Component-Oriented High-level Synthesis for Continuous-Flow Microfluidics Considering Hybrid-Scheduling

Mengchu Li[†▽], Tsun-Ming Tseng[†], Bing Li[†], Tsung-Yi Ho[⋆◇], and Ulf Schlichtmann[†]
mengchu.li@campus.lmu.de, {tsun-ming.tseng, b.li, ulf.schlichtmann}@tum.de, tyho@cs.nthu.edu.tw
[†]Institute for Electronic Design Automation, Technical University of Munich, Arcisstraße 21, 80333 München, Germany
[▽]Ludwig-Maximilians-Universität München, Geschwister-Scholl-Platz 1, 80539 München, Germany
[⋆]Department of Computer Science, National Tsing Hua University, No. 101, Section 2, Kuang-Fu Road, 30013 Hsinchu, Taiwan
[◇]Institute for Advanced Study, Technical University of Munich, Lichtenbergstrasse 2 a, 85748 Garching, Germany

ABSTRACT

Technological innovations in continuous-flow microfluidics require updated automated synthesis methods. As new microfluidic components and biochemical applications are constantly introduced, the current functionality-based application mapping methods and the fixed-time-slot scheduling methods are insufficient to solve the new design challenges. In this work, we propose a component-oriented general device concept that enables precise description of operations and devices, and adapts well to technological updates. Applying this concept, we propose a layering algorithm together with a mathematical modeling method to synthesize binding and hybrid-scheduling solutions that support both fixed schedule and real-time decisions. We also consider potential chip layout and optimize the number of flow channels among devices to save routing efforts. Experimental results demonstrate that our solution fully utilizes the chip resources and can handle operations with different requirements.

1 Introduction

Continuous-flow microfluidic biochips are the mainstream microfluidic technology for cell-based applications including cell sorting, single-cell analysis, and DNA amplification. The rapid development of lab-on-a-chip technology involves lots of design efforts and thus results in increasing demand for design automation.

Most continuous-flow microfluidics comprises a combinable set of devices, which enables sophisticated bioassays to be performed on a single chip within mature fabrication technology [1]. An early automatic synthesis work [2] therefore proposed a fluidic instruction set, where operations and devices were classified into several types according to their functionality, such as mixing, heating, detecting, etc., and the execution duration of operations was specified by exact values. Also, operations could only be mapped to a device if their types matched exactly. These specifications simplified the scheduling and binding process and provided a basis for later synthesis work [3, 4, 5, 6], which made it become the accepted standard.

However, as lab-on-a-chip technology evolves, new microfluidic components are constantly introduced, the integration of which broadens the bioassay scope and challenges the conventional automatic synthesis standard.

DAC 2017, DOI: 10.1145/3061639.3062213
http://ieeexplore.ieee.org/document/8060423/
https://dl.acm.org/citation.cfm?id=3062213

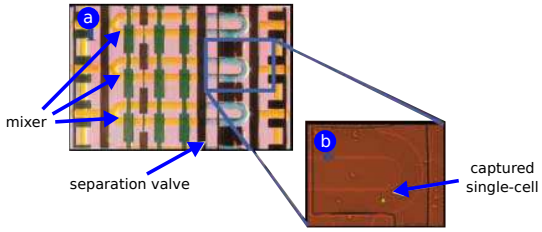

Figure 1: Micrograph of a chip designed for a gene expression profiling assay [7]. (a) Mixers integrated with cell-separation modules. (b) Cell separation module.

Mixers, as one of the most important microfluidic devices, are evolving. A mixer typically consists of a ring-shaped channel segment and a peristaltic pump formed by valves. Figure 1(a) shows a micrograph of a chip with three mixers [7], where pumps are marked by green color. Unlike conventional designs, the flow channels of the mixers are separated by black separation valves into two parts. When the separation valves are open, the mixers serve as normal mixers for mixing operations. But when the separation valves are closed, the blue U-shaped parts of the flow channels are separated from the original mixers and become cell-separation modules for cell-isolation operations as shown in Figure 1(b). Both the mixing and the cell-isolation operations monopolize the ring-shaped flow-channel segments during their execution. In other word, cell-isolation operations are bound to mixers in spite of the conventional type-matching rules.

