# 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 preisolated 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 co-synthesis (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 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.**

***Index Terms*—Cyberphysical integration, design automation, graph search, hybrid system, Markov chains, microfluidics, synthesis.**

## 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.

Manuscript received October 22, 2017; revised February 3, 2018; accepted May 17, 2018. Date of publication June 12, 2018; date of current version June 18, 2019. The work of M. Ibrahim was supported by the U.S. National Science Foundation under Grant CCF-1702596 and Grant CCF-1135853. The work of K. Chakrabarty was supported in part by the U.S. National Science Foundation under Grant CCF-1702596 and Grant CCF-1135853, and in part by the Technische Universität München—Institute for Advanced Study, through the German Excellence Initiative and the European Union Seventh Framework Programme under Grant 291763. This paper was recommended by Associate Editor S. Reda. *(Corresponding author: Mohamed Ibrahim.)*

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@duke.edu; krishnendu.chakrabarty@duke.edu).

U. Schlichtmann is with the Chair of Electronic Design Automation, Technical University of Munich, 80333 Munich, Germany (e-mail: ulf.schlichtmann@tum.de).

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

Digital Object Identifier 10.1109/TCAD.2018.2846662

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.
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 [4]. Therefore, indexing of droplets using barcodes is needed to keep track of their identity [5].
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 has been identified in step 1 [1]. To draw meaningful conclusions, the experimental outcomes are associated with droplet barcodes injected in step 2 [6].

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.

### A. Integration of Heterogeneous 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) [7] (step 3). However, DMFBs are not as effective for interfacing to the external world [8]. 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.

### B. 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 [6] (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 [9], it is unrealistic to assume that each dispensed droplet contains only one barcoding particle.

### C. Dynamic Synthesis

Due to the inherent uncertainty about cell types, cyber-physical 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.

### D. Stochasticity of Protocol Conditions

Despite the efforts made toward 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 [10]. Failing to characterize the variation in sonication time and other parameters among cell types may lead to degradation in down-stream immunoprecipitation (IP) and thus the overall ChIP performance [11]. 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.

1) 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.
2) We describe CoSyn, which enables coordinated control of the microfluidic components, and allows dual-domain synthesis for concurrent sample pathways.
3) 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.
4) We construct a DTMC model, which utilizes probabilistic information related to protocol steps and experiment budget to investigate the efficiency of probabilistic protocol decisions.
5) We evaluate system performance and reconfigurability while exploring various configurations of the valve-based system.

The rest of this paper is organized as follows. Section II presents an overview of related prior work and probabilistic formal methods that are relevant to this paper. 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 the 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 2-D array of electrodes and a set of on-chip resources [12]. Valve-based biochips, on the other hand, rely on special-purpose components (e.g., microvalves and micro-pumps) to manipulate liquid flow [13].

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) [14]–[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 methods.

Cyberphysical synthesis techniques have enabled online error recovery [27]–[30], volume precision [31], termination of biochemical applications, such as qPCR [32], and protocols for multiple samples [33], [34]. 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.

Recently, a flow-based design methodology has been introduced to support high-throughput single-cell applications [35]. This design employs a delay model of pressure-driven transport to satisfy a given throughput constraint, enabling single-cell applications to be efficiently executed. This paper, however, considers only the early stages of single-cell analysis, namely cell isolation and barcoding, and it does not consider the complete analysis flow.

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 $\mathcal{D} = \langle \mathcal{S}, s_{\text{init}}, \mathcal{U}, \mathcal{J} \rangle$, where: 1) $\mathcal{S}$ is a finite set of states (state space) and $s_{\text{init}} \in \mathcal{S}$ is the initial state; 2) $\mathcal{U} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ is a transition probability matrix such that $\sum_{s' \in \mathcal{S}} \mathcal{U}(s, s') = 1 \ \forall s \in \mathcal{S}$; and 3) $\mathcal{J} : \mathcal{S} \rightarrow 2^{\text{AP}}$ 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 Formulas 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 formulas to describe path or numerical queries.

1) $Q_1 \coloneqq P_{\geq 0.95}[\Diamond \phi]$, 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) $Q_2 \coloneqq P_{\max=?}[\Diamond^{<10} \phi]$, which seeks the maximum probability that the system will eventually reach a state $s$ satisfying $\phi$ in less than ten 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 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 [9].

### 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 [9]. A flow-based system can be configured to function as a droplet-in-channel device, allowing a two-phase flow to be generated. More specifically, encapsulation of individual cells is easily accomplished by considering three intersecting flows; an aqueous flow (containing cells) and two oil flows. These flows are pressure-driven by syringe pumps and therefore they can be carefully balanced, allowing aqueous droplets (containing single cells) to be automatically formed with a surrounding oil phase. Next, the resulting two-phase flow is transported to the digital microfluidic device through a capillary interface. Fig. 2 shows cell encapsulation and droplet generation using a two-phase flow.

On the other hand, the digital microfluidic device consists of two parallel plates. The gap that separates the two plates is flooded with oil, which acts as a filler medium. According

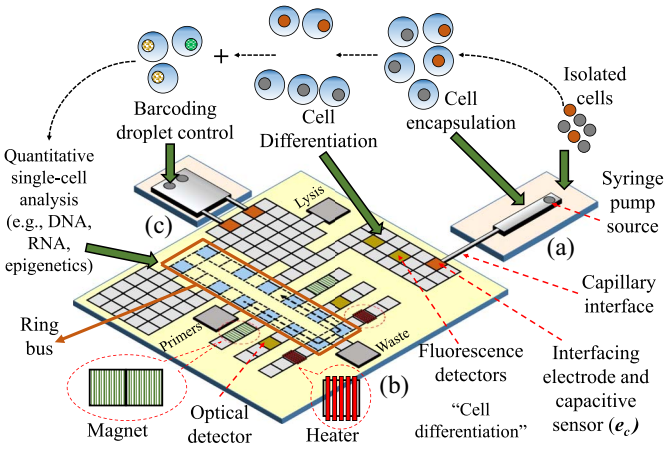



Fig. 1. Schematic of the hybrid platform for single-cell analysis. (a) Flow-based biochip used for cell encapsulation and droplet generation, (b) DMFB used for quantitative analysis, and (c) Reconfigurable valve-based fabric used for barcoding.

