CoSyn: Efficient Single-Cell Analysis Using a Hybrid Microfluidic Platform

Mohamed Ibrahim[†], Krishnendu Chakrabarty[†], and Ulf Schlichtmann[‡]
[†]Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA
[‡]Institute for Electronic Design Automation, Technische Universität München, Arcisstraße 21, 80333 München, Germany

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. 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. Simulation results show that CoSyn efficiently utilizes platform resources and outperforms baseline techniques.**

I. INTRODUCTION

Single-cell analysis using affordable microfluidic technologies has now become a reality [1]. 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 [2]. 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 [3].

(2) Droplet Indexing (Barcoding): Each droplet is manipulated through a sequence of biochemical procedures such as cell lysis and mRNA analysis. Indexing of droplets using barcodes is needed to keep track of their identity.

(3) Type-Driven Cell Analysis: 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 [4].

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) [5] (Step 3). However, DMFBs are not as effective for interfacing to the external world [6]. 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 [7], 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 [4] (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 [8], it is unrealistic to assume that each dispensed droplet contains only one barcoding particle.

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. 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 and evaluate CoSyn, which enables coordinated control of the microfluidic components, and allows dual-domain synthesis for concurrent sample pathways.

The rest of the paper is organized as follows. Section II describes related prior work. An overview of the hybrid microfluidic platform and its use for single-cell analysis are presented in Section III. Next, we present the single-cell analysis flow (Section IV) and describe CoSyn (Section V). Our experimental evaluation is presented in Section VI and conclusions are drawn in Section VII.

II. RELATED PRIOR WORK

A DMFB manipulates picoliter droplets, and consists of a two-dimensional array of electrodes and a set of on-chip resources [9]. Valve-based biochips, on the other hand, rely

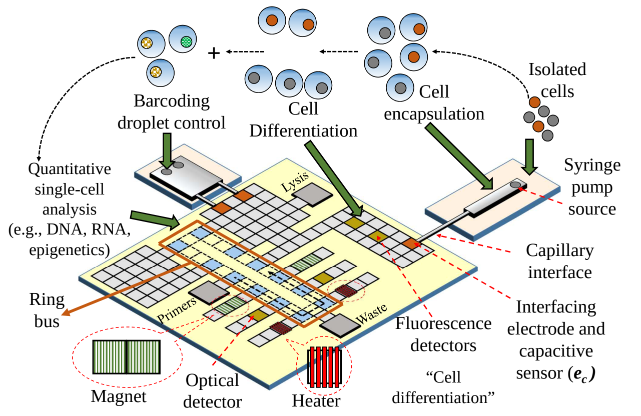

Fig. 1: The hybrid platform for single-cell analysis.

on special-purpose components (e.g., microvalves and micropumps) to manipulate liquid flow [10].

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, and sharing of control pins (or pressure sources) [11]–[14]. 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 [15], [16], volume precision [17], termination of biochemical applications such as qPCR [18], and protocols for multiple samples [19]. 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.

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 [8].

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 [8]. 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 [20]. When the digital array is unable to accommodate additional droplets, it stops the flow by switching off the pump.

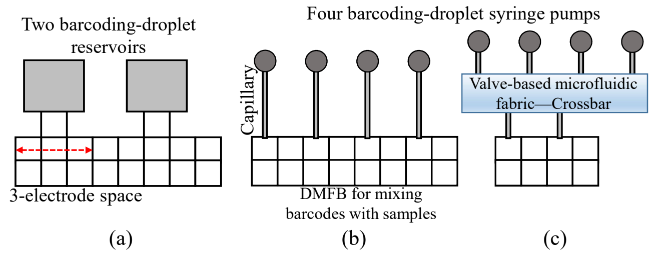
Two barcoding-droplet reservoirs
3-electrode space
(a)
Four barcoding-droplet syringe pumps
Capillary
DMFB for mixing barcodes with samples
(b)
Valve-based microfluidic fabric—Crossbar
(c)

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.

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 [19], 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 in CoSyn due to its demonstrated ability to carry out high-throughput fluorescence detection and distinguish between hundreds of cell types [21]; 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 [4]. 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 [4].

