reserved valve-based channels are released by the time-wheel. Hence, real-time resource allocation for

---

**Algorithm 1 DMFB Resource Allocation**


---

**Input:**  $C_i, \mathcal{G}$ , current simulation time “ $t$ ”

**Output:** Assigned Resource “ $r$ ”

- 1:  $\tilde{R} \leftarrow \text{GetCurrentlyUnoccupiedResources}(\mathcal{G}, t)$ ;
  - 2: **if** ( $\tilde{R}$  is empty) **then return** NULL;
  - 3:  $y \leftarrow \text{GetOperationType}(C_i)$ ;
  - 4:  $s \leftarrow \text{CalculateMinimumCostAllAvailableResources}(y, \tilde{R})$ ;
  - 5: **if** ( $s = \infty$ ) **then return** NULL; // No suitable resource
  - 6:  $r \leftarrow \text{GetSelectedResource}(s)$ ; **return**  $r$ ;
- 

the DMFB is also initiated by the time-wheel, which in turn, commits a cell pathway whenever its particular single-cell bioassays have executed. Based on an intermediate decision point whose outcome is communicated to the sample model, the cell might also be discarded during analysis.

### C. DMFB Synthesizer

We use a greedy method to solve the resource-allocation problem in the DMFB; the pseudocode is shown in Algorithm 1. We denote a DMFB resource by  $r \in R^d$ , where  $R^d$  encapsulates all DMFB resources. Thus, the cost of allocating resource  $r$  to execute a fluidic operation of type  $y \in \mathcal{Y}$  (the set  $\mathcal{Y}$  incorporates all operation types) is  $\rho(\hat{r}, r, y) = \gamma(r, y) + \mathcal{E}(\hat{r}, r)$ , where  $\gamma(r, y)$  is the operation time on  $r$  and  $\mathcal{E}(\hat{r}, r)$  is the routing distance from  $\hat{r}$  (the currently occupied resource) to  $r$ . The worst-case computational complexity of this algorithm is  $O(|\mathcal{V}|)$ .

## VI. VALVE-BASED SYNTHESIZER

Our goal is to design a fully connected fabric such that a droplet can be forwarded from any of the  $n$  inputs to any of the  $m$  output ports. We present a sufficient criterion for achieving a fully connected fabric. The proof can be found in Appendix A.

**Theorem 1.** *An  $n$ -to- $m$ ,  $q$ -level valve-based crossbar is a fully connected fabric if  $n$  and  $m$  are even integers, and  $q \geq \frac{m+n}{2}$ .*

Using this theorem, we can automatically generate the graph model  $\mathcal{F}_{n \times m}$ , thereby guaranteeing that any barcoding input can reach all  $m$  outputs. The algorithm is described in Appendix B.

An  $n$ -to- $m$  valve-based crossbar allows only  $m$  barcoding droplets to be delivered simultaneously to the DMFB. We increase throughput by allowing pipelined routing of droplets. With pipelining, the routing algorithm allows a droplet to be routed even though a complete path to an output is unavailable. In this case, a droplet is immobilized at the furthest intermediate decision node that is not reserved by other droplets (a pipeline stage), then allowed to move forward when a path is freed. Fig. 7 illustrates pipelined and non-pipelined routing.

In the following subsections, we introduce two schemes for solving the routing problem. The first scheme in Section VI-A uses graph search to efficiently route barcoding droplets, whereas the second scheme in Section VI-B aims to improve system responsiveness (i.e., reduce computation time) via an incremental routing procedure.

### A. Method 1: Graph-Theoretic Routing

We utilize a graph-theoretic algorithm to find vertex-disjoint shortest paths [42]; see Algorithm 2. By computing disjoint

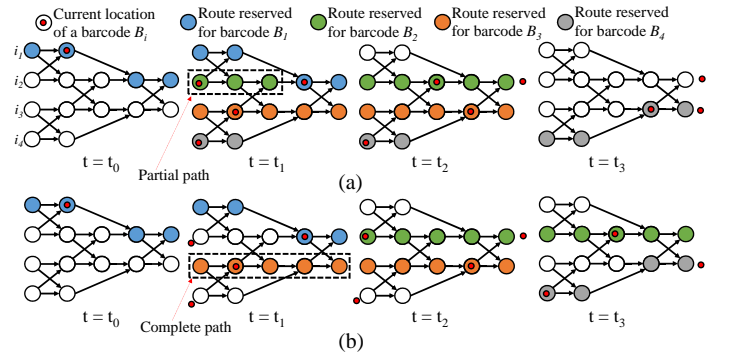


Fig. 7: Valve-based routing of 4 barcoding droplets (4-to-2 biochip): (a) with pipelining; (b) without pipelining.

---

**Algorithm 2 Pipelined Valve-Based Routing**


---

**Input:**  $C_i, \mathcal{F}_{n \times m}$ , current simulation time “ $t$ ”

**Output:** Generated routing path “ $P$ ”, is  $P$  complete “ $\theta$ ”

- 1:  $\mathcal{U} \leftarrow \text{GetCurrentlyUnoccupiedSubGraph}(\mathcal{F}_{n \times m}, t)$ ;
  - 2:  $d \leftarrow \text{GetVertexCurrentlyHoldingBarcode}(C_i, \mathcal{F}_{n \times m})$ ;
  - 3:  $P \leftarrow \text{GenerateVertexDisjointShortestPath}(d, \mathcal{U})$ ;
  - 4: **if** ( $P$  is empty) **then return** {NULL, false};
  - 5: **else if** ( $P$  is complete) **then return** { $P$ , true};
  - 6: **else return** { $P$ , false};
- 

paths, we ensure that different barcoding droplets do not interfere with each other during routing. The routing algorithm is invoked whenever a cell transitions from the identification state to the barcoding state. If all the  $m$  outputs of the chip are currently reserved, the algorithm generates a partially disjoint shortest-length path from the input source to the furthest node (Fig. 7). This is equivalent to routing the associated barcoding droplet up to an intermediate point, and holding the droplet until another disjoint path (partial or complete) can be computed to advance the droplet. The channels currently reserved for routing a barcoding droplet cannot be accessed by any other droplet until the droplet being held moves out of the valve-based crossbar.

Since the vertices of  $\mathcal{F}_{n \times m}$  are generated in topological order, the shortest paths can be quickly computed; the worst-case complexity of this algorithm is  $O(|\mathcal{D}_{n \times m}| + |\mathcal{S}_{n \times m}|)$ .

### B. Method 2: Incremental Routing

Method 1 is computationally expensive because it performs exhaustive search to find a vertex-disjoint shortest-length path whenever a barcoding droplet is allowed to move forward. Therefore, to reduce the computation time, we need to limit the size of the search space so that the routing of a droplet can be determined quickly. For this purpose, we replace the graph-theoretic method (Line 3 in Algorithm 2) with an incremental routing procedure, which computes the route of a droplet only for the next time step. Hence, by using this approach, the search space for routing a droplet is limited to three choices only: move straight, move crossed, or stay immobilized. The reduction in the search space can potentially reduce the overall computation time, although this procedure will be executed more often compared to the graph-theoretic method.

To facilitate the making of a routing decision, the valve-based synthesizer adopts the following priority scheme: (1) a

barcoding droplet must stay immobilized if both the straight and crossed channels are occupied; (2) a droplet must move straight (crossed) if only the straight (crossed) channel is unoccupied; (3) a droplet must move straight if both the straight and crossed channels are unoccupied. Fig. 8 illustrates incremental routing using a 4-to-2 crossbar. The computational complexity of this procedure is  $O(1)$ .

### C. Droplet Routing Using Variants of Crossbar Architecture

Our discussion so far has focused on valve-based routing using a fully connected crossbar  $\mathcal{F}_{n \times m}$ , which exhibits the highest degree of routing flexibility and fault tolerance. According to Theorem 1, this design requires at least  $\frac{n+m}{2}$  vertical levels. A drawback of using this number of vertical levels is that it increases the number of time steps needed to transport a barcoding droplet from an inlet to the DMFB side, which in turn may increase the total completion time. Therefore, we need to investigate the tradeoff between routing flexibility (or fault tolerance) and system performance.

To perform this study, we explore various configurations of an  $n$ -to- $m$  crossbar; these configurations differ in the number of vertical levels and therefore channel connectivity. For example, a 16-to-8 crossbar can be constructed using one of five different configurations, as shown in Fig. 9. It is obvious that the configuration in Fig. 9(a) provides the highest connectivity, but it uses the largest number of vertical levels (12 levels). In contrast, the configuration in Fig. 9(e) offers the lowest connectivity, but it utilizes the smallest number of vertical levels (3 levels).