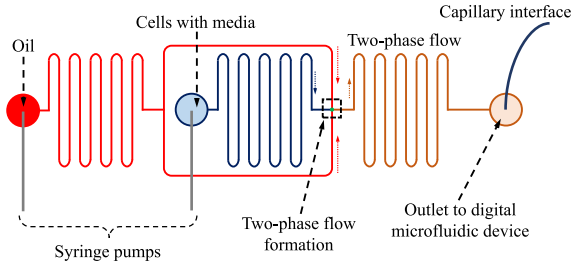



Fig. 2. Flow-based device used to generate aqueous droplets containing single cells as part of a two-phase flow [9].

to [9], the rate of oil injection between the two plates can be controlled using a feedback system in order to prevent the evaporation of droplets that are collected from the flow-based side. The oil medium also facilitates the injection of the two-phase flow into the digital side through the capillary interface.

To efficiently integrate both sides, the droplet generator uses the syringe pumps 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 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 (GEA) [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

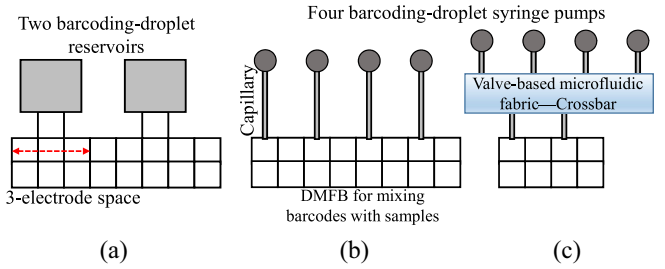



Fig. 3. Droplet barcoding using (a) DMFB, (b) valve-based biochip with one-to-one mapping between pumps and DMFB ports, and (c) valve-based biochip with many-to-many mapping.

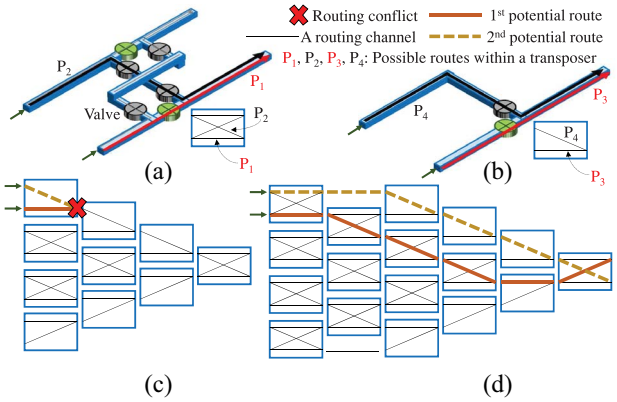



Fig. 4. Valve-based routing fabric for droplet barcoding (a) 2-input full transposer, (b) 2-input half transposer, (c) 4-level, 8-to-2 routing fabric, and (d) 6-level, 8-to-2 fabric.

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 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 [6]. 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. 3(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. 3(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 [6]. 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 that has $n$ input ports and $m$ output ports [41]; see Fig. 3(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 $n >> m$. The full connectivity offered by this crossbar allows droplets to be easily rerouted 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 $2m + k$. By unlocking this capability of valve-based 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.

In addition to the above advantages, the use of a reconfigurable valve-based fabric is important since it allows us to employ a design methodology that investigates the tradeoff between biochip cost and single-cell throughput and therefore provides more choices for biochip designers. For example, a biochip designer may prefer to utilize a cost-effective design, in which fewer electrodes are used at the digital side. To achieve this goal, i.e., to optimize for cost, sharing of flow channels among several barcoding inputs gains significant importance in order to minimize the number of peripherals to the DMFB. In other words, the role of the reconfigurable valve-based fabric becomes more significant. On the other hand, if a designer opts for a high-throughput platform, in which a large DMFB size is utilized, then sharing of flow channels among barcoding particles can be reduced, and therefore the role of the reconfigurable valve-based system becomes less significant. This observation has been studied and the results are reported in Section VIII-C.

We utilize the "transposer" primitive introduced in [41]. As shown in Fig. 4(a) and (b), a valve-based transposer appears in two forms: 1) a two-input, two-output transposer, which is composed of six valves, controlled via two pneumatic inputs (full transposer) and 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. 4(c), or using six levels, as shown in Fig. 4(d). A six-level crossbar, while incurring higher cost, provides a higher degree of reconfigurability and flexibility in routing. In Section VI-D, we study valve-based routing using various configurations of the crossbar.

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. |

### 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 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 GEA using qPCR [7], [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.

Ultimately, by supporting type-driven analysis, biologists can launch multiple single-cell applications, aiming for drawing a holistic picture of the interactions between different biomolecules (e.g., DNA, mRNA, protein, etc.). Note that our proposed framework can support the execution of multiple applications, a single application for all cell types, or even multiple applications for a stream of cells that are of the same type (cell barcoding can be used to distinguish these cells at the end of down-stream analysis).

### E. Platform Throughput and Scalability

By combining the two microfluidic formats (flow-based and digital microfluidics), we can maintain scalable single-cell analysis. As described earlier, flow-based microfluidics is efficient in generating thousands of droplets that encapsulate individual cells using a two-phase format. However, on the negative side, flow-based microfluidics relies on an etched micro-structure that, despite its capability in droplet preparation, fails to support reconfigurable analysis at the down-stream part. To overcome this drawback, digital microfluidics comes into play–digital microfluidics is well-suited for adding reagents in parallel, and reagents can be mixed on demand without the requirement of optimal and precise flow rates.

A DMFB is scalable in terms of handling multiple cells concurrently. In [7], a 230-electrode chip can be used to process 20 cells concurrently. By using our design-automation technique, the number of cells can be drastically increased since we allow resource sharing among pathways. However, note that a DMFB still provides a lower throughput compared to the flow-based devices; this observation does not mean that digital microfluidics cannot process droplets quickly, but it means that a DMFB is burdened with the largest portion of work for single-cell analysis. Further increase in the chip size can also increase the number of cells manipulated concurrently, thus leading to a higher throughput. Nevertheless, it needs to be clear that our objective is not to achieve the optimal throughput; our objective is to provide a cost-effective design methodology that employs the hybrid platform to process a continuous stream of cells with a reasonable throughput.