Besides cell-separation modules, current designs demonstrate that mixers can easily be integrated with several other microfluidic components and therefore be utilized by numerous types of operations, such as detecting [7], heating [8], washing [9], etc. Meanwhile, mixing operations are not necessarily executed in mixers.

For example, Figure 2(a) shows a micrograph of a chip designed for a kinase activity assay, where mixing operations are executed in the blue flow-channel segments controlled by green sieve valves. Sieve valves are important microfluidic components for cell-based operations. When a sieve valve is closed, it leaves a narrow gap so that large particles are blocked but small particles and fluids are allowed to flow through. In this design, sieve valves are integrated to form bead columns, which will be mixed with a large amount of input samples to capture target peptide substrate. Because of the large input volume, this operation can hardly be supported by a conventional fixed-volume mixer. However, as shown in Figure 2(b)-(e), since beads are blocked by the sieve valves while fluids are not, by performing a flow reversal protocol, input samples can easily pass through the bead column in both directions. Beads are pushed from one end to the other iteratively, and thus efficient mixing of beads and samples is enabled without a classical mixer.

The dividing lines in the type-classification standards have

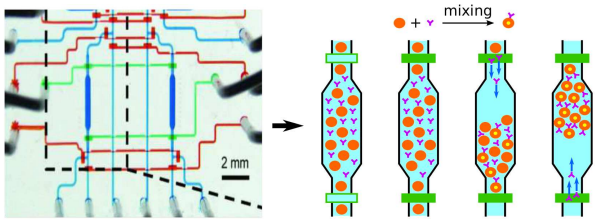

Figure 2: (a) Micrograph of a chip designed for a kinase activity assay [10]. (b)-(e) Flow reversal protocol enables liquid samples to pass through the bead column in both directions alternatively.

become increasingly blurred over time. A single device may support various types of operations. Also, operations with similar functionality may require totally different devices for execution. Therefore, more precise descriptions for operations and devices, as well as more flexible operation-device mapping methods are needed.

Besides, conventional scheduling methods are also challenged by the fact that the execution duration of some operations cannot be specified as exact values. For example, in single-cell capturing operations, the chance that a cell trap captures exactly one cell is about 53% [11]. Therefore, most of the time it is necessary to check the number of cells. In [12], cells can be detected by fluorescent signals. When a signal comes, an image will be taken and analyzed to count the number of cells. If the number is not equal to one, this cell capturing operation needs to be rerun. Therefore, the exact duration of this operation cannot be confirmed until its completion, and it is impossible to allocate this operation to any fixed time slot in the scheduling results.

Cyberphysical integration has recently been considered for digital microfluidics to deal with this indeterminacy by making real-time decisions [13]. However, merely depending on real-time decisions can be time-consuming if there is a large number of operations, especially when manual observation is involved, which is common in continuous-flow microfluidic bioassay protocols. Besides, some operations require precise time control, such as heating in quantitative polymerase chain reaction (qPCR) applications [14]. For these operations, a pre-generated schedule is important for resource reservation. Therefore, a hybrid-scheduling method that supports both pre-generated schedule and real-time decisions is indispensable.

Another indeterminacy in automatic scheduling comes from the reagent transportation time for sequential operations. Reagent transportation time is closely related to the lengths of flow channels, which are determined by the chip physical layout. However, previous work demonstrates that the layout-generation should be carried out after the high-level synthesis process to achieve optimal layout-solutions [4, 15, 16], which means that it is difficult to predict the transportation time during scheduling.

In this work, we propose a component-oriented *general device* concept that enables precise description of operations and devices, and adapts well to technological updates. For large assays involving operations with indeterminate execution duration, we propose a layering algorithm together with a mathematical modeling method to generate binding and hybrid-scheduling solutions.

Our contribution includes:

I We briefly review important microfluidic components, some of which are discussed here for the first time in the design automation field. Based on this review, we propose a component-oriented *general device* concept, which can easily be extended and thus adapted to continuous biological innovations.

II We propose a layering algorithm for large bioassays involving operations with indeterminate execution duration to support cyberphysical integration. For each layer, we propose a mathematical modeling method to generate scheduling and binding results. The results from different layers are iteratively refined by a progressive re-synthesis process to achieve a comprehensive optimization.