We observe that a crossbar configuration can be constructed hierarchically using a set of fully connected crossbars. For instance, a 16-to-8 configuration can be constructed using a single 8-to-4 crossbar  $\mathcal{F}_{8 \times 4}$  and two 4-to-2 crossbars  $\mathcal{F}_{4 \times 2}$  (Fig. 9(c)), or it can be designed using a single 16-to-8 fully connected crossbar  $\mathcal{F}_{16 \times 8}$  (Fig. 9(a)). Clearly, the hierarchical design approach not only facilitates the control of the crossbar connectivity, but it also allows us to reuse the definitions and routing methods proposed earlier for fully connected fabrics.

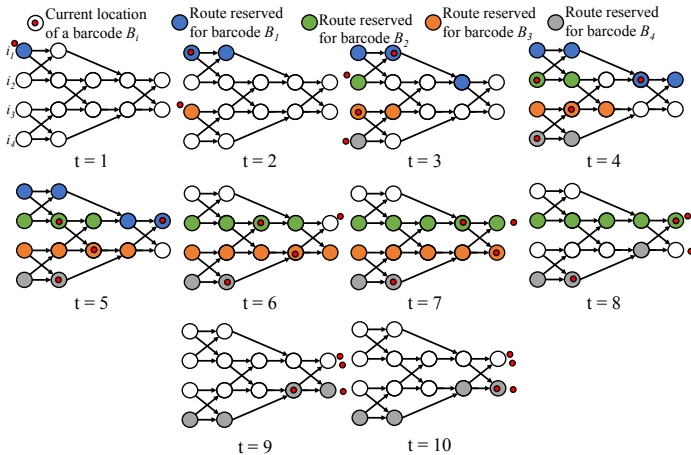


Fig. 8: Incremental routing of 4 barcoding droplets (4-to-2 biochip).

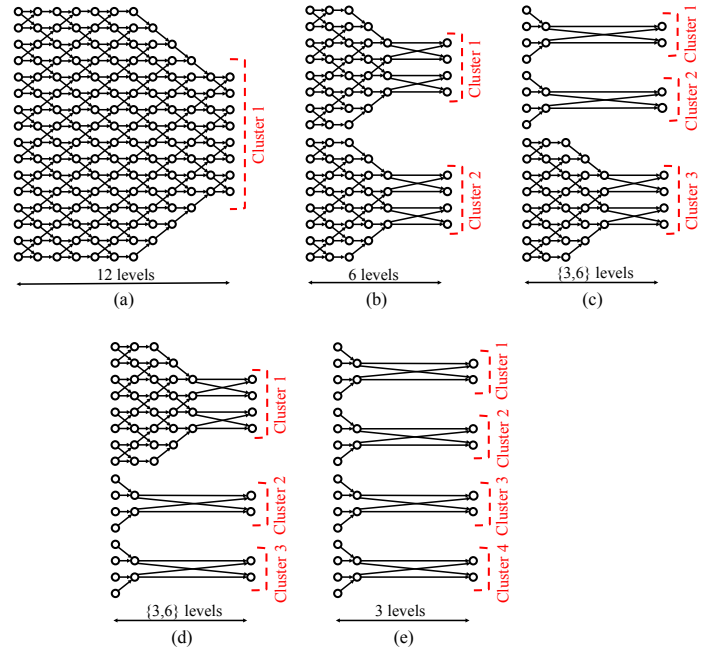


Fig. 9: Architectural variants of a 16-to-8 crossbar: (a)  $\mathcal{F}_{16 \times 8}^1$ ; (b)  $\mathcal{F}_{16 \times 8}^2$ ; (c)  $\mathcal{F}_{16 \times 8}^3$ ; (d)  $\mathcal{F}_{16 \times 8}^3$ ; (e)  $\mathcal{F}_{16 \times 8}^4$ .

A formal discussion of a crossbar configuration can be established by partitioning the crossbar outputs into clusters. The output ports within a single cluster are accessible to the same set of input ports, i.e., a cluster forms a fully connected crossbar. On the other hand, the output ports that belong to two different clusters cannot be accessed from the same input. By using this characterization mechanism, we can define a crossbar configuration based on the number of output clusters  $cl$ . For example, the configuration in Fig. 9(a) has a single cluster ( $cl = 1$ ), the configuration in Fig. 9(b) has two clusters ( $cl = 2$ ), the configurations in Fig. 9(c-d) have three clusters ( $cl = 3$ ), and the configuration in Fig. 9(e) has four clusters ( $cl = 4$ ). Note that both designs in Fig. 9(c-d) are treated similarly, since they exhibit the same number of vertical levels. Hence, we denote a crossbar configuration by  $\mathcal{F}_{n \times m}^{cl}$ , where  $n$  and  $m$  are the number of inputs and outputs, respectively. Also, note that we can use  $\mathcal{F}_{n \times m}^1$  and  $\mathcal{F}_{n \times m}$  interchangeably since  $\mathcal{F}_{n \times m}^1$  is the only crossbar variant that is fully connected.

The proposed crossbar configurations can be employed to route droplets using our methods from the previous subsections. Our algorithms will automatically comply with the clustering constraints imposed by the new crossbar connectivity, since these constraints are captured by the graph model. In Section VIII, we study the impact of crossbar configurations on single-cell performance via simulations.

## VII. PROTOCOL MODELING USING MARKOV CHAINS

In this section, we present a probabilistic scheme to address the stochasticity of protocol conditions. The outcome of this step is used as an input to CoSyn.

TABLE II: Experimental data describing protocol-condition space for ChIP using yeast cells.

Case number	# Replicates	Fixation time (min)	Incubation Temp. (°C)	# Washes	IP ( $\mu, \sigma$ )
1	32	10	25	4	(16.4,3.4)
2	6	10	18	4	(18.2,5.1)
3	5	10	32	4	(15.8,3.7)
4	4	8	25	4	(13.2,2.9)
5	2	8	18	4	(16.9,6.8)
6	3	10	32	4	(22.1,5.6)
7	4	15	25	4	(18.7,4.3)
8	3	15	18	4	(19.1,3.6)
9	3	15	32	4	(16.8,6.2)
10	5	10	25	8	(16.7,4.1)
11	3	5	25	4	(12.2,3.1)
12	3	30	25	4	(15.4,4.7)

### A. Probabilistic Modeling Approach

While the graph model described in Section IV can effectively support type-driven single-cell execution, it suffers from the following drawbacks: (1) it assumes that the protocol conditions (e.g., fixation time and incubation temperature) are insensitive to cell types and consequently they do not have an impact on the protocol efficiency; (2) it also assumes that the optimal settings of these conditions can be uniquely defined. These assumptions however may not be valid in many real-life scenarios, particularly due to the inherent stochasticity in cellular interactions.

To analyze the stochastic behavior of single-cell protocols, we collected experimental data for a population of yeast cells after conducting several benchtop implementations of the ChIP protocol. The collected data (Table II) shows the normalized immunoprecipitation (IP) value, i.e., protocol efficiency, while varying the experimental conditions; this table is referred to as *protocol-condition space*. By analyzing the obtained data, we observe that changing the protocol settings leads to different IP outcomes; the settings shown in Case #6 provide the highest efficiency. In addition, despite the use of biological replicates in each case, the results obtained by these replicates are not identical and they follow a normal distribution  $\mathcal{N}(\mu, \sigma^2)$ . Hence, according to this analysis, it is necessary to redesign the protocol model to address the above challenges.

An effective solution to this problem is based on a probabilistic approach. The steps of this approach are given below: (1) For each cell type, we collect data from previous experiments and construct the associated protocol-condition space, similar to the example in Table II.

(2) We specify an arbitrary lower-bound threshold  $\Gamma$  for the protocol efficiency. For example, we choose  $\Gamma = 16.5$  for the example in Table II. Based on this threshold, we classify cell population (replicates) into two sets: a set of cells that contributes to high efficiency, denoted by ( $\mathcal{HE}$ ), and another set of cells that contributes to low efficiency, denoted by ( $\mathcal{LE}$ ).

(3) Let  $A$  be an event that an arbitrary input cell belongs to  $\mathcal{HE}$ . We compute the probability that the input cell contributes to high efficiency, i.e., belongs to  $\mathcal{HE}$ , as follows:  $P(A) = \frac{|\mathcal{HE}|}{|\mathcal{HE}|+|\mathcal{LE}|}$ . Based on Table II,  $P(A) = \frac{26}{73} = 0.36$ .