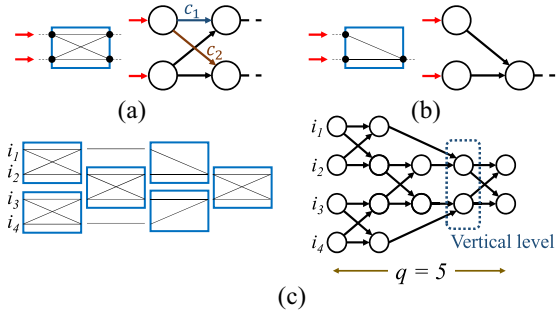



Fig. 5. Mapping a valve-based crossbar to a graph model. (a) Full transposer. (b) Half transposer. (c) 4-to-2 crossbar.

Based on the above discussion, flow-based microfluidics offers temporal scalability (high-throughput), whereas digital microfluidics offers reconfigurability and spatial scalability (concurrent single-cell analysis). Combining both technologies with an adequate feedback system (to synchronize throughput rates at both domains) provides spatio-temporal scalability for complete single-cell analysis.

## IV. MAPPING TO ALGORITHMIC MODELS

### A. Modeling of 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. 5(a) and (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. 5(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. 4 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. 5(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 Digital-Microfluidic Biochip

While DMFBs are highly reconfigurable and can support a diverse set of transport paths, we reduce the burden of

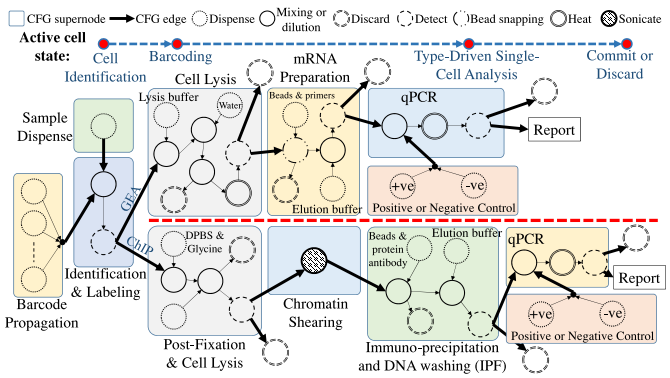



Fig. 6. CFG of type-driven analysis for a single cell pathway.

managing droplet transport in real-time by considering a unidirectional ring-based architecture, as shown in Fig. 1. Connected 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 bar-coding 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. 6. 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.

The model shown in Fig. 6 represents a protocol, where a dispensed cell can be processed using one of two biochemical procedures, namely GEA and ChIP procedures. As shown in the CFG model, the execution of the protocol starts with dispensing an aqueous sample droplet (sample dispense supernode). The type of the sample is then identified and mixed with an associated barcoding droplet that is routed through the reconfigurable valve-based fabric (identification and labeling supernode). Next, according to the cell type, either GEA or ChIP procedures will be executed. If GEA is selected, then fluid-handling operations of cell-lysis, mRNA preparation, control preparation, and qPCR will be carried out. At the end of each bioassay, a detection operation is performed to ensure that the efficiency of the resulting solution is above a certain threshold [33]. Similarly, if ChIP is selected, then fluid-handling operations of post-fixation, cell-lysis, chromatin shearing, immuno-precipitation, DNA washing, control preparation, and qPCR will be performed.

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. 6) is necessary to keep track of the cells that are being processed simultaneously.

TABLE II
NOTATION USED IN THE ALGORITHMS

| | | | |
|---|---|---|---|
| $C_i$ | Cell metadata | $\mathcal{G}$ | DMFB graph model |
| $R^d$ | Set of DMFB resources | $\mathcal{Y}$ | Set of fluidic-operation types |
| $\tilde{R}$ | Set of unoccupied DMFB resources ($\tilde{R} \subset R^d$) | | |
| $s$ | Minimum value of the cost function associated with $\tilde{R}$ | | |
| $\mathcal{F}_{n\times m}$ | Graph model of an $n$-to-$m$ crossbar | | |
| $P$ | Set of vertices in $\mathcal{F}_{n\times m}$ representing a routing path | | |
| $\mathcal{U}$ | Set of unoccupied vertices in $\mathcal{F}_{n\times m}$ | | |
| $\theta$ | Boolean variable: True if a path $P$ is complete | | |

## V. Co-Synthesis Methodology

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

### A. Problem Formulation

To approach the co-synthesis problem, we introduce the problem formulation as follows.

*Inputs:*

1) The protocol CFG $\mathcal{A}$.
2) 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 $\mathcal{A}$.
3) 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 $m$.
4) 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 by the individual cells, protocol completion time $T_{\text{comp}}$.

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

The notation used in this paper is summarized in Table II.

### B. Proposed Solution

The proposed CoSyn scheme, depicted in Fig. 7, 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; and 4) time-wheel engine, which seamlessly coordinates real-time interactions between

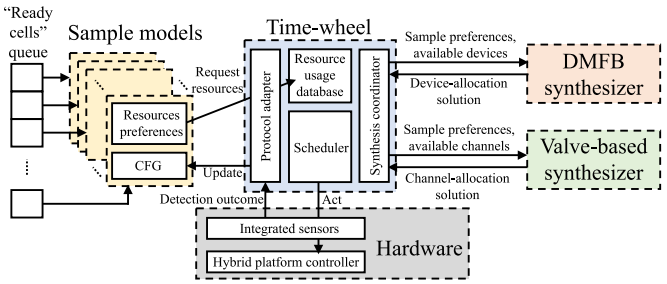



Fig. 7. Overview of the proposed CoSyn methodology.

**Algorithm 1** DMFB Resource Allocation

**Input:** $C_i$, $\mathcal{G}$, current simulation time "$t$"
**Output:** Assigned Resource "$r$"

```
R̃ ← GetCurrentlyUnoccupiedResources(G, t);
if (R̃ is empty) then return NULL;
y ← GetOperationType(C_i);
s ← CalculateMinimumCostAllAvailableResources(y, R̃);
if (s = ∞) then return NULL; // No suitable resource
r ← GetSelectedResource(s); return r;
```

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. 6).