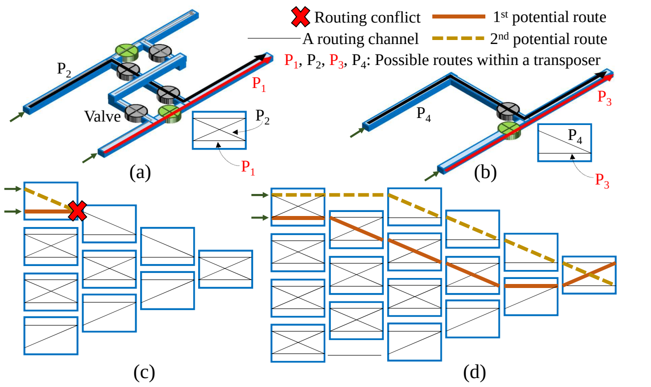

Fig. 3: A valve-based routing fabric for droplet barcoding: (a) a 2-input full transposer; (b) a 2-input half transposer; (c) a 4-level, 8-to-2 routing fabric; (d) a 6-level, 8-to-2 fabric.

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 [22]; 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 $n >> m$. 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 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 [22]. 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. Due to lack of space, we skip the architectural design challenge; we focus instead on the modeling and synthesis problems listed in Table I.

TABLE I: Design Problems for Assembling and Integrating a 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.

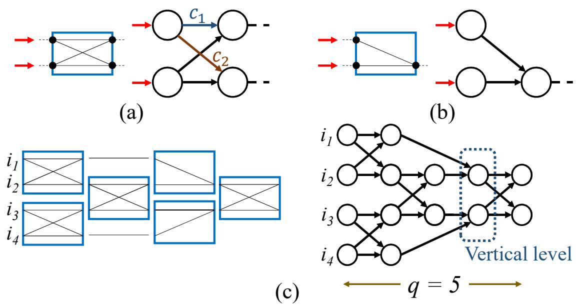

Fig. 4: Mapping a valve-based crossbar to a graph model: (a) a full transposer; (b) a half transposer; (c) a 4-to-2 crossbar.

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 gene-expression analysis using qPCR [5], [19], 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 [23].

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 [22]. Fig. 4(c) depicts the graph $\mathcal{F}_{4\times2}$ 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 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 [19], 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

A. *Problem Formulation*

Our goal is to design a fully connected fabric such that a droplet can be forwarded from any of the n inputs to any

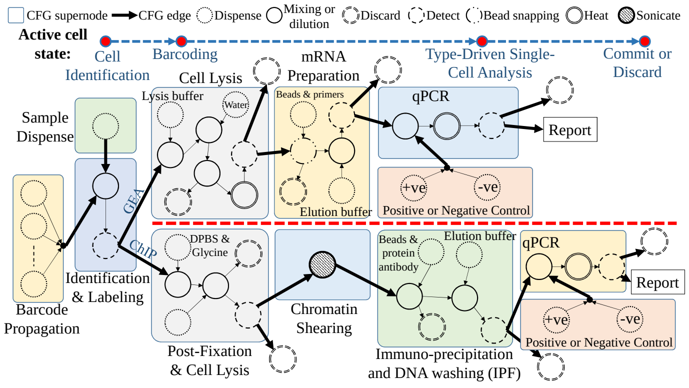

Fig. 5: CFG of type-driven analysis for a single cell pathway.

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. 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 and total completion time T_{comp}.

Objective: Minimize T_{comp} to provide high throughput.

B. *Solution Methods*

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. 6 illustrates pipelined and non-pipelined routing.

To solve the routing problem, we utilize a graph-theoretic algorithm to find vertex-disjoint shortest paths [24]; see Algorithm 1. 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. 6). 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

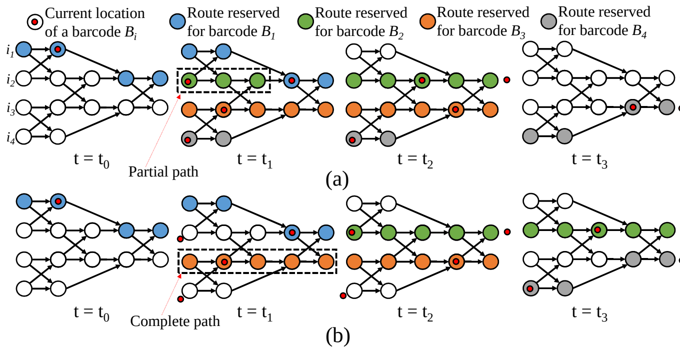
Current location of a barcode B_i
Route reserved for barcode B_1
Route reserved for barcode B_2
Route reserved for barcode B_3
Route reserved for barcode B_4
i_1
i_2
i_3
i_4
$t = t_0$
$t = t_1$
$t = t_2$
$t = t_3$
Partial path
(a)
Complete path
(b)

Fig. 6: Valve-based routing of 4 barcoding droplets (4-to-2 biochip): (a) with pipelining; (b) without pipelining.

```
Algorithm 1: 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};
```

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 $O(|\mathcal{D}_{n\times m}| + |\mathcal{S}_{n\times m}|)$.