(4) Consider a certain protocol condition such as the fixation time. We extract all possible settings  $S_i$  of this condition from the table. According to Table II and based on the fixation time,

$S_1$ : 5-min,  $S_2$ : 8-min,  $S_3$ : 10-min,  $S_4$ : 15-min, and  $S_5$ : 30-min. (5) Let  $B_i$  be an event that an arbitrary cell is processed using the setting  $S_i$ . We compute the probability that an arbitrary cell will be processed using the setting  $S_i$  as follows:  $P(B_i) = \frac{|S_i|}{\sum_i |S_i|}$ , where  $|S_i|$  represents the number of replicates processed using  $S_i$ . By considering ( $S_3$ : 10-min) from Table II, we obtain  $P(B_3) = \frac{51}{73} = 0.7$ .

(6) We next compute the conditional probability  $P(B_i|A) = \frac{P(B_i \cap A)}{P(A)}$ , which represents the probability that an arbitrary cell will be processed using  $S_i$  given that this cell belongs to  $\mathcal{HE}$ . The probability  $P(B_i \cap A)$  represents the percentage of cell population that satisfy the following conditions: *i* the cells are processed using the setting  $S_i$ ; *ii* the cell belong to  $\mathcal{HE}$ . In the example shown in Table II,  $P(B_3 \cap A) = \frac{14}{73} = 0.2$ , and therefore  $P(B_3|A) = 0.53$ .

(7) We apply Bayesian inference to estimate the posterior probability  $P(A|B_i)$ , which indicates the probability that an arbitrary cell contributes to high performance given that this cell is processed using the setting  $S_i$ —Bayes’ theorem mathematically states that  $P(A|B_i) = \frac{P(B_i|A) \cdot P(A)}{P(B_i)}$ . According to the example in Table II,  $P(A|B_3) = \frac{0.53 \times 0.36}{0.7} = 0.27$ . We can also compute  $P(A|B_1)$ ,  $P(A|B_3)$ , and the other posterior probabilities similarly.

(8) We normalize  $P(A|B_i)$  using the following relation:  $\hat{P}(A|B_i) = \frac{P(A|B_i)}{\sum_i P(A|B_i)}$ .

By adopting the above systematic approach, we are able to capture the inherent stochasticity in the protocol environment. This approach therefore allows us to redefine the protocol interactions via a Markov-chain model.

### B. Modeling Using Discrete-Time Markov Chains

Consider a given sequence of cells, where each cell belongs to one of  $\tau$  cell types. We consider that cell-type selection follows a uniform distribution  $P(X)$ ;  $X$  is a random variable associated with the percentage of the cell-population count for each type. Based on the probabilistic approach discussed earlier, each cell type is associated with specific experimental steps for single-cell analysis and unique probabilistic distributions for selecting protocol settings in each step. We also take into account cost limitations of the overall experimental environment such as the maximum quantities of master mixes and washing liquids. Such cost limitations (also known as experiment budget) play a key role in tuning the overall performance of the protocol.

Based on the above characteristics, our objective is defined as follows: *Given a certain experiment budget, find the protocol strategy for a given sequence of cells that maximizes the probability that the cells contribute to high efficiency.* To achieve this goal, we model the execution of single-cell analysis as a DTMC and solve the strategy synthesis problem via probabilistic model checking. The outcome of strategy synthesis can then be fed into CoSyn for resource allocation. In the future, we plan to integrate both strategy synthesis and resource allocation (CoSyn) into a single unified framework.

Fig. 10 shows a DTMC model for single-cell analysis for  $\tau = 3$ . The model constitutes a finite set of states  $\mathcal{S}$ , which represents a sequence of biochemical operations (e.g., *mRNA*

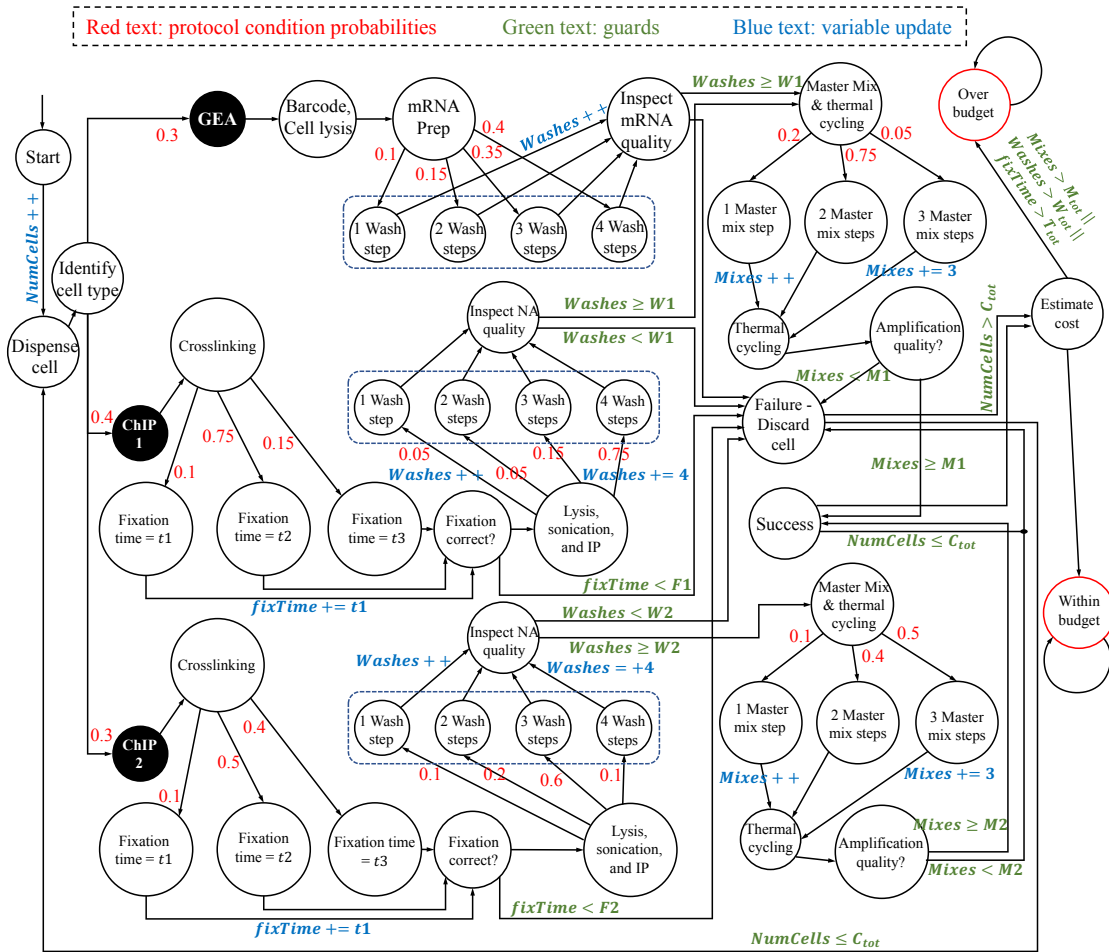


Fig. 10: A DTMC model for a single-cell protocol that considers three cell types.

*Prep*) or analysis procedures (e.g., *Estimate cost*). In addition to the set  $\mathcal{S}$ , we define a transition probability matrix  $\mathcal{U} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$  that represents decisions related to the protocol conditions and the associated probabilities; these probabilities are highlighted in red in Fig. 10. Recall that we compute these probabilities using Bayesian inference as discussed earlier.

To model the experiment budget constraints (and other constraints), we augment the state transitions of the proposed DTMC model with a set of *guards*, which ensure the fulfillment of these constraints and therefore transition the system towards success (*Within budget*) or failure (*Over budget*). For example, in Fig. 10, the transition from state “*Success*” to state “*Dispense cell*” is possible only if the number of processed cells is still below the total population count  $C_{tot}$ . Clearly, these guards, which are tuned based on the experiment budget, play a key role in computing the optimal protocol strategy. In Section VIII-E, we examine the budget’s role in controlling the protocol efficiency using PCTL.

## VIII. SIMULATION RESULTS

We implemented CoSyn using C++. All evaluations were carried out using a 2.7 GHz Intel i5 CPU with 8 GB RAM. The set of bioassays constituting the single-cell analysis protocol

(Section III) were used as a benchmark. Cell types were assigned to the cells using a uniform distribution function.