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. 7) 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 is 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 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 decisions 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 $O(|\mathcal{V}|)$.

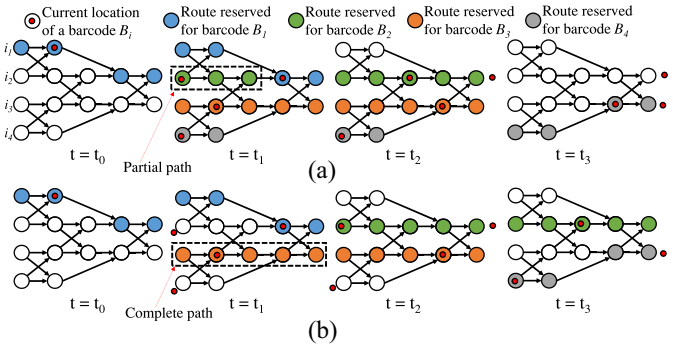



Fig. 8. Valve-based routing of four barcoding droplets (4-to-2 biochip). (a) With pipelining and (b) without pipelining.

## 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 [42].

*Theorem 1:* An $n$-to-$m$, $q$-level valve-based crossbar is a fully connected fabric if $n$ and $m$ are even integers, and $q \geq [(m+n)/2]$.

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 [42]. Using this model, a systematic methodology for droplet routing can be developed. In Section VI-A, we introduce the formulation of the routing problem and the solution approach.

### A. Problem Formulation and Solution Approach

To approach the valve-based routing problem, we introduce the problem formulation as follows.

*Inputs:*

1) The fabric model $\mathcal{F}_{n\times m}$, the number of inputs $n$ and the number of outputs $m$.
2) The matrix $C$ that encodes the cell state machine; each vector $C_i \in C$ corresponds to a cell (Section V).

*Constraints:*

1) A droplet must be routed through a path from its specified input port to any output port.
2) A droplet requires at least one time step to be transported from an intermediate vertex to another directly connected vertex (i.e., no jumps allowed).
3) At any time $t$, the routing paths of different droplets cannot overlap.

*Output:* Allocation of the graph vertices to barcoding droplets at all time steps.

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 point that is not reserved by other droplets (a pipeline stage), then allowed to move forward when a path is freed. Fig. 8 illustrates pipelined and nonpipelined routing.

In the following sections, we introduce two schemes for solving the routing problem. The first scheme in Section VI-B

**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: $\mathcal{U} \leftarrow$ GetCurrentlyUnoccupiedSubGraph($\mathcal{F}_{n\times m}$, $t$);
2: $d \leftarrow$ GetVertexCurrentlyHoldingBarcode($C_i$, $\mathcal{F}_{n\times m}$);
3: $P \leftarrow$ 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};

uses graph search to efficiently route barcoding droplets, whereas the second scheme in Section VI-C aims to improve system responsiveness (i.e., reduce computation time) via an incremental routing procedure.

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

We utilize a graph-theoretic algorithm to find vertex-disjoint shortest paths [43]; see Algorithm 2. By computing disjoint 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. 8). 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 computation of shortest paths can be simplified; the worst-case complexity of this algorithm is $O(|\mathcal{D}_{n\times m}| + |\mathcal{S}_{n\times m}|)$.

### C. Method 2: Incremental Routing

Method 1 is computationally expensive, despite the use of topological ordering, 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: 1) move straight; 2) move crossed; or 3) 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; and 3) a droplet must move straight if both the straight and crossed channels are unoccupied. Fig. 9 illustrates incremental routing using a 4-to-2 crossbar. The computational complexity of this procedure is $O(1)$.

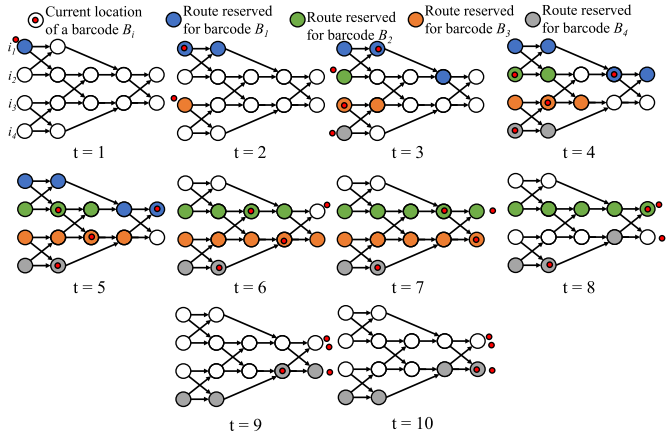



Fig. 9. Incremental routing of four droplets (4-to-2 biochip).

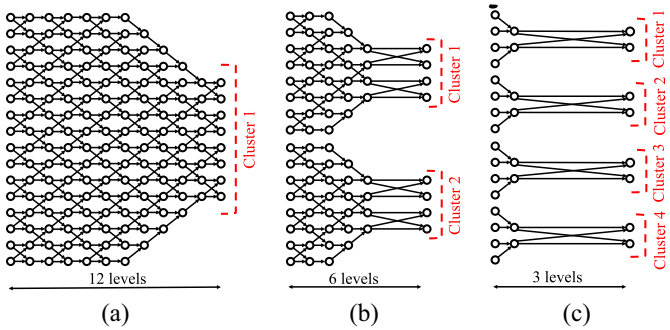



Fig. 10. Architectural variants of a 16-to-8 crossbar. (a) $\mathcal{F}^1_{16\times 8}$, (b) $\mathcal{F}^2_{16\times 8}$, and (c) $\mathcal{F}^4_{16\times 8}$.

### D. 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 $(n + m/2)$ 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 paper, 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, three of which are shown in Fig. 10. It is obvious that the configuration in Fig. 10(a) provides the highest connectivity, but it uses the largest number of vertical levels (12 levels). In contrast, the configuration in Fig. 10(c) 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 $\mathcal{F}_{8\times4}$ and two 4-to-2 crossbars $\mathcal{F}_{4\times2}$ [Fig. 10(c)], or it can be designed using a single 16-to-8 fully connected crossbar $\mathcal{F}_{16\times8}$ [Fig. 10(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. In other words, we can automatically generate the graph models of such crossbar variants using the algorithm described in [42]; only a minor modification is required to systematically combine the graph models associated with the constituting fully connected crossbars into a unified model.