We use a greedy method to solve the resource-allocation problem in the DMFB; the pseudocode is shown in Algorithm 2. 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 $O(|\mathcal{V}|)$.

CoSyn uses a time-wheel to simulate the real-time interactions between the individual cells and the components of the hybrid system. The stages of the pipeline match the states of the cell state machine (Fig. 5). The time-wheel interacts with both microfluidic domains through APIs. Whenever the time-wheel locates an available fluorescence detector at the DMFB, it allocates a cell to it in order to perform type identification. Next, when the cell type is identified and there are available valve-based routes (partial or complete) to route the associated barcoding droplet, the time-wheel starts the pipelined routing process through iterations until the droplet reaches the electrode interface at the digital-microfluidic side and mixed with the cell. When a DMFB resource is available to further process the cell, the previously reserved valve-based channels are released.

Real-time resource allocation for the DMFB is managed by the time-wheel, which in turn, commits a cell pathway

```
Algorithm 2: DMFB Resource Allocation
Input: $C_i$, $\mathcal{G}$, current simulation time "$t$"
Output: Assigned Resource "$r$"
1: $\tilde{R} \leftarrow$ GetCurrentlyUnoccupiedResources($\mathcal{G}$, $t$);
2: if ($\tilde{R}$ is empty) then return NULL;
3: $y \leftarrow$ GetOperationType($C_i$);
4: $s \leftarrow$ CalculateMinimumCostAllAvailableResources($y$, $\tilde{R}$);
5: if ($s = \infty$) then return NULL; // No suitable resource
6: $r \leftarrow$ GetSelectedResource($s$); return $r$;
```

whenever its particular single-cell bioassays have executed. Based on an intermediate decision point, the cell might also be discarded during analysis.

VI. Simulation Results

We implemented CoSyn using C++. All evaluations were carried out using a 3.4 GHz Intel i3 CPU with 4 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.

A. Performance Evaluation

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). 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. 7(a)-(b) 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. Design-Quality Assessment

We also evaluate the quality of the designs generated by CoSyn in terms of the number of single-cell experiments

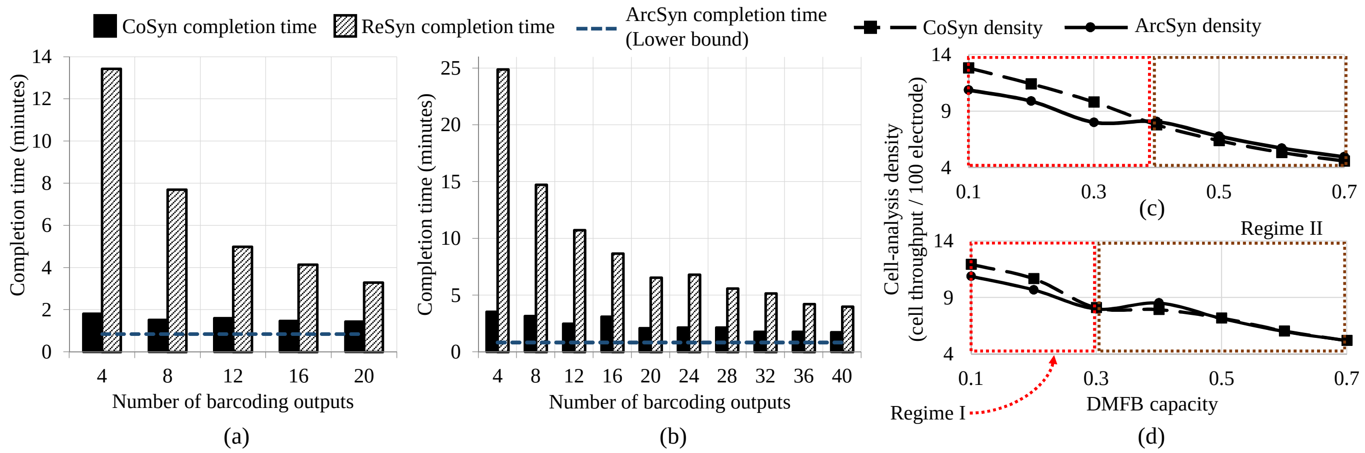

Fig. 7: Comparison between CoSyn, ArcSyn, and ReSyn: (a) completion time (T) using 20 barcoding inputs, (b) T for 40 barcoding inputs, (c) cell-analysis density (D) using 4 barcoding outputs, (d) D using 8 barcoding outputs.