Since this is the first work on synthesis for hybrid microfluidic platforms, we have developed two baseline frameworks: (1) architectural baseline (ArcSyn), wherein the barcoding fabric is valveless and it utilizes a one-to-one mapping between syringe pumps and DMFB ports as in Fig. 2(b); (2) algorithmic baseline (ReSyn), in which resource allocation is initially performed for each microfluidic domain separately. However, we must ensure that the system behavior at the boundary between the two domains is deterministic—the synthesis tool for the DMFB must be aware of the order of the barcoding droplets generated from the valve-based crossbar. The only way to meet this constraint is to disallow pipelining in the valve-based system; thus an upper bound on the number of barcoding droplets that can be processed simultaneously is equal to  $m$ . Since we consider a large number of cells, we divide the cells into batches, each of a maximum size of  $m$  cells, such that ReSyn executes them iteratively.

In the rest of this section, we assume a microfluidic platform that contains a fully connected crossbar, except Section VIII-D, in which we investigate variants of the crossbar architecture.

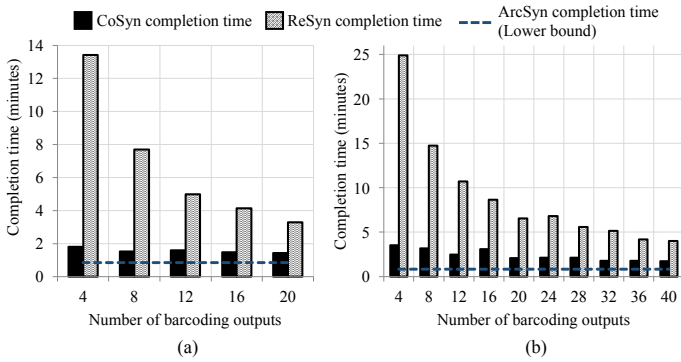


Fig. 11: Comparison between CoSyn, ArcSyn, and ReSyn in terms of completion time: (a) using 20 barcoding inputs, (b) using 40 barcoding inputs.

### A. Comparison with Baselines

We evaluate the performance of CoSyn, ArcSyn, and ReSyn in terms of the total completion time for the protocol, measured in minutes (we assume  $T_f = T_d = 0.2$  s). For valve-based routing, we consider the graph theoretic scheme (i.e., Method 1 from Section VI-A). We fix the number of input cells to 100, and we consider 20 and 40 barcoding inputs (or cell types). To ensure that this evaluation is independent of the platform architecture, the results were obtained using a DMFB with no resource constraints.

Fig. 11 compares the three synthesis frameworks in terms of completion times. ReSyn leads to the highest completion times due to the loose coordination between the DMFB and the valve-based crossbar. The completion time of CoSyn is close to the lower bound, which is obtained using ArcSyn. ArcSyn uses the maximum number of barcoding outputs due to the one-to-one mapping between the barcoding inputs and outputs. Hence, these results indicate that pipelined valve-based routing and the coordination between the components of CoSyn play a key role in increasing cell-analysis throughput.

### B. Tradeoffs of Valve-Based Routing Schemes

Next, we evaluate the performance and the computation time of the valve-based routing schemes by using CoSyn simulations. We refer to CoSyn that utilizes the graph-theoretic method (Method 1) and the incremental method (Method 2) as CoSyn-Graph and CoSyn-Inc, respectively. We fix the number of input cells to 1000 and we consider a DMFB with no resource constraints. Fig. 12(a) and Fig. 12(b) compare CoSyn-Graph and CoSyn-Inc in terms of completion times and overall computation times, respectively, while varying the number of inputs  $n$  and outputs  $m$  in the crossbar.

As shown in Fig. 12(a), we observe that the completion times of both methods exhibit a parabolic behavior with respect to the crossbar size  $n \times m$ . First, the completion times decrease when we increase the crossbar size from  $20 \times 8$  to  $50 \times 20$ ; the increase in the crossbar size within this range allows more barcoding droplets to be consumed by the crossbar, which in turn overshadows the negative impact of adding more vertical levels. However, when the size increases beyond  $50 \times 20$ , the impact of having additional vertical levels

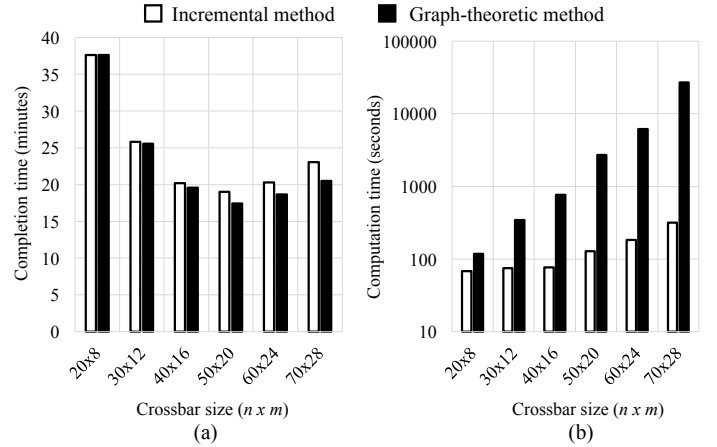


Fig. 12: Comparison between CoSyn-Graph and CoSyn-Inc: (a) completion time, (b) computation time (logarithmic scale).

appears to be more prominent, leading to an increase in the completion times for both methods. Finally, we also observe that the completion time obtained by CoSyn-Inc is at least the completion time obtained by CoSyn-Graph, and the difference in the completion time between CoSyn-Graph and CoSyn-Inc becomes significant when the crossbar size is increased.

Despite the high performance gained by using CoSyn-Graph, Fig. 12(b) shows that the computation time of CoSyn-Graph dramatically increases when larger crossbar designs are used. Therefore, it is obvious that CoSyn-Inc needs to be used to synthesize large-scale single-cell analysis.

### C. Design-Quality Assessment

We also evaluate the quality of the designs generated by CoSyn-Graph in terms of the number of single-cell experiments that can be completed for a given time limit and the given number of DMFB resources. In addition, we also quantify the fraction of input cells that can be processed simultaneously by the given set of DMFB resources. Our objective here is to investigate the conditions under which CoSyn-Graph is effective. Therefore, we introduce the following terms:

**Cell-analysis density:** The number of cells (samples) that completed analysis during a specific window of time (cell throughput), using a given array of electrodes. The time window is set to be a minute and the size of the array is equal to 100 electrodes.

**DMFB capacity:** A real number  $z \in [0, 1]$  that provides the fraction of input cells that can be processed simultaneously using DMFB resources. For example, a capacity of 1 indicates that there are sufficient resources to process all the cells simultaneously. On the other hand, a capacity of 0.5 means that the existing resources are sufficient for simultaneously processing only half of the cells.

We investigate the design-quality for valve-based crossbars by evaluating the cell-analysis density of CoSyn-Graph and ArcSyn. We simulate the execution of 50 cells using 4 barcoding outputs (Fig. 13(a)), 8 barcoding outputs (Fig. 13(b)), 12 barcoding outputs (Fig. 13(c)), and 20 barcoding outputs (Fig. 13(d)). The density values are computed while the

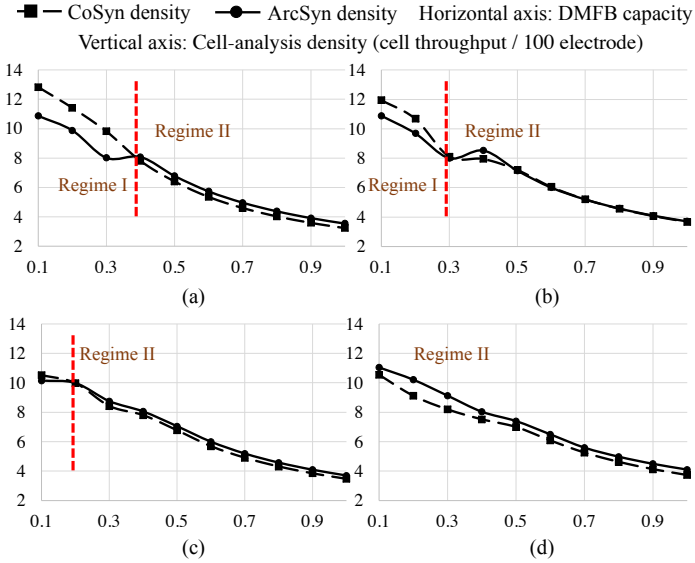


Fig. 13: Comparison between CoSyn-Graph and ArcSyn in terms of cell-analysis density using (a) 4 barcoding outputs, (b) 8 barcoding outputs, (c) 12 barcoding outputs, and (d) 20 barcoding outputs.