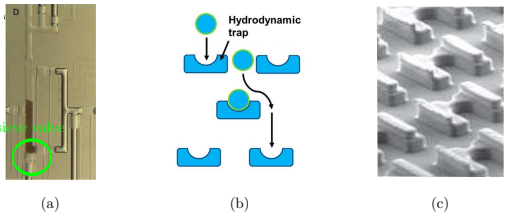

Figure 3: (a) Sieve valve [20]. (b) Single-cell isolation with cell trap [21]. (c) U-shaped cell trap [22].

III We take potential chip layout into consideration and optimize the number of flow channels among devices to save routing efforts. We also estimate reagent transportation time according to the synthesis results from the previous iteration.

2 Component-oriented Synthesis Concept

2.1 Microfluidic Components

Instead of naming microfluidic devices according to their functionality, we take a closer look at the microfluidic components that constitute these devices. We start with a brief review of important microfluidic components on continuous-flow biochips, some of which have never been discussed in previous design automation work. Based on the area cost and processing cost of integrating these components in a chip, we classify them into two categories: *containers*, and *accessories*.

2.1.1 Containers

Containers are microfluidic components, the integration of which requires both exclusive chip areas and processing costs.

Chamber is a segment of a flow channel separated by two valves. Chambers can vary in length and width according to different operation protocols. Diverse operations can be performed in chambers, such as mixing [17], amplification [18], heating [7], neutralization [12], and cell culturing [19].

Ring is a specialized chamber which is connected end to end and thus enables circulation flow. It is mainly used to perform highly efficient mixing operations.

2.1.2 Accessories

Accessories are microfluidic components with functional specialization. They can be integrated into containers and thus require no area cost. However, the integration of accessories involves additional processing costs, including mask fabrication, yield loss, testing costs, as well as the implementation of extra chip ports and control channels.

Pump is a group of valves providing pressure for fluid movement. Each valve can be assigned to an individual pressure source or sequentially connected with other valves driven by the same pressure source.

Heating pad consists of a heating layer and a heating circuit, and is usually integrated under the flow layer to support heating operations.

Optical system is a general term that refers to detection components consisting of a light source and a receiver (detector).

Sieve valve is a specialized valve as shown in Figure 3(a), which leaves a gap when it is closed. A closed sieve valve can halt large particles while allowing small particles and fluids to flow, and thus supports *washing operations* that increase sample concentration by forming solid-phase support [7].

Cell trap is a passive microfluidic component used to capture single cells. It has not been discussed in previous design automation work. There are two major single cell isolation methods on continuous-flow microfluidic biochips: one is to adjust cell concentration and flow rate to obtain an optimal distance between two floating cells, so that single cells can be captured in cell-separation modules as mentioned in Section 1; the other is to apply cell traps that can fit and hold single cells as shown in Figure 3(b), which allows a large number of cell isolation operations to be operated in parallel [21]. Cell traps vary in shapes and sizes: some U-shaped Polydimethylsiloxane (PDMS) traps are shown in Figure 3(c).

2.2 General Device and Component-oriented Operation Definition

Based on the above catagories of microfluidic components, instead of building fences between devices and distributing them to dedicated types, we formulate a *general device* concept to describe devices that are adaptable for various microfluidic integration.

A general device is a general platform for operation execution. It consists of one container and a certain number of accessories. All kinds of microfluidic devices can fit into this concept. For example, a conventional rotary mixer can be regarded as a general device with a ring as its container and a pump as its accessory; and the flow-channel segment mentioned in Figure 2 can be regarded as a general device with a chamber as its container and sieve valves as its accessories.

Similarly, instead of classifying biological operations into different types, we introduce a component-oriented definition method to accurately describe the characteristics of operations.

The definition of a component-oriented operation shall include the following attributes:

a. container (with specified capacity) and accessories required for execution;

b. execution duration, which can be an accurate value, or specified as indeterminate with a minimum duration;

c. dependency: if an operation o_c takes the outputs of another operation o_p as its inputs, we specify o_c as the *child operation* of o_p, and o_p as the *parent operation* of o_c.

According to this definition, an operation is allowed to be bound to a device, if their containers match with each other and the device includes the accessories required by the operation. If the container type of an operation is not specified, it can be bound to either a ring or a chamber of corresponding size.