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. 10(a) has a single cluster ($cl = 1$), the configuration in Fig. 10(b) has two clusters ($cl = 2$), and the configuration in Fig. 10(c) has four clusters ($cl = 4$). Hence, we denote a crossbar configuration by $\mathcal{F}^{cl}_{n\times m}$, where $n$ and $m$ are the number of inputs and outputs, respectively. Also, note that we can use $\mathcal{F}^{1}_{n\times m}$ and $\mathcal{F}_{n\times m}$ interchangeably since $\mathcal{F}^{1}_{n\times m}$ 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 sections. 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.

### 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 and 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 III) shows the normalized IP value, i.e., protocol efficiency,[1] while varying some experimental conditions, such as the number of washing steps after IP; 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.

TABLE III
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) |

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 III.
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 III. 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: $P(A) = [(|\mathcal{HE}|)/(|\mathcal{HE}| + |\mathcal{LE}|)]$. Based on Table III, $P(A) = (26/73) = 0.36$.
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 III (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: $P(B_i) = [|S_i|/(\sum_i |S_i|)]$, where $|S_i|$ represents the number of replicates processed using $S_i$. By considering ($S_3$: 10-min) from Table III, we obtain $P(B_3) = (51/73) = 0.7$.
6) We next compute the conditional probability $P(B_i|A) = [(P(B_i \cap A))/P(A)]$, 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: 1) the cells are processed using

[1]The higher the IP value, the more "enriched" the DNA is in the specific chromatin mark; i.e., leading to higher efficiency.

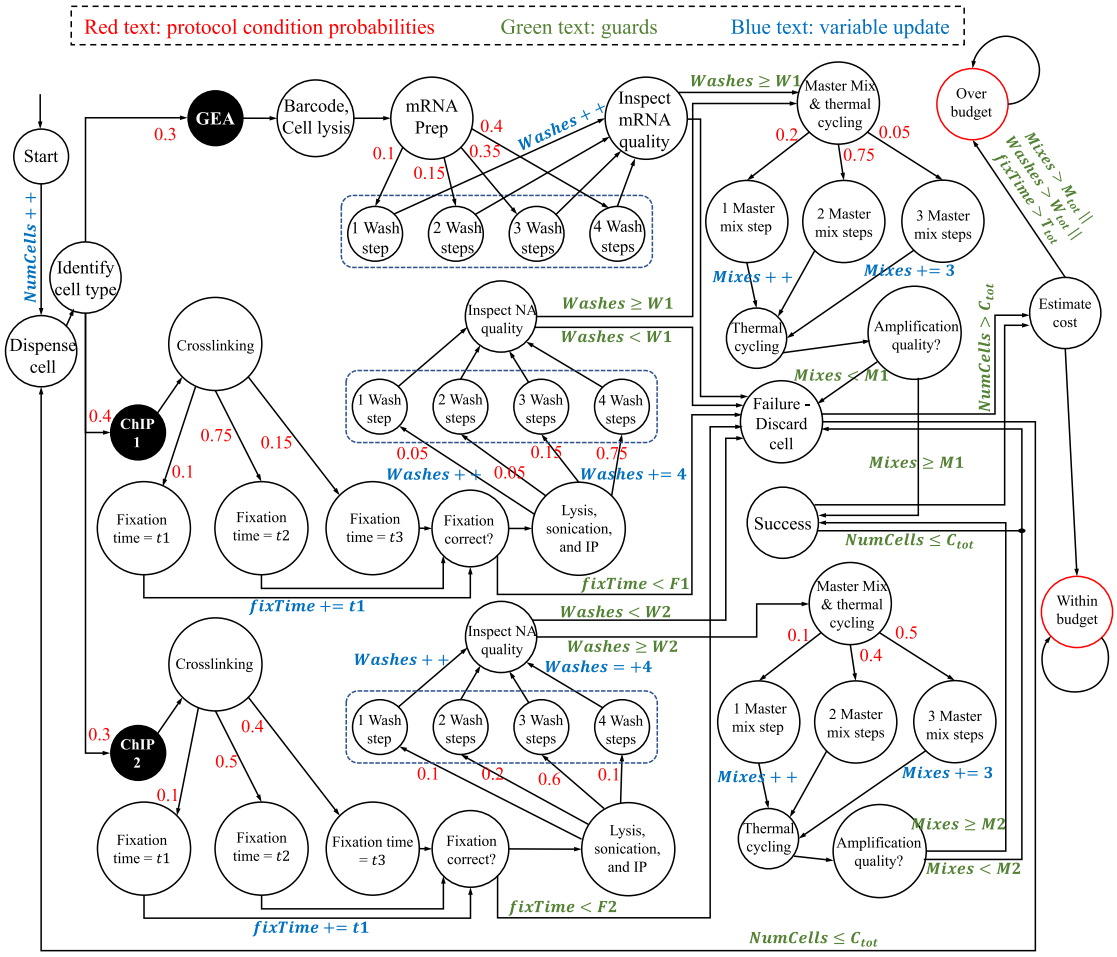



Fig. 11. DTMC model for a single-cell protocol that considers three cell types.

the setting $S_i$ and 2) the cell belong to $\mathcal{HE}$. In the example shown in Table III, $P(B_3 \cap A) = (14/73) = 0.2$, 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 $P(A|B_i) = [(P(B_i|A) \cdot P(A))/P(B_i)]$. According to the example in Table III, $P(A|B_3) = [(0.53 \times 0.36)/0.7] = 0.27$. 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: $\hat{P}(A|B_i) = [(P(A|B_i))/(\sum_i (A|B_i))]$.

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. Strategy synthesis is a problem that is concerned with finding a strategy, in our case a sequence of protocol steps, which satisfies a property or optimizes a long-term objective, such as the probability of protocol success [44]. Strategy synthesis has widely been used in many applications including planning of robots motion under uncertainty, security analysis via synthesis of malicious strategies, and dynamic power management using optimal control strategies.

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. 11 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 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. 11. 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 toward success (*Within budget*) or failure (*Over budget*). For example, in Fig. 11, 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 $C_{\text{tot}}$. 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.