By comparing the density values for CoSyn-Graph and ArcSyn, we observe two regimes: (1) Regime I in which the cell-analysis density of CoSyn is higher, i.e., it is more effective; (2) Regime II in which the density of CoSyn-Graph is less than or equal to the density of ArcSyn. Regime I highlights the fact that CoSyn-Graph efficiently exploits valve-based barcoding, and the power of valve-based pipelining is evident when the DMFB resources are limited. On the other hand, the overprovisioning of resources leads to Regime II, where a lower cell-analysis density is reported. Finally, we note that Regime I shrinks as we increase the number of barcoding outputs; this is expected since CoSyn is more effective in the realistic case of a limited number of barcoding interfaces.

#### D. Crossbar Connectivity: Performance vs. Flexibility

We also study the impact of crossbar connectivity on the system performance and the routing flexibility. We use architecture variants  $\mathcal{F}_{n \times m}^{cl}$  of an  $n$ -to- $m$  crossbar; the parameters  $n$ ,  $m$ , and  $cl$  denote the number of crossbar inputs, the number of crossbar outputs, and the number of output clusters, respectively (Section VI-C). For evaluation purposes, we simply measure the flexibility, denoted by  $fx(m, cl)$ , of a crossbar variant  $\mathcal{F}_{n \times m}^{cl}$  using the following relation:  $fx(m, cl) = \frac{m}{cl}$ , which represents the effective number of output ports reachable from an input port. Also, we simulate the execution of 1000 cells using CoSyn-Inc, and we consider an 80-to-32 crossbar ( $n = 80$  and  $m = 32$ ). Moreover, to ensure that this evaluation applies to any DMFB architecture, the results were obtained using three DMFBs that has the following resource capacities  $z$ : 0.6, 0.8, and 1.

As shown in Table. III, as the number of output clusters increases, the flexibility  $fx$  of the crossbar design decreases, which eventually leads to degradation in the fault tolerance. Theoretical discussion on the fault tolerance of the valve-based system is skipped due to lack of space. On the other hand, as

TABLE III: Flexibility of crossbar variants ( $fx$ ).

1 Cluster	2 Clusters	4 Clusters	8 Clusters
32	16	8	4

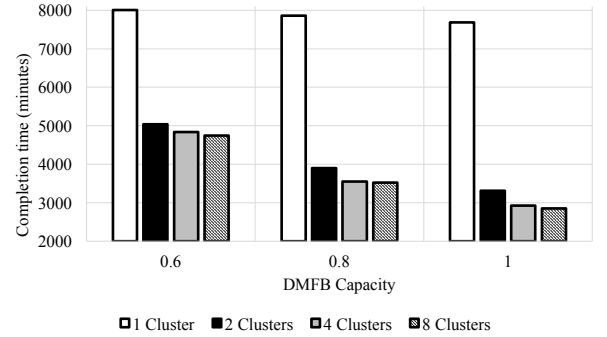


Fig. 14: Comparison between crossbar variants using protocol completion time.

shown in Fig. 14, we observe that the total completion time, i.e., performance improvement, decreases when the number of clusters increases—such an improvement in system performance is due to the reduction in the number of vertical levels  $q$  and thus the routing distance of the barcoding droplets.

#### E. Analysis of Markov Model

To understand the impact of protocol conditions and experiment budget on the protocol efficiency, we run model checking on the DTMC model described in Fig. 10, where the number of cells is equal to 6. We investigate the impact of three experimental factors: the number of washing steps ( $W_{Tot}$ ), the number of fixation time steps ( $fixTime_{Tot}$ ), and the number of master mixes ( $Mixes_{Tot}$ ). For this purpose, we use PRISM [43] to implement the DTMC model along with the following PCTL query that expresses the model objective:

$$Q := P_{max=?}[(\Box(\neg Failure)) U (\Diamond WithinBudget)]$$

The query  $Q$  seeks the maximum probability that the protocol will globally avoid the state *Failure* “until” eventually reaching the success state, i.e., the state *WithinBudget*. The operator  $\Box$  means “globally”, the operator  $U$  means “until”, and the operator  $\Diamond$  means “eventually”. The maximum probability can be computed using *value iteration* algorithm [44]. By running this query using PRISM, we can obtain the optimal protocol strategy that maximizes the probability of satisfying the target predicate, i.e., the processed cells contribute to high efficiency and the experiment budget is not exceeded.

Fig. 15 shows the obtained results, which illustrate the impact of experiment budget on the probability that single-cell analysis completes successfully under the optimal protocol strategy. It is obvious that the success probability increases when we increase the experiment budget. We also observe that the success probability is close to 1 only if there are sufficiently large quantities of washing liquid and master mixes and if we allow more time for fixation.

These results highlight the need to incorporate stochastic behavior of a single-cell protocol into any synthesis methodology since it has a significant impact on the protocol efficiency.

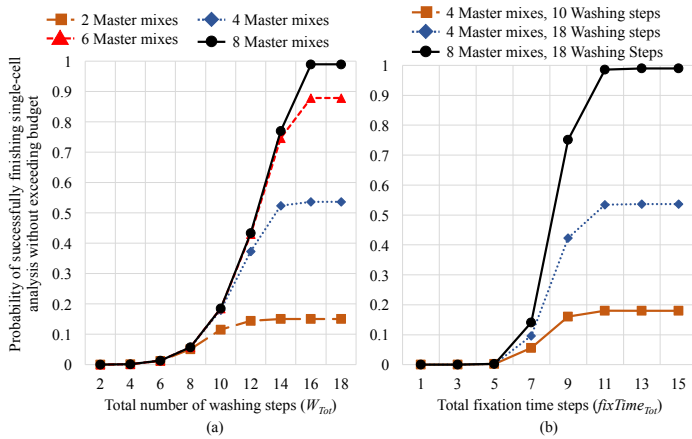


Fig. 15: Impact of experiment budget on the probability of successful completion of single-cell analysis under the optimal protocol strategy: (a) with varying number of washing steps; (b) with varying fixation time.

## IX. CONCLUSION

We have introduced the first automated design method for single-cell analysis using a cyberphysical microfluidic platform. This design coordinates the control of diverse microfluidic components and concurrently processes a large number of sample pathways. We have also presented a probabilistic model of the single-cell analysis protocol based on DTMCs. This model captures experimental settings where protocol conditions are specified in a probabilistic manner. The proposed platform has been evaluated on the basis of the time needed for analysis and the biochip size needed for realistic test cases.

## ACKNOWLEDGMENT

The authors thank Prof. Kristin Scott from the Department of Molecular Genetics and Microbiology at Duke University for providing the data related to the performance of the ChIP protocol.

## REFERENCES

- [1] S. Hoscic *et al.*, “Microfluidic sample preparation for single cell analysis,” *Analytical Chemistry*, vol. 88, no. 1, pp. 354–380, 2015.
- [2] T. Q. Vu, R. M. B. de Castro, and L. Qin, “Bridging the gap: microfluidic devices for short and long distance cell–cell communication,” *Lab on a Chip*, vol. 17, no. 6, pp. 1009–1023, 2017.
- [3] A. Saadatpour *et al.*, “Single-cell analysis in cancer genomics,” *Trends in Genetics*, vol. 31, no. 10, pp. 576–586, 2015.
- [4] J. Chen *et al.*, “Microfluidic impedance flow cytometry enabling high-throughput single-cell electrical property characterization,” *International Journal of Molecular Sciences*, vol. 16, no. 5, pp. 9804–9830, 2015.
- [5] N. Shembekar *et al.*, “Droplet-based microfluidics in drug discovery, transcriptomics and high-throughput molecular genetics,” *Lab on a Chip*, vol. 16, no. 8, pp. 1314–1331, 2016.
- [6] R. Zilionis *et al.*, “Single-cell barcoding and sequencing using droplet microfluidics,” *Nature Protocols*, vol. 12, no. 1, pp. 44–73, 2017.
- [7] A. M. Klein *et al.*, “Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells,” *Cell*, pp. 1187–1201, 2015.
- [8] A. Rival *et al.*, “An EWOD-based microfluidic chip for single-cell isolation, mRNA purification and subsequent multiplex qPCR,” *Lab on a Chip*, vol. 14, no. 19, pp. 3739–3749, 2014.