Our component-oriented synthesis concept allows operations of different types to be bound to the same microfluidic device, which promotes the utilization of on-chip resources, and adapts better to current fabrication technology.

3 Layering for Hybrid-Scheduling

Operations with indeterminate execution duration cannot be allocated to fixed time slots in a schedule. However, a pre-generated schedule is indispensable in many applications for device reservation. In this section, we propose a hybrid synthesis method that not only provides pre-generated schedules, but also supports real-time decisions.

In the rest of this paper, we call operations with indeterminate duration *indeterminate operations* for short.

3.1 Layering Algorithm

To support real-time control for indeterminate operations, we divide an assay schedule into sequential sub-schedules, by distributing operations in a complex assay into $n \in \mathbb{N}$ sequential layers. Each layer (except for the n-th) contains at least one indeterminate operation, and all indeterminate operations are allocated to the end of their sub-schedules. In this manner, cyberphysical integration only needs to be applied between layers for termination control, and the synthesis problems for different layers can be solved separately.

Algorithm 1: Layering Algorithm

```
L1  L: a set of currently non-layered operations
L2  L_i: the set of operations allocated to layer L_i
L3  O: the set of indeterminate operations
L4  D_{o_j}: the set of descendant operations of o_j
L5  A_{o_j}: the set of ancestor operations of o_j
L6  R_{o_j} the set of to-be-removed operations resulted from the
    removal of o_j
L7  c_{o_j}: the cost resulted from the removal of o_j
L8  t: the constant representing the threshold number of
    indeterminate operations in each layer
L9  Initialize L as the set of all operations; i=0;
L10 while |L| ≠ 0 do
L11     i++;
L12     /* dependency-based allocation */
L13     for o_j ∈ O ⋂ L do
L14         if A_{o_j} ⋂ O ⋂ L = ∅ then
L15             L_i.insert(o_j);
L16             L.remove(o_j);
L17             for o ∈ D_{o_j} do
L18                 L.remove(o);
L19             end
L20         end
L21     end
L22     for o ∈ L do
L23         L_i.insert(o);
L24     end
L25     /* resource-based allocation */
L26     while |O ⋂ L_i| > t do
L27         for o ∈ O ⋂ L_i do
L28             (c_o, R_o) ← min_cut(o);
L29         end
L30         find R_o with minimum c_o;
L31         for o ∈ R_o do
L32             L_i.remove(o);
L33         end
L34     end
L35     Reset L;
L36 end
```

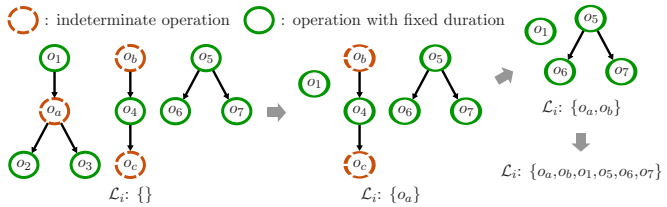

Figure 4: Maximum independent set algorithm.

Algorithm 1 shows the overall flow of our layering algorithm, which consists of two phases: dependency-based allocation and resource-based allocation. Some symbols defined in Algorithm 1 are also used in the following.

Since partitioning the problem may trap solutions into local optima, we reduce the number of layers by maximizing the number of operations in each layer. As indeterminate operations are allocated to the end of each sub-schedule, their descendant operations need to be allocated to subsequent layers. Therefore, we first perform a modified maximum independent set algorithm based on operation dependency (L12-L24).

As shown in Figure 4, we represent the non-layered operations as vertices and their dependency as edges. We first randomly choose an indeterminate operation, which has no indeterminate ancestor in $\mathcal{L}$. Suppose that we choose o_a. Since o_a cannot be reached from any other indeterminate operation, we add o_a to $\mathcal{L}_i$ and remove o_a and all its descendant operations from the graph ($\mathcal{L}$). Then we repeat the above steps until no indeterminate operation remains in the graph, and add all the remaining operations to $\mathcal{L}_i$.