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 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 and ArcSyn. We simulate the execution of 50 cells using 4 barcoding outputs (Fig. 7(c)) and 8 barcoding outputs (Fig. 7(d)). The density values are computed while the capacity is varied. By comparing the density values for CoSyn 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 is less than or equal to the density of ArcSyn. Regime I highlights the fact that CoSyn 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. This study has been confirmed using different settings corresponding to the number of cells, the number of barcoding inputs and outputs; see Appendix C.

VII. 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. The proposed platform has been evaluated on the basis of the time needed for analysis and the biochip size needed for realistic test cases.

References

[1] S. Hosic *et al.*, “Microfluidic sample preparation for single cell analysis,” *Analytical Chemistry*, vol. 88, no. 1, pp. 354–380, 2015.
[2] A. Saadatpour *et al.*, “Single-cell analysis in cancer genomics,” *Trends in Genetics*, vol. 31, no. 10, pp. 576–586, 2015.
[3] 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.
[4] A. M. Klein *et al.*, “Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells,” *Cell*, pp. 1187–1201, 2015.
[5] 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.
[6] 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.
[7] BioNumbers. Number of cell types in human body. [Online] http://bionumbers.hms.harvard.edu/bionumber.aspx?id=103626.
[8] 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.
[9] R. B. Fair, “Digital microfluidics: is a true lab-on-a-chip possible?” *Microfluidics and Nanofluidics*, vol. 3, no. 3, pp. 245–281, 2007.
[10] T. Thorsen *et al.*, “Microfluidic large-scale integration,” *Science*, vol. 298, no. 5593, pp. 580–584, 2002.
[11] F. Su and K. Chakrabarty, “High-level synthesis of digital microfluidic biochips,” *ACM JETC*, vol. 3, no. 4, p. 1, 2008.
[12] D. T. Grissom and P. Brisk, “Fast online synthesis of digital microfluidic biochips,” *IEEE TCAD*, vol. 33, no. 3, pp. 356–369, 2014.
[13] W. H. Minhass *et al.*, “System-level modeling and synthesis of flow-based microfluidic biochips,” in *Proc. CASES*, 2011, pp. 225–233.
[14] T.-M. Tseng *et al.*, “Columba: co-layout synthesis for continuous-flow microfluidic biochips,” in *Proc. DAC*, 2016.
[15] Y. Luo *et al.*, “Error recovery in cyberphysical digital microfluidic biochips,” *IEEE TCAD*, vol. 32, no. 1, pp. 59–72, 2013.
[16] P. Pop *et al.*, “Compilation for error recovery,” in *Fault-Tolerant Digital Microfluidic Biochips*. Springer, 2016, pp. 145–174.
[17] J. Gong *et al.*, “All-electronic droplet generation on-chip with real-time feedback control for EWOD digital microfluidics,” *Lab on a Chip*, 2008.
[18] 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.