- [9] M. J. Jebrail *et al.*, “World-to-digital-microfluidic interface enabling extraction and purification of RNA from human whole blood,” *Analytical Chemistry*, vol. 86, no. 8, pp. 3856–3862, 2014.
- [10] S. C. Shih *et al.*, “A droplet-to-digital (D2D) microfluidic device for single cell assays,” *Lab on a Chip*, vol. 15, no. 1, pp. 225–236, 2015.
- [11] L. Arrigoni, A. S. Richter, E. Betancourt, K. Bruder, S. Diehl, T. Manke, and U. Bönisch, “Standardizing chromatin research: a simple and universal method for ChIP-seq,” *Nucleic Acids Research*, vol. 44, no. 7, pp. e67:1–13, 2016.
- [12] T. S. Furey, “ChIP-seq and beyond: new and improved methodologies to detect and characterize protein–DNA interactions,” *Nature Reviews Genetics*, vol. 13, no. 12, p. 840, 2012.
- [13] R. B. Fair, “Digital microfluidics: is a true lab-on-a-chip possible?” *Microfluidics and Nanofluidics*, vol. 3, no. 3, pp. 245–281, 2007.
- [14] T. Thorsen *et al.*, “Microfluidic large-scale integration,” *Science*, vol. 298, no. 5593, pp. 580–584, 2002.
- [15] F. Su and K. Chakrabarty, “High-level synthesis of digital microfluidic biochips,” *ACM JETC*, vol. 3, no. 4, p. 1, 2008.
- [16] T. Xu and K. Chakrabarty, “Integrated droplet routing in the synthesis of microfluidic biochips,” in *Proc. IEEE/ACM Design Automation Conference (DAC)*, 2007, pp. 948–953.
- [17] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, “BioRoute: A network-flow based routing algorithm for digital microfluidic biochips,” in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2007, pp. 752–757.
- [18] O. Keszcze *et al.*, “A general and exact routing methodology for digital microfluidic biochips,” in *Proc. IEEE/ACM ICCAD*, 2015, pp. 874–881.
- [19] T. Xu and K. Chakrabarty, “Broadcast electrode-addressing for pin-constrained multi-functional digital microfluidic biochips,” in *Proc. IEEE/ACM Design Automation Conference (DAC)*, 2008, pp. 173–178.
- [20] D. T. Grissom and P. Brisk, “Fast online synthesis of digital microfluidic biochips,” *IEEE TCAD*, vol. 33, no. 3, pp. 356–369, 2014.
- [21] T.-W. Huang *et al.*, “A network-flow based pin-count aware routing algorithm for broadcast-addressing EWOD chips,” *IEEE TCAD*, vol. 30, no. 12, pp. 1786–1799, 2011.
- [22] M. Ibrahim *et al.*, “Advances in design automation techniques for digital-microfluidic biochips,” in *Formal Modeling and Verification of Cyber-Physical Systems*. Springer, 2015, pp. 190–223.
- [23] W. H. Minhass *et al.*, “System-level modeling and synthesis of flow-based microfluidic biochips,” in *Proc. CASES*, 2011, pp. 225–233.
- [24] T.-M. Tseng *et al.*, “Columbia: co-layout synthesis for continuous-flow microfluidic biochips,” in *Proc. DAC*, 2016.
- [25] K. Hu *et al.*, “Control-layer optimization for flow-based mvlsi microfluidic biochips,” in *Proc. IEEE/ACM CASES*, 2014, p. 16.
- [26] H. Yao *et al.*, “PACOR: practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips,” in *Proc. IEEE/ACM DAC*, 2015, p. 142.
- [27] Y. Luo *et al.*, “Error recovery in cyberphysical digital microfluidic biochips,” *IEEE TCAD*, vol. 32, no. 1, pp. 59–72, 2013.
- [28] M. Alistar *et al.*, “Redundancy optimization for error recovery in digital microfluidic biochips,” *Design Automation for Embedded Systems*, vol. 19, no. 1-2, pp. 129–159, 2015.
- [29] C. Jaress *et al.*, “Rapid online fault recovery for cyber-physical digital microfluidic biochips,” in *Proc. IEEE VLSI Test Symposium (VTS)*, 2015, pp. 1–6.
- [30] M. Ibrahim and K. Chakrabarty, “Efficient error recovery in cyberphysical digital-microfluidic biochips,” *IEEE Transactions on Multi-Scale Computing Systems (TMSCS)*, vol. 1, no. 1, pp. 46–58, 2015.
- [31] S. Poddar *et al.*, “Error-correcting sample preparation with cyberphysical digital microfluidic lab-on-chip,” *ACM TODAES*, vol. 22, no. 1, p. 2, 2016.
- [32] J. Gong *et al.*, “All-electronic droplet generation on-chip with real-time feedback control for EWOD digital microfluidics,” *Lab on a Chip*, 2008.

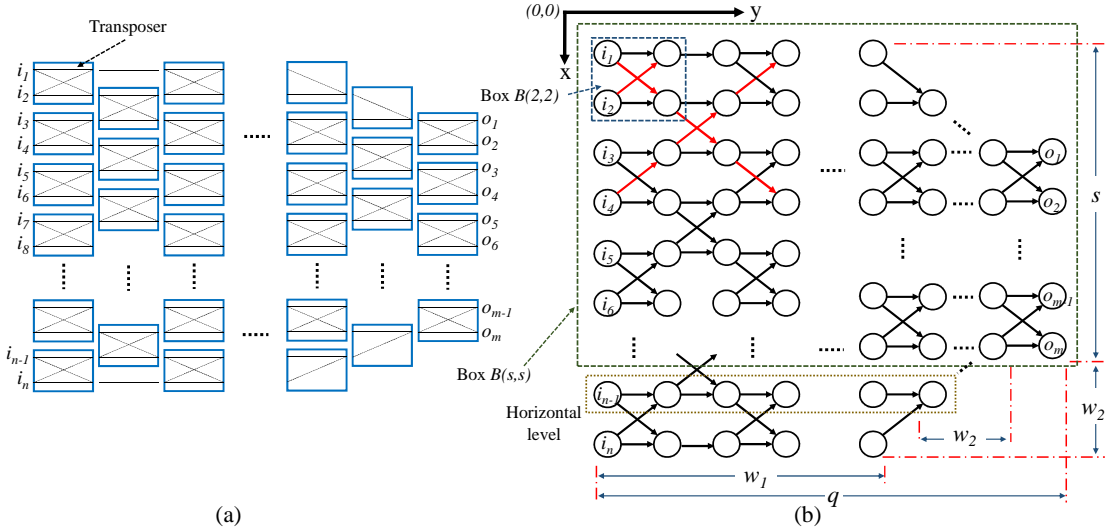


Fig. 16: (a) A schematic view of an  $n$ -to- $m$  routing crossbar; (b) the associated graph model  $\mathcal{F}_{(n \times m, \mathcal{T})}$ .

- [33] Y. Luo *et al.*, "Design and optimization of a cyberphysical digital-microfluidic biochip for the polymerase chain reaction," *IEEE TCAD*, vol. 34, no. 1, pp. 29–42, 2015.
- [34] M. Ibrahim *et al.*, "Synthesis of cyberphysical digital-microfluidic biochips for real-time quantitative analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, no. 5, pp. 733–746, 2017.
- [35] M. Ibrahim *et al.*, "A real-time digital-microfluidic platform for epigenetics," in *Proc. IEEE/ACM CASES*, 2016, p. 10.
- [36] S. Gómez, A. Arenas, J. Borge-Holthoefer, S. Meloni, and Y. Moreno, "Discrete-time markov chain approach to contact-based disease spreading in complex networks," *EPL (Europhysics Letters)*, vol. 89, no. 3, p. 38009, 2010.
- [37] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of model checking*, 2008.
- [38] M. Kwiatkowska, "Quantitative verification: models techniques and tools," in *Proc. 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2007, pp. 449–458.
- [39] M. A. Murran and H. Najjaran, "Capacitance-based droplet position estimator for digital microfluidic devices," *Lab on a Chip*, 2012.
- [40] M. Ibrahim *et al.*, "Integrated and real-time quantitative analysis using cyberphysical digital-microfluidic biochips," in *Proc. DATE*, 2016.
- [41] R. Silva *et al.*, "A reconfigurable continuous-flow fluidic routing fabric using a modular, scalable primitive," *Lab on a Chip*, 2016.
- [42] T. Cormen *et al.*, *Introduction to Algorithms*. MIT Press Cambridge, 2001.
- [43] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*. Springer, 2011, pp. 585–591.
- [44] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis, "Automatic verification of competitive stochastic systems," *Formal Methods in System Design*, vol. 43, no. 1, pp. 61–92, 2013.