As the last operations in each layer, indeterminate operations are mapped to different devices to allow parallel execution. Therefore, if the number of indeterminate operations

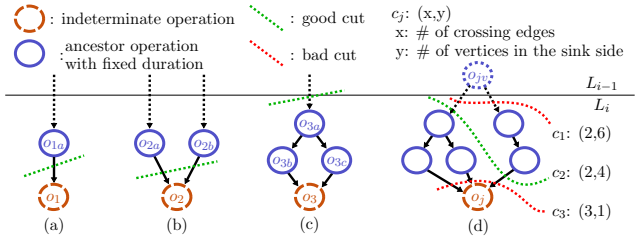

Figure 5: (a)(b)(c) Selection process. Storage usage: 1, 2, 1; to be removed ancestor operations: 0, 0, 3. (d) Different cut strategies.

in a layer surpasses a given threshold, we will perform a modified minimum-cut algorithm based on resource usage (L25-L34).

The idea of our resource-based allocation strategy can be illustrated by some examples shown in Figure 5(a)(b)(c). Suppose that o_1, o_2, and o_3 are indeterminate operations, which have ancestor operations allocated to both layer L_{i-1} and L_i. If we remove o_1, o_2, or o_3 from the current layer due to resource limitation, and leave some of their ancestor operations unmoved, the outputs of their unmoved ancestor operations need to be stored, which will occupy 1, 2, or 1 storage, respectively. In order to minimize the storage usage without removing too many operations, we tend to remove indeterminate operations that involve less reagent inheritance. Since o_1 involves less storage usage than o_2, and results in removing fewer ancestor operations than o_3, we choose o_1 to be removed from the current layer as our first priority. The cost for removing each indeterminate operation is formulated as a minimum-cut problem. As shown in Figure 5(d), suppose that o_j is an indeterminate operation in layer L_i. We set o_j as the *sink*, and create a virtual operation o_{jv} as the *source*, which is allocated to layer L_{i-1} and is the ancestor of all o_j's ancestors in L_i. c_1, c_2, and c_3 are three different cut strategies, among which c_1 and c_2 result in fewer crossing edges than c_3. Since c_2 puts fewer vertices to the sink side than c_1, we choose c_2 as the final cut solution for o_j. We implement our min-cut algorithm based on the Ford-Fulkerson algorithm [23].

3.2 Refinement by Progressive Re-synthesis

Applying our layering algorithm, we partition the high-level synthesis problem for a complex bioassay into sequential sub-problems, which are solved separately by our mathematical modeling method (details of this method will be discussed in Section 4). In order not to trap our solutions into local optima, we refine our solutions by a progressive re-synthesis process.

In the first phase of our synthesis process, device usage of operations in prior layers will be inherited by posterior layers for resource re-utilization. For example, suppose that there are two operations o_1, o_2, and the component-oriented requirements of o_1 and o_2 are formulated as: $C_{o_1}=\{\text{ring}\}, A_{o_1}=\{\text{sieve valve, pump}\}$, $C_{o_2}=\{\}, A_{o_2}=\{\text{sieve valve}\}$, where C_o represents the container specification of operation o, and A_o represents the accessory specifications. Since $C_{o_2}\subseteq C_{o_1}$, $A_{o_2}\subseteq A_{o_1}$, o_2 can be bound to a device that fulfills the requirements of o_1. If $o_1\in L_{i-1}$, $o_2\in L_i$, o_1 is bound to a device d, as shown in Figure 6(a), when performing synthesis for L_i, o_2 can be bound to d, too, so that no extra device is needed.

However, if $o_2\in L_{i-1}$, $o_1\in L_i$, as shown in Figure 6(b), without information of device usage in L_i, a device d' will be built for o_2, as a chamber involves less area cost than a ring. d' cannot be re-utilized for the execution of o_1, and thus extra device integration cost is required.

Since we cannot foresee the device usage of posterior layers, potential waste of device integration cannot be avoided. Therefore, we apply a progressive re-synthesis process to

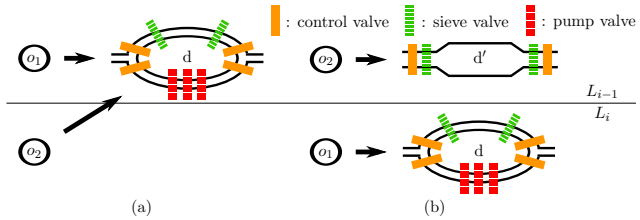

Figure 6: Potential risk of unnecessary device integration. (a) Device d is re-utilized in posterior layer. (b) Unnecessary device integration due to lack of information.

obtain more information about device usage. After synthesizing once for all layers, we edit the device inheritance rule and run the synthesis again.