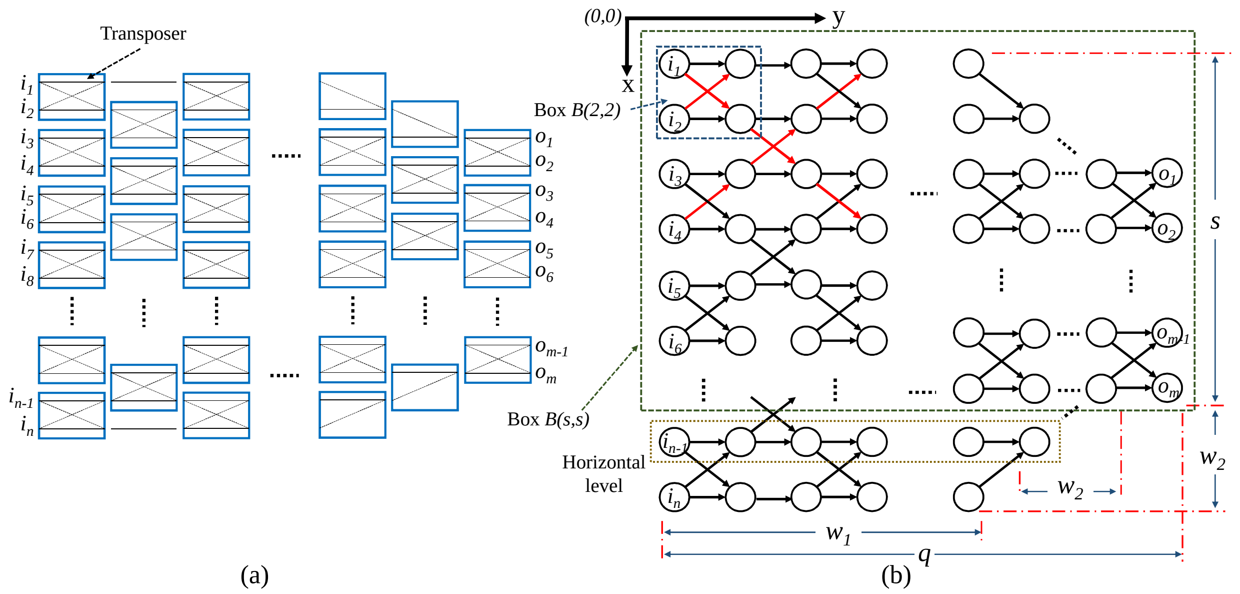

Fig. 8: (a) A schematic view of an n-to-m routing crossbar; (b) the associated graph model $\mathcal{F}_{(n\times m,\mathcal{T})}$.

[19] M. Ibrahim *et al.*, "Integrated and real-time quantitative analysis using cyberphysical digital-microfluidic biochips," in *Proc. DATE*, 2016.
[20] M. A. Murran and H. Najjaran, "Capacitance-based droplet position estimator for digital microfluidic devices," *Lab on a Chip*, 2012.
[21] A. Ng *et al.*, "Digital microfluidic immunocytochemistry in single cells." *Nature communications*, vol. 6, pp. 7513–7513, 2014.
[22] R. Silva *et al.*, "A reconfigurable continuous-flow fluidic routing fabric using a modular, scalable primitive," *Lab on a Chip*, 2016.
[23] A. R. Wu *et al.*, "High throughput automated chromatin immunoprecipitation as a platform for drug screening and antibody validation," *Lab on a Chip*, vol. 12, no. 12, pp. 2190–2198, 2012.
[24] T. Cormen *et al.*, *Introduction to Algorithms*. MIT Press Cambridge, 2001.

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. 8(a). The associated graph model $\mathcal{F}_{(n\times m,\mathcal{T})}$ (defined in the paper) is depicted in Fig. 8(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 $i_k \in \{i_1, i_2, ..., i_n\}$ can reach any output $o_j \in \{o_1, o_2, ..., o_m\}$.*

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. 8(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. 9. 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. 8(b). Based on the reconfigurable design of the transposer (Section III in the paper), we introduce the following lemma:

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. □

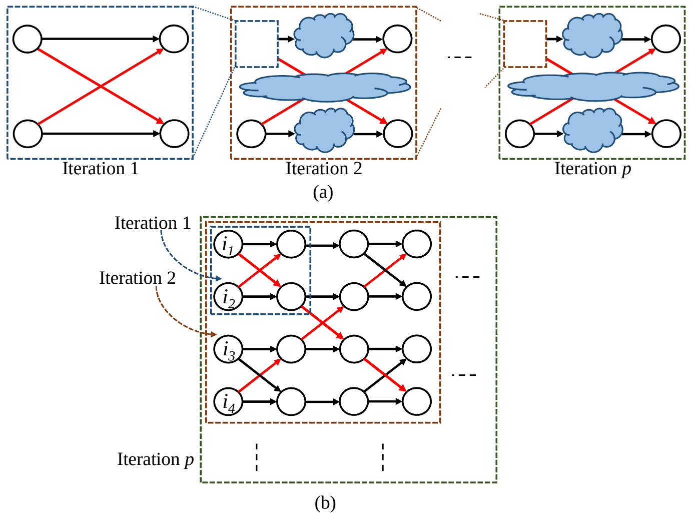

Fig. 9: Iterative construction of an $\mathcal{F}_{n\times m}$.