## APPENDIX A

### PROOF OF THEOREM 1: A FULLY CONNECTED ROUTING CROSSBAR

Our objective here is to provide an inductive proof for Theorem 1 in the paper. We aim to design an  $n$ -to- $m$  routing crossbar, as shown in Fig. 16(a). The associated graph model

$\mathcal{F}_{(n \times m, \mathcal{T})}$  (defined in the paper) is depicted in Fig. 16(b);  $n$  is the number of inputs,  $m$  is the number of outputs, and  $\mathcal{T}$  is a directed acyclic graph that represents the crossbar transposers and their interconnections. It is required to design a routing crossbar such that it is fully connected; the definition of full connectivity is given below.

**Definition A.1.** An  $n$ -to- $m$  routing crossbar is fully connected if and only if we can construct  $\mathcal{F}_{(n \times m, \mathcal{T})}$  such that a droplet at any input  $i_k \in \{i_1, i_2, \dots, i_n\}$  can reach any output  $o_j \in \{o_1, o_2, \dots, o_m\}$ .

More specifically, the problem statement is described as follows:

"What constraint must be enforced on  $q$  such that a droplet at  $i_1$  can reach  $o_m$ , and a droplet at  $i_n$  can reach  $o_1$  through  $\mathcal{F}_{(n \times m, \mathcal{T})}$ , given that  $n$  and  $m$  are even integers?"

To simplify notation, we denote  $\mathcal{F}_{(n \times m, \mathcal{T})}$  by  $\mathcal{F}_{n \times m}$  throughout this document. Assume that  $\mathcal{F}_{n \times m}$  is placed in a two-dimensional space with x-y coordinates; the origin is located at the top-left corner (Fig. 16(b)). Let  $B(x, y)$  be a bounding box applied to  $\mathcal{F}_{n \times m}$  such that it covers  $x$  horizontal levels and  $y$  vertical levels starting from the origin. Clearly, the crossbar consists of interconnected transposers that are topologically equivalent; thus we view the model  $\mathcal{F}_{n \times m}$  as it is constructed by iteration, as illustrated in Fig. 17. This feature intuitively inspires us to use proof by induction to investigate crossbar full connectivity.

**The Base Case:** We start with the smallest scale of a routing crossbar, which is a single transposer (defined in Section III). At this scale, the inputs and outputs of the associated graph  $\mathcal{F}_{2 \times 2}$  are directly connected. The graph model  $\mathcal{F}_{2 \times 2}$  of the top left-most transposer is encapsulated in the bounding box  $B(2, 2)$ , as shown in Fig. 16(b). Based on the reconfigurable design of the transposer (Section III in the paper), we introduce the following lemma:

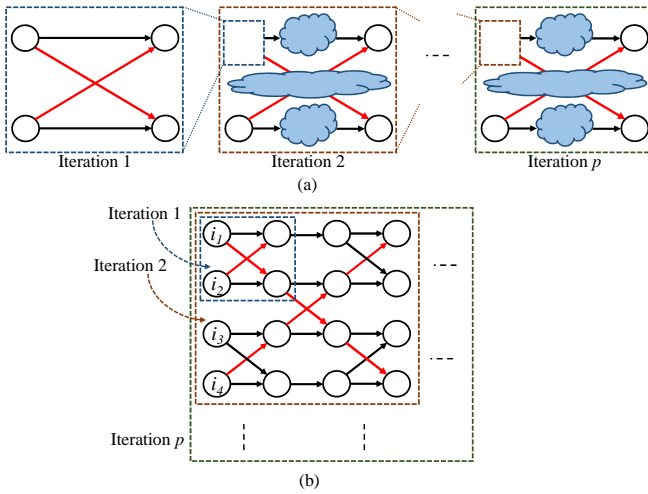


Fig. 17: Iterative construction of an  $\mathcal{F}_{n \times m}$ .

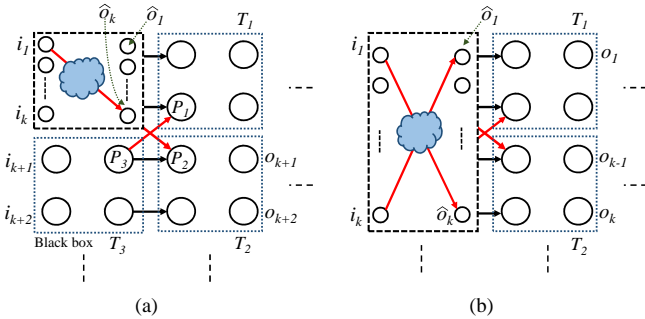


Fig. 18: Expanding the graph model  $\mathcal{F}_{k \times k}$  to (a)  $\mathcal{F}_{(k+2) \times (k+2)}$ , (b)  $\mathcal{F}_{k \times (k+2)}$ .

**Lemma A.1.** A full (half) transposer is a 2-to-2 (2-to-1) fully connected crossbar.

*Proof:* We present the proof by contradiction. Suppose that a full transposer is not fully connected. This implies that in the associated graph  $\mathcal{F}_{2 \times 2}$ , there must be at least one input  $i_k \in \{i_1, i_2\}$  that is not directly connected to one of the outputs  $o_j \in \{o_1, o_2\}$ . However, since only input and output nodes exist in  $\mathcal{F}_{2 \times 2}$ , i.e., there are no intermediate vertical levels between the inputs and outputs, then there are no alternative paths between  $i_k$  and  $o_j$ . This contradicts the original assumption about reconfigurability of transposers; that is, there is always a path between any input and any output. As a result, a full transposer must be fully connected to preserve reconfigurability. The proof is similar for half transposers. ■

**The Induction Step:** Next, we study a generic model  $\mathcal{F}_{k \times k}$ , where  $k$  is an even integer, and  $\mathcal{F}_{k \times k}$  is bounded by  $B(k, k)$ . Suppose that  $\mathcal{F}_{k \times k}$  represents a crossbar that is fully connected. Then, the input  $i_1$  is connected to the output  $\hat{o}_k$ . Similarly,  $i_k$  is connected to  $\hat{o}_1$ ; see Fig. 18(a). By expanding the graph model to  $\mathcal{F}_{(k+2) \times (k+2)}$  (bounded by  $B(k+2, k+2)$ ), we obtain the following result.

**Lemma A.2.** If a graph  $\mathcal{F}_{k \times k}$  represents a fully connected crossbar, then the expanded graph  $\mathcal{F}_{(k+2) \times (k+2)}$  also models

a fully connected crossbar.

*Proof:* By expanding  $\mathcal{F}_{k \times k}$  to  $\mathcal{F}_{(k+2) \times (k+2)}$ , we connect three transposers  $T_1$ ,  $T_2$ , and  $T_3$  to  $\mathcal{F}_{k \times k}$ , as shown in Fig. 18(a). Since each transposer is fully connected (Lemma A.1), we guarantee that the inputs of  $T_3$  (i.e.,  $i_{(k+1)}$  and  $i_{(k+2)}$ ) are connected to  $P_3$ . Likewise,  $P_2$  in  $T_2$  is connected to the two outputs  $o_{(k+1)}$  and  $o_{(k+2)}$ , and  $P_1$  is connected to the outputs of  $T_1$ . According to the graph model topology, directed edges are constructed from  $P_3$  to  $P_1$  and  $P_2$ , and from  $\hat{o}_k$  to  $P_1$  and  $P_2$ . This implies that there is always a path from any input in  $\mathcal{F}_{k \times k}$  and  $T_3$  to all the outputs of  $T_1$  and  $T_2$ . As a result,  $\mathcal{F}_{(k+2) \times (k+2)}$  represents a fully connected crossbar. We reach the same conclusion if  $T_1$ ,  $T_2$ , or both are replaced with half transposers. ■

Furthermore, consider expanding  $\mathcal{F}_{k \times k}$  to  $\mathcal{F}_{k \times (k+2)}$  following the same approach; see Fig. 18(b). Intuitively,  $\mathcal{F}_{k \times (k+2)}$  also represents a fully connected crossbar, since  $i_1$  is connected to  $o_k$ , and  $i_k$  is connected to  $o_1$ . Therefore, we introduce the following lemma.