Note that the above discussion considers a static DTMC model, which is computed only once based on the protocol-condition space. However, since stochasticity is ubiquitous in single-cell applications, protocol conditions can significantly change over time, and therefore there is a need to update the transition probabilities in the DTMC model. To cope with this dynamic behavior, a closed-loop system can be constructed, where multiple iterations can be performed. In this closed-loop system, the results obtained by the hybrid microfluidic system are added to the protocol-condition space, allowing a new set of transition probabilities to be computed; the new DTMC model is then used in the next iteration of strategy synthesis and single-cell analysis. The number of cells per iteration can be specified by the user. By following this approach, the DTMC model can be changed to adapt to protocol dynamics.

## 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. 3(b) and 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.

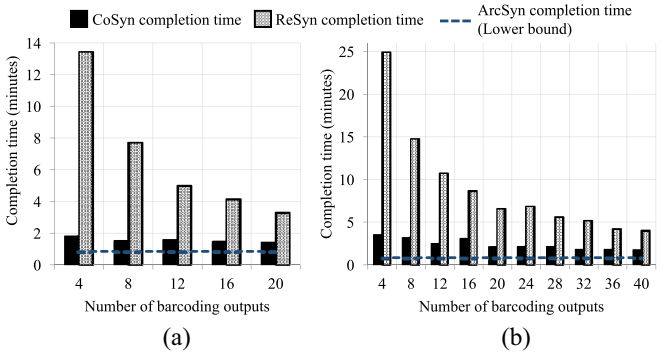



Fig. 12. Comparison between CoSyn, ArcSyn, and ReSyn in terms of completion time (a) using 20 barcoding inputs and (b) using 40 barcoding inputs.

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.

### 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-B). 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. 12 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. 13(a) and (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. 13(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

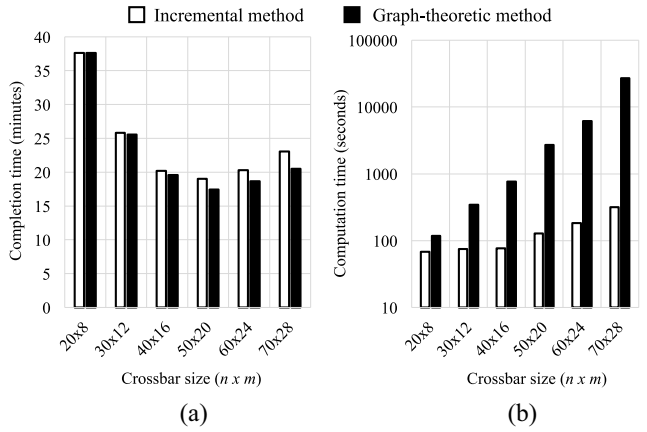



Fig. 13. Comparison between CoSyn-Graph and CoSyn-Inc (a) completion time and (b) computation time (logarithmic scale).

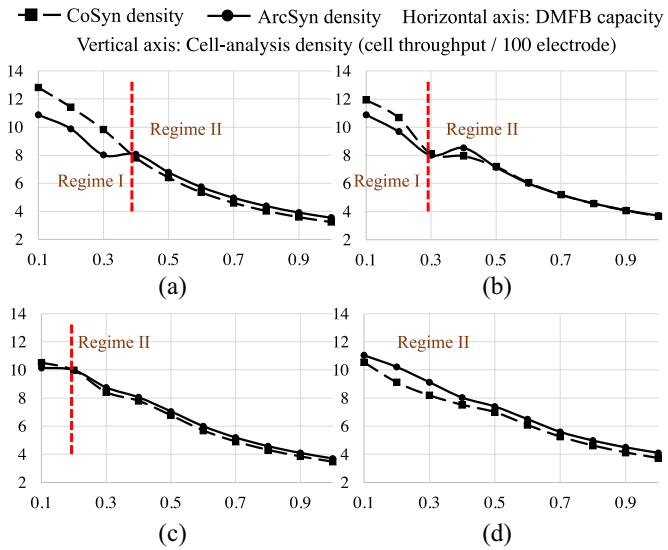



Fig. 14. Comparison between CoSyn-Graph and ArcSyn in terms of cell-analysis density using (a) 4 barcoding outputs, (b) 8 barcoding outputs, (c) 12 barcoding outputs, and (d) 20 barcoding outputs.

more vertical levels. However, when the size increases beyond $50 \times 20$, the impact of having additional vertical levels 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. 13(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 dynamically 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. 14(a)], 8 barcoding outputs [Fig. 14(b)], 12 barcoding outputs [Fig. 14(c)], and 20 barcoding outputs [Fig. 14(d)]. The density values are computed while the capacity is varied. 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-Graph is higher, i.e., it is more effective and 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-Graph is more effective in the realistic case of a limited number of barcoding interfaces.

TABLE IV
FLEXIBILITY OF CROSSBAR VARIANTS ($fx$)

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

### D. Crossbar Connectivity: Performance Versus Flexibility

We also study the impact of crossbar connectivity on the system performance and the routing flexibility. We use architecture variants $\mathcal{F}^{cl}_{n\times m}$ 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-D). For evaluation purposes, we simply measure the flexibility, denoted by $fx(m, cl)$, of a crossbar variant $\mathcal{F}^{cl}_{n\times m}$ using the following relation: $fx(m, cl) = (m/cl)$, 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 have the following resource capacities $z$: 0.6, 0.8, and 1.

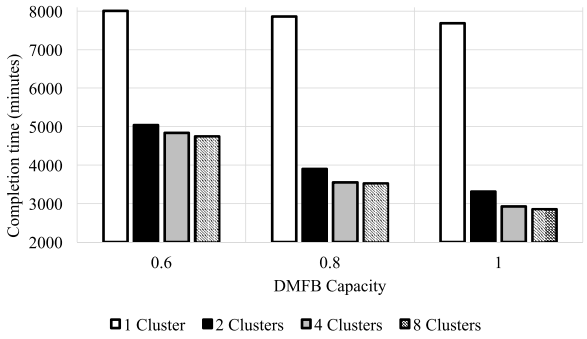



Fig. 15. Comparison between crossbar variants using protocol completion time.