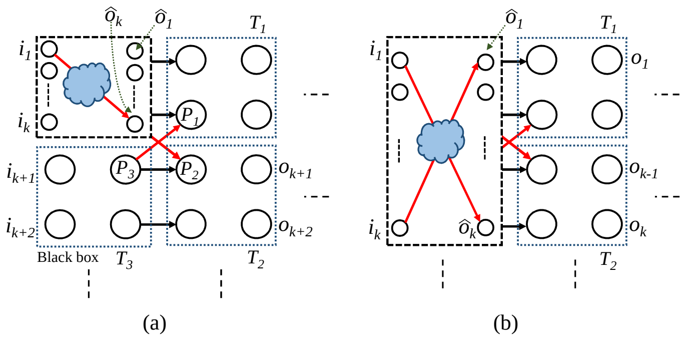

Fig. 10: Expanding the graph model $\mathcal{F}_{k\times k}$ to (a) $\mathcal{F}_{(k+2)\times(k+2)}$, (b) $\mathcal{F}_{k\times(k+2)}$.

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. 10(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. 10(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. $\square$

Furthermore, consider expanding $\mathcal{F}_{k\times k}$ to $\mathcal{F}_{k\times(k+2)}$ following the same approach; see Fig. 10(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. $\square$

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. 8(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. 8(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. 8(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, i.e., $q = \frac{m+n}{2}$.

$$w_2 = \frac{n-m}{2} \tag{1}$$

$$\begin{aligned} w_1 &= q - w_2 - 1 \\ &= \frac{m+n}{2} - \frac{n-m}{2} - 1 \\ &= m - 1 \end{aligned} \tag{2}$$

Algorithm 3 is implemented to generate the graph model $\mathcal{F}_{n\times m}$ of a fully connected routing crossbar. For the sake of

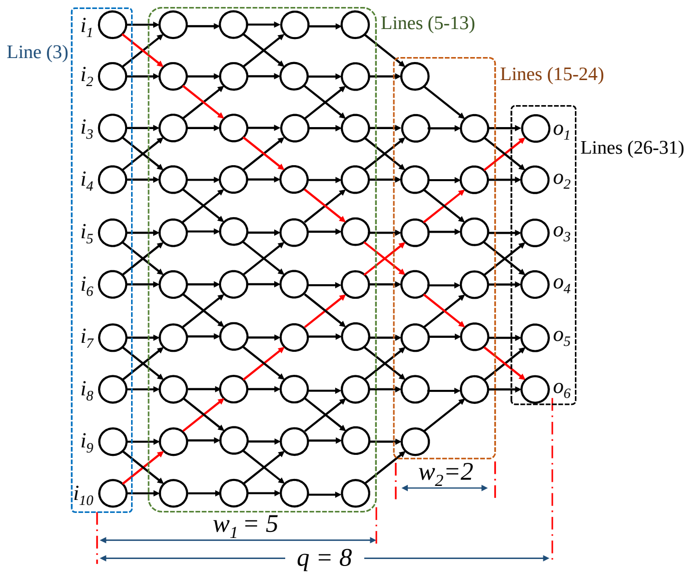

Fig. 11: The application of Algorithm 3 on a graph $\mathcal{F}_{10\times6}$.

illustration, we apply the steps of the algorithm to generate the graph $\mathcal{F}_{10\times6}$; see Fig. 11. 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 $\mathcal{F}_{10\times6}$. Then the algorithm generates the input nodes which reside at the first vertical level of $\mathcal{F}_{10\times6}$ (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 $O(n^2 \cdot m)$.

Appendix C
More Results on Design-Quality

We present additional results to evaluate the quality of the designs generated by CoSyn. Our objective is to confirm that the values of cell-analysis density follows a two-regime profile, as explained in the paper. Here, we show simulation results based on the configurations shown in Table II. For each case, we plot cell-analysis density while the number of barcoding outputs and DMFB capacity (defined in the paper) are varied.