**Lemma A.3.** If a graph  $\mathcal{F}_{k \times k}$  represents a fully connected crossbar, then the expanded graph  $\mathcal{F}_{k \times (k+2)}$  also models a fully connected crossbar.

*Proof:* Similar to the proof of Lemma A.2. ■

Using Lemma A.1 and Lemma A.2, and by induction, we obtain the following theorem.

**Theorem A.1.** A graph  $\mathcal{F}_{k \times k}$  represents a fully connected crossbar if  $k$  is an even integer and  $\mathcal{F}_{k \times k}$  is bounded by  $B(k, k)$ .

Consider Fig. 16(b), the graph model  $\mathcal{F}_{s \times s}$  bounded by  $B(s, s)$  is fully connected if  $s$  is even, according to Theorem A.1. To test this condition, the value of  $s$  can be calculated as follows:  $s = m + \frac{n-m}{2} = \frac{n+m}{2}$ . It is obvious that if  $m$  and  $n$  are even integers (given assumption), then  $s$  is even. Therefore,  $\mathcal{F}_{s \times s}$  is fully connected. Furthermore, due to the square shape of  $B(s, s)$ , we conclude that  $q = s = \frac{n+m}{2}$ . Based on Lemma A.3, the previous formula can be generalized to the following inequality:  $q \geq \frac{n+m}{2}$ , where  $n$  and  $m$  are even.

The same proof can also be applied if we move the origin of the space to the bottom left of Fig. 16(b) and repeat the calculation of  $q$  based on a bounding box expanding from the bottom part of the graph  $\mathcal{F}_{n \times m}$ . We conclude that  $o_1$  is connected to  $i_n$ . Therefore, we obtain the required result as follows.

**Theorem 1.** An  $n$ -to- $m$ ,  $q$ -level valve-based crossbar is a fully connected fabric if  $n$  and  $m$  are even integers, and  $q \geq \frac{m+n}{2}$ .

Note that this theorem provides a sufficient condition, which requires  $m$  and  $n$  to be even. A necessary condition remains to be derived as part of future work.

## APPENDIX B

### ALGORITHM FOR GENERATING THE GRAPH CORRESPONDING TO A FULLY CONNECTED ROUTING CROSSBAR

Using Fig. 16(b), we can calculate the parameters  $w_1$  and  $w_2$ , given  $m$ ,  $n$ , and  $q$ . Note that  $q$  is set to the lower bound,

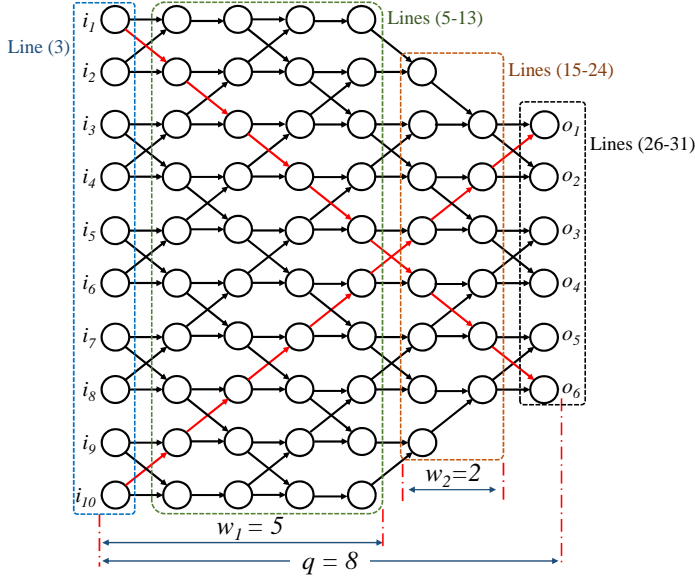


Fig. 19: The application of Algorithm 3 on a graph  $\mathcal{F}_{10 \times 6}$ .

i.e.,  $q = \frac{m+n}{2}$ .

$$w_2 = \frac{n-m}{2} \quad (1)$$

$$w_1 = q - w_2 - 1 = m - 1 \quad (2)$$

Algorithm 3 is implemented to generate the graph model  $\mathcal{F}_{n \times m}$  of a fully connected routing crossbar. For the sake of illustration, we apply the steps of the algorithm to generate the graph  $\mathcal{F}_{10 \times 6}$ ; see Fig. 19. The algorithm begins by calculating the parameters  $w_1$  and  $w_2$  (Line 2) using Equations (1)-(2);  $w_1 = 5$  and  $w_2 = 2$  for  $\mathcal{F}_{10 \times 6}$ . Then the algorithm generates the input nodes which reside at the first vertical level of  $\mathcal{F}_{10 \times 6}$  (Line 3). Next, the subsequent  $w_1 - 1$  vertical levels are generated (Lines 5-13); note that these levels incorporate decision nodes for only full transposers. In each level (Lines 7-12),  $n$  nodes are generated and connected with the appropriate nodes located in the previous level.

After completing the generation of the first  $w_1$  vertical levels, the following ( $w_2$ ) levels contain decision nodes that represent half transposers. Therefore, the variable  $s_h$  (skipped horizontal levels) is used to account for the change in the number of nodes generated at each vertical level (Line 16). The steps used for generating the nodes and the associated edges (Lines 15-24) are similar to the previous part. Finally, the algorithm generates the output nodes and the edges that connect them to the previous level (Lines 26-31).

The worst-case computational complexity of this algorithm is  $O(n^2 \cdot m)$ .

---

### Algorithm 3 Graph Model $\mathcal{F}_{n \times m}$ Generator

---

**Input:**  $m, n, q$

**Output:** Set of nodes  $\mathcal{D}_{n \times m}$  and set of edges  $\mathcal{S}_{n \times m}$

```

1:  $\mathcal{D}_{n \times m} \leftarrow \emptyset$ ;  $\mathcal{S}_{n \times m} \leftarrow \emptyset$ ;
2:  $w_1, w_2 \leftarrow \text{Initialize}(n, m, q)$ ;
3:  $\mathcal{D}_{n \times m} \leftarrow \text{GenerateInputNodes}(n)$ ; // 1st vertical level
4: // Generate the nodes and the directed edges for the next  $w_1 - 1$ 
   vertical levels.
5: for  $v_l = 2$  to  $w_1$  do
6:   //  $v_l$ : vertical level
7:   for  $h_l = 1$  to  $n$  do
8:      $\tau \leftarrow \text{GenerateNode}(v_l, h_l)$ ;
9:      $e \leftarrow \text{GenerateEdgesFromPreviousLevelToCurrentNode}(v_l - 1, N)$ ;
10:     $\mathcal{D}_{n \times m} \leftarrow \mathcal{D}_{n \times m} \cup \tau$ ;
11:     $\mathcal{S}_{n \times m} \leftarrow \mathcal{S}_{n \times m} \cup e$ ;
12:   end for
13: end for
14: // Generate the nodes and the directed edges for the next  $w_2$ 
   vertical levels (including half transposers)
15: for  $v_l = w_1 + 1$  to  $w_1 + w_2$  do
16:    $s_h = 1$ ; // Number of skipped horizontal levels
17:   for  $h_l = 1 + s_h$  to  $n - s_h$  do
18:      $\tau \leftarrow \text{GenerateNode}(v_l, h_l)$ ;
19:      $e \leftarrow \text{GenerateEdgesFromPreviousLevelToCurrentNode}(v_l - 1, N)$ ;
20:     $\mathcal{D}_{n \times m} \leftarrow \mathcal{D}_{n \times m} \cup \tau$ ;
21:     $\mathcal{S}_{n \times m} \leftarrow \mathcal{S}_{n \times m} \cup e$ ;
22:   end for
23:    $s_h = s_h + 1$ ;
24: end for
25: // Generate the nodes and the edges for the last (output) vertical
   level.
26: for  $h_l = w_2$  to  $w_2 + m$  do
27:    $\tau \leftarrow \text{GenerateOutputNodes}(h_l)$ ;
28:    $e \leftarrow \text{GenerateEdgesFromPreviousLevelToCurrentNode}(N)$ ; //
   Last vertical level
29:    $\mathcal{D}_{n \times m} \leftarrow \mathcal{D}_{n \times m} \cup \tau$ ;
30:    $\mathcal{S}_{n \times m} \leftarrow \mathcal{S}_{n \times m} \cup e$ ;
31: end for
32: return  $\mathcal{D}_{n \times m}, \mathcal{S}_{n \times m}$ ;

```

---