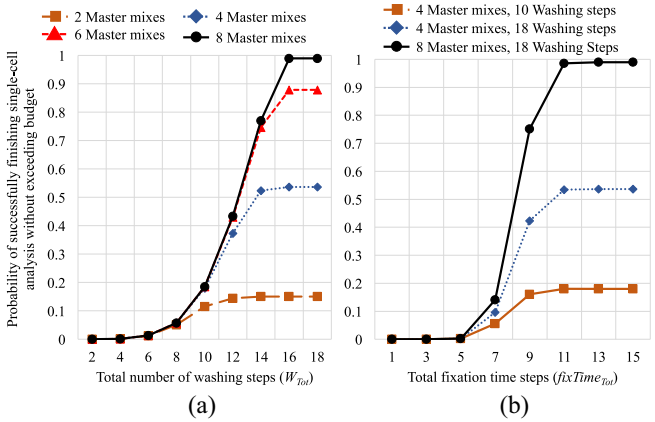



Fig. 16. Impact of experiment budget on the probability of successful completion of single-cell analysis under the optimal protocol strategy (a) with varying number of washing steps and (b) with varying fixation time.

As shown in Table IV, as the number of output clusters increases, the flexibility *fx* of the crossbar design decreases. On the other hand, as shown in Fig. 15, we observe that the total completion time decreases when the number of clusters increases—this result 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. 11, 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 [45] to implement the DTMC model along with the following PCTL query that expresses the model objective:

$$Q \coloneqq P_{\max=?}\big[(\Box(!Failure))\ U\ (\Diamond\ WithinBudget)\big].$$

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 [46]. 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. 16 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.

## 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 would like to thank Prof. K. Scott from the Department of Molecular Genetics and Microbiology, Duke University for providing the data related to the performance of the ChIP protocol.

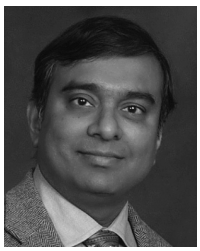
## References

[1] S. Hosic, S. Murthy, and A. Koppes, "Microfluidic sample preparation for single cell analysis," *Anal. Chem.*, vol. 88, no. 1, pp. 354–380, 2015.
[2] T. Q. Vu, R. M. D. Castro, and L. Qin, "Bridging the gap: Microfluidic devices for short and long distance cell–cell communication," *Lab Chip*, vol. 17, no. 6, pp. 1009–1023, 2017.
[3] A. Saadatpour, S. Lai, G. Guo, and G.-G. Yuan, "Single-cell analysis in cancer genomics," *Trends Genet.*, vol. 31, no. 10, pp. 576–586, 2015.
[4] N. Shembekar, C. Chaipan, R. Utharala, and C. A. Merten, "Droplet-based microfluidics in drug discovery, transcriptomics and high-throughput molecular genetics," *Lab Chip*, vol. 16, no. 8, pp. 1314–1331, 2016.
[5] R. Zilionis *et al.*, "Single-cell barcoding and sequencing using droplet microfluidics," *Nat. Protocols*, vol. 12, no. 1, pp. 44–73, 2017.
[6] A. M. Klein *et al.*, "Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells," *Cell*, vol. 161, no. 5, pp. 1187–1201, 2015.
[7] A. Rival *et al.*, "An EWOD-based microfluidic chip for single-cell isolation, mRNA purification and subsequent multiplex qPCR," *Lab Chip*, vol. 14, no. 19, pp. 3739–3749, 2014.
[8] M. J. Jebrail *et al.*, "World-to-digital-microfluidic interface enabling extraction and purification of RNA from human whole blood," *Anal. Chem.*, vol. 86, no. 8, pp. 3856–3862, 2014.
[9] S. C. Shih *et al.*, "A droplet-to-digital (D2D) microfluidic device for single cell assays," *Lab Chip*, vol. 15, no. 1, pp. 225–236, 2015.
[10] L. Arrigoni *et al.*, "Standardizing chromatin research: A simple and universal method for ChIP-seq," *Nucleic Acids Res.*, vol. 44, no. 7, pp. 1–13, 2016.
[11] T. S. Furey, "ChIP-seq and beyond: New and improved methodologies to detect and characterize protein-DNA interactions," *Nat. Rev. Genet.*, vol. 13, no. 12, pp. 840–852, 2012.