TABLE II: Configurations of the simulation results

	# Cells	# Barcoding inputs
Case I	20	10
Case II	50	20

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: $w_1, w_2 \leftarrow$ Initialize($n, m, q$);
3: $\mathcal{D}_{n\times m} \leftarrow$ 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:         $\tau \leftarrow$ GenerateNode($v_l$,$h_l$);
9:         $e \leftarrow$ 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:         $\tau \leftarrow$ GenerateNode($v_l$,$h_l$);
19:         $e \leftarrow$ 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:     $\tau \leftarrow$ GenerateOutputNodes($h_l$);
28:     $e \leftarrow$ 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 $\mathcal{D}_{n\times m}$, $\mathcal{S}_{n\times m}$;
```

A. Case I:

Fig. 12 depicts the values of cell-analysis density when the number of cells is 20 and the number of barcoding inputs is 10. For each one of the three plots (Fig. 12(a)-(c)), we investigate cell-analysis density while DMFB capacity is varied. We observe the two-regime profile in Fig. 12(a)-(b), whereas Regime I disappears in Fig. 12(c). This is because when the number of barcoding outputs is 10, the architecture of CoSyn becomes similar to that of ArcSyn (i.e., number of inputs = number of outputs = 10), while ArcSyn is valveless. As a result, CoSyn does not benefit from pipelined routing in this case to increase cell-analysis density.

B. Case II:

Fig. 13 illustrates the values of cell-analysis density when the number of cells is 50 and the number of barcoding inputs is 20. It is clear that Regime I shrinks as we increase the number of barcoding outputs from 4 outputs (Fig. 13(a)) to 20 outputs (Fig. 13(e)).

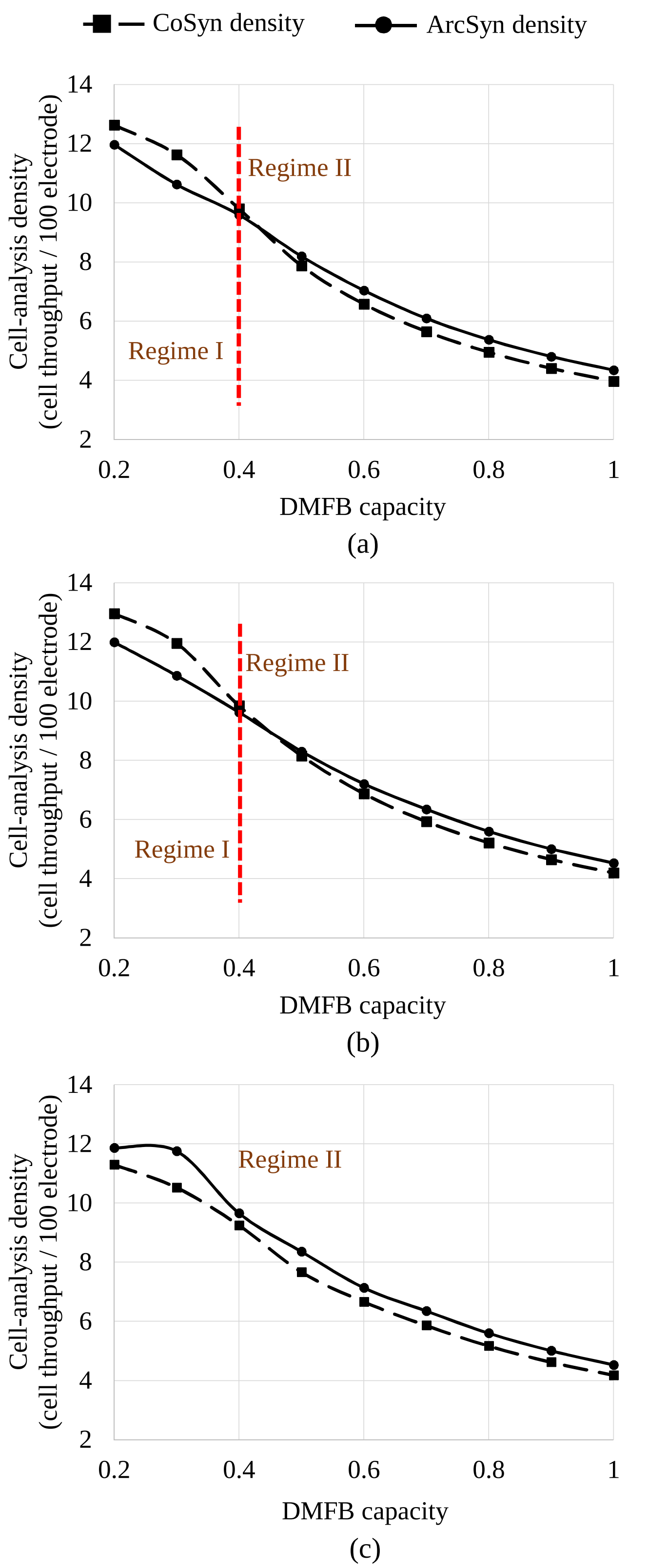

Fig. 12: (Case I) Comparison between CoSyn and ArcSyn when 20 cells are executed and 10 barcoding inputs are used: (a) cell-analysis density using 2 barcoding outputs, (b) cell-analysis density using 6 barcoding outputs, (c) cell-analysis density using 10 barcoding outputs.

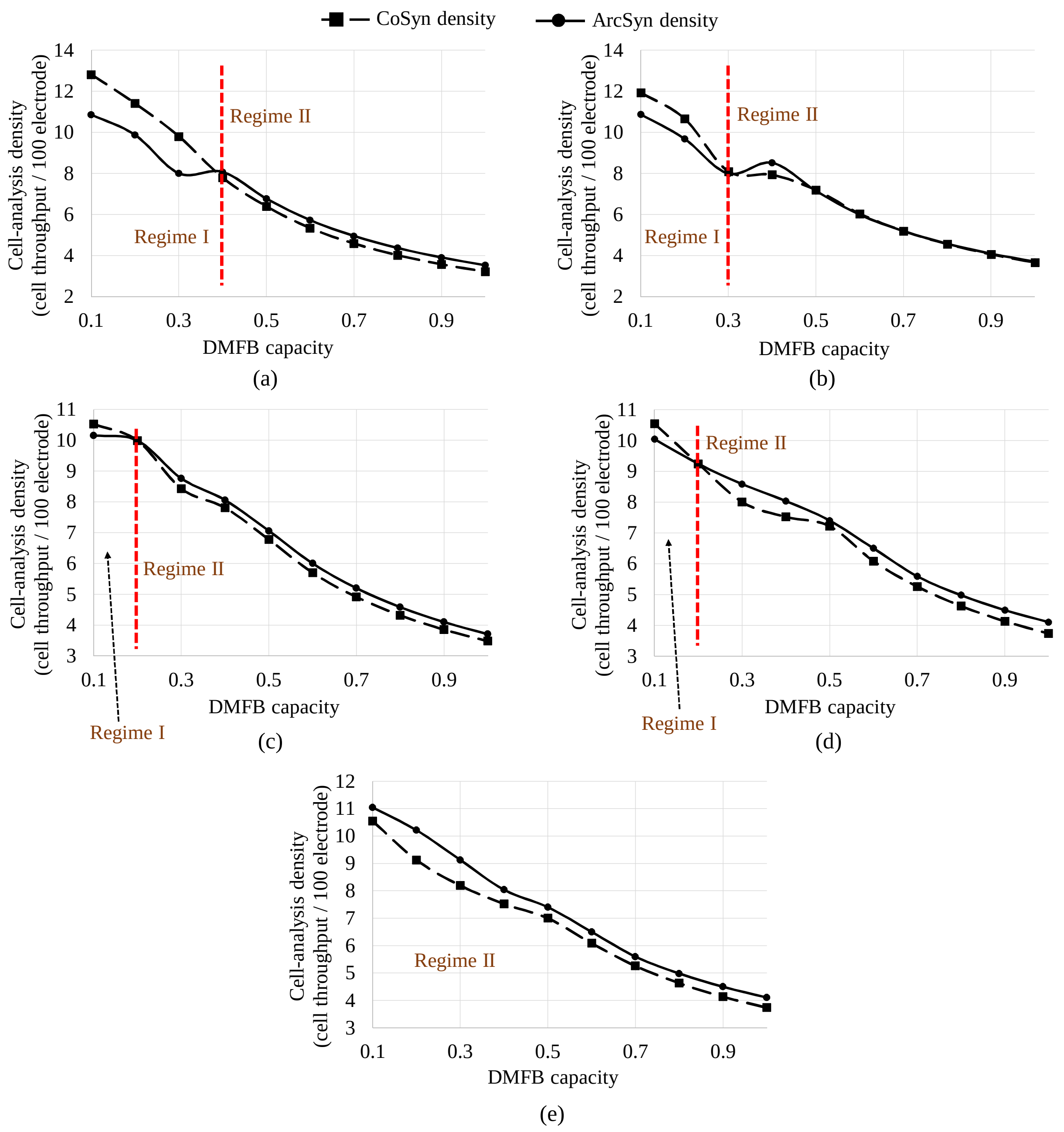

Fig. 13: (Case II) Comparison between CoSyn and ArcSyn when 50 cells are executed and 20 barcoding inputs are used: (a) cell-analysis density using 4 barcoding outputs, (b) cell-analysis density using 8 barcoding outputs, (c) cell-analysis density using 12 barcoding outputs, (d) cell-analysis density using 16 barcoding outputs, (e) cell-analysis density using 20 barcoding outputs.