[12] R. B. Fair, "Digital microfluidics: Is a true lab-on-a-chip possible?" *Microfluidics Nanofluidics*, vol. 3, no. 3, pp. 245–281, 2007.
[13] T. Thorsen *et al.*, "Microfluidic large-scale integration," *Science*, vol. 298, no. 5593, pp. 580–584, 2002.
[14] F. Su and K. Chakrabarty, "High-level synthesis of digital microfluidic biochips," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 3, no. 4, p. 1, 2008.
[15] O. Keszocze, R. Wille, K. Chakrabarty, and R. Drechsler, "A general and exact routing methodology for digital microfluidic biochips," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2015, pp. 874–881.
[16] T. Xu and K. Chakrabarty, "Broadcast electrode-addressing for pin-constrained multi-functional digital microfluidic biochips," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, Anaheim, CA, USA, 2008, pp. 173–178.
[17] D. T. Grissom and P. Brisk, "Fast online synthesis of digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 3, pp. 356–369, Mar. 2014.
[18] T.-W. Huang, S.-Y. Yeh, and T.-Y. Ho, "A network-flow based pin-count aware routing algorithm for broadcast-addressing EWOD chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 12, pp. 1786–1799, Dec. 2011.
[19] M. Ibrahim, Z. Li, and K. Chakrabarty, "Advances in design automation techniques for digital-microfluidic biochips," in *Formal Modeling and Verification of Cyber-Physical Systems*. Wiesbaden, Germany: Springer, 2015, pp. 190–223.
[20] W. H. Minhass, P. Pop, and J. Madsen, "System-level modeling and synthesis of flow-based microfluidic biochips," in *Proc. CASES*, Taipei, Taiwan, 2011, pp. 225–233.
[21] T.-M. Tseng, M. Li, B. Li, T.-Y. Ho, and U. Schlichtmann, "Columba: Co-layout synthesis for continuous-flow microfluidic biochips," in *Proc. DAC*, Austin, TX, USA, 2016, pp. 1–6.
[22] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer optimization for flow-based mVLSI microfluidic biochips," in *Proc. IEEE/ACM CASES*, 2014, p. 1–10.
[23] M. Alistar, P. Pop, and J. Madsen, "Synthesis of application-specific fault-tolerant digital microfluidic biochip architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 5, pp. 764–777, May 2016.
[24] H. Yao, Q. Wang, Y. Shen, T.-Y. Ho, and Y. Cai, "Integrated functional and washing routing optimization for cross-contamination removal in digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1283–1296, Aug. 2016.
[25] H. Yao, Q. Wang, Y. Ru, Y. Cai, and T.-Y. Ho, "Integrated flow-control codesign methodology for flow-based microfluidic biochips," *IEEE Design Test*, vol. 32, no. 6, pp. 60–68, Dec. 2015.
[26] R. Wille, O. Keszocze, R. Drechsler, T. Boehnisch, and A. Kroker, "Scalable one-pass synthesis for digital microfluidic biochips," *IEEE Design Test*, vol. 32, no. 6, pp. 41–50, Dec. 2015.
[27] Y. Luo, K. Chakrabarty, and T.-Y. Ho, "Error recovery in cyberphysical digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 59–72, Jan. 2013.
[28] M. Alistar *et al.*, "Redundancy optimization for error recovery in digital microfluidic biochips," *Design Autom. Embedded Syst.*, vol. 19, nos. 1–2, pp. 129–159, 2015.
[29] M. Ibrahim and K. Chakrabarty, "Efficient error recovery in cyberphysical digital-microfluidic biochips," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 1, no. 1, pp. 46–58, Jan./Mar. 2015.
[30] S. Poddar, S. Ghoshal, K. Chakrabarty, and B. B. Bhattacharya, "Error-correcting sample preparation with cyberphysical digital microfluidic lab-on-chip," *ACM Trans. Design Autom. Elect. Syst.*, vol. 22, no. 1, p. 2, 2016.
[31] J. Gong *et al.*, "All-electronic droplet generation on-chip with real-time feedback control for EWOD digital microfluidics," *Lab Chip*, vol. 6, no. 6, pp. 898–906, 2008.
[32] Y. Luo, B. B. Bhattacharya, T.-Y. Ho, and K. Chakrabarty, "Design and optimization of a cyberphysical digital-microfluidic biochip for the polymerase chain reaction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 1, pp. 29–42, Jan. 2015.
[33] M. Ibrahim, K. Chakrabarty, and K. Scott, "Synthesis of cyberphysical digital-microfluidic biochips for real-time quantitative analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 5, pp. 733–746, May 2017.
[34] M. Ibrahim, C. Boswell, K. Chakrabarty, K. Scott, and M. Pajic, "A real-time digital-microfluidic platform for epigenetics," in *Proc. IEEE/ACM CASES*, Pittsburgh, PA, USA, 2016, pp. 1–10.
[35] M. Ibrahim, A. Sridhar, K. Chakrabarty, and U. Schlichtmann, "Synthesis of reconfigurable flow-based biochips for scalable single-cell screening," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2017, pp. 623–630.
[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 Lett.*, vol. 89, no. 3, 2010, Art. no. 38009.
[37] C. Baier *et al.*, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.
[38] M. Kwiatkowska, "Quantitative verification: Models techniques and tools," in *Proc. 6th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. (ESEC/FSE)*, 2007, pp. 449–458.
[39] M. A. Murran and H. Najjaran, "Capacitance-based droplet position estimator for digital microfluidic devices," *Lab Chip*, vol. 12, pp. 2053–2059, Mar. 2012.
[40] M. Ibrahim, K. Chakrabarty, and K. Scott, "Integrated and real-time quantitative analysis using cyberphysical digital-microfluidic biochips," in *Proc. DATE*, 2016, pp. 630–635.
[41] R. Silva *et al.*, "A reconfigurable continuous-flow fluidic routing fabric using a modular, scalable primitive," *Lab Chip*, vol. 16, pp. 2730–2741, Jun. 2016.
[42] M. Ibrahim *et al.*, "Synthesis of a cyberphysical hybrid microfluidic platform for single-cell analysis," Duke University, Durham, NC, USA, Rep. 10161/15637, 2017. [Online]. Available: http://hdl.handle.net/10161/15637
[43] T. Cormen *et al.*, *Introduction to Algorithms*. Cambridge, U.K.: MIT Press, 2001.
[44] M. Kwiatkowska and D. Parker, "Automated verification and strategy synthesis for probabilistic systems," in *Proc. Int. Symp. Autom. Technol. Verification Anal.*, 2013, pp. 5–22.
[45] M. Kwiatkowska *et al.*, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*. Heidelberg, Germany: Springer, 2011, pp. 585–591.
[46] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis, "Automatic verification of competitive stochastic systems," *Formal Methods Syst. Design*, vol. 43, no. 1, pp. 61–92, 2013.

**Mohamed Ibrahim** (S'13) received the M.Sc. degree and the Ph.D. degree in electrical and computer engineering from Duke University, Durham, NC, USA, in 2017 and 2018, respectively.

His current research interests include electronic design automation of lab-on-a-chip systems, security and trust of emerging technologies, and artificial intelligence applications of advanced microfluidic systems.



**Krishnendu Chakrabarty** (F'08) received the B.Tech. degree from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, MI, USA, in 1992 and 1995, respectively.

He is the William H. Younger Distinguished Professor and the Department Chair of Electrical and Computer Engineering with Duke University, Durham, NC, USA. His current research interest include design-for-testability of integrated circuits and systems, microfluidic biochips, data analytics for failure prediction, anomaly detection and hardware security, and neuromorphic computing systems.

**Ulf Schlichtmann** (M'90) received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering and information technology from the Technical University of Munich (TUM), Munich, Germany, in 1990 and 1995, respectively.

He is a Professor and the Head of the Chair of Electronic Design Automation with the TUM, Munich, Germany. In 2003, he joined TUM, following ten years in industry. His current research interests include computer-aided design of electronic circuits and systems, with an emphasis on designing reliable and robust systems, and emerging technologies, such as lab-on-a-chip and photonics.