In the first iteration, we let D_{i-1} be the set of all available devices in layer L_{i-1}. For operation $o\in L_i$, we first search in D_{i-1} for a suitable device to execute o, so that extra device integration can be avoided. Therefore, device usage in L_i can be formulated as $D_i=D_{i-1}\cup D'_i$, where D'_i is the set of devices that are newly integrated.

In the re-synthesis iterations, let D be the set of all devices instantiated in the previous iteration. When dealing with layer L_i, $D\backslash D'_i$ will be inherited for binding, which means that the device usage of operations in both previous and posterior layers are available, and thus a more comprehensive view can be achieved for better synthesis solutions.

This progressive re-synthesis process will be repeated until no further significant improvement is obtained anymore.

4 Mathematical Modeling Method

We propose an integer-linear-programming (ILP) model to synthesize scheduling and binding solutions for each layer. The inputs of our model include:

a. a set O, which is a collection of component-oriented definitions of biological operations in the current layer;

b. a set D indicating the usage of general devices, the cardinality of which represents the maximal number of devices allowed to be integrated on the chip, and is given by the user. D is shared among all models for different layers and is edited according to the inheritance rule as mentioned in Section 3.2.

The notations of variables in our model are listed in Table 1.

4.1 Preparation: Transportation Estimation

As mentioned in Section 3.1, sequential operations may require reagent inheritance. If the child operation is bound to a different device other than its parent operation, reagent transportation between devices must be considered. However, transportation time is closely related to the lengths of flow channels, which usually remain undetermined during the scheduling process.

Our progressive re-synthesis method enables us to estimate the transportation time of different operations according to potential chip layout. We assume that if sequential operations are bound to two devices d and d', a flow-channel path must be integrated to connect d with d'. Since devices may be re-utilized, paths between devices may also be used more than once. If a path p_a is used more often than another path p_b, to promote transportation efficiency, the channel length of p_a should be designed shorter than the channel length of p_b. As a result, the transportation time of operations utilizing p_a should also be shorter than the transportation time of operations utilizing p_b.

In this work, we first assign a constant t, which can be defined by the user, to all operations as their transportation time. Then we ask the user to define an arithmetic progression to represent different potential transportation times, and specify the minimum and maximum term, as well as the number of terms. After we run the synthesis process once for all layers, we refine the transportation time for each

operation as one of the terms in the pre-defined arithmetic progression based on the binding solutions. If sequential operations are bound to the same device, the transportation time will be set to 0. Each time we start a new iteration of our progressive re-synthesis process, the transportation time will be refined based on the latest results.

4.2 Model Construction

Device Configuration: according to our general device concept, every $d_j \in D$ has exactly one container, which is either a ring or a chamber. We formulate this constraint as:

$$\forall d_j \in D, \quad d_{j,r} + d_{j,ch} = 1. \tag{1}$$

To support operations with different reagent volumes, we define four different capacities for containers: *large*, *medium*, *small* and *tiny*. Then we introduce the following constraint to specify the volume for a certain container:

$$\forall d_j \in D, \quad d_{j,c_l} + d_{j,c_m} + d_{j,c_s} + d_{j,c_t} = 1. \tag{2}$$

Since the capacity of a ring is usually larger than the capacity of a chamber, we define that the capacity of a ring may vary among large, medium and small, and the capacity of a chamber may vary among medium, small and tiny, which is formulated as:

$$\forall d_j \in D, \quad d_{j,c_l} + d_{j,c_m} + d_{j,c_s} = d_{j,r}, \tag{3}$$

$$d_{j,c_m} + d_{j,c_s} + d_{j,c_t} = d_{j,ch}. \tag{4}$$

Component-oriented Consistence: in order to model the binding behavior among operations and devices, we introduce an operation-device-mapping variable $o\_d_{i,j}$ for each operation o_i and device d_j to represent whether o_i is bound to d_j. Then we introduce the following constraint to ensure that each operation is bound to exactly one device, the container type and accessories integration of which fulfill the corresponding requirements:

$$\forall o_i \in O, d_j \in D, x \in \{r, ch\}, y \in \{p, h, o, s, c\}, z \in \{c_l, c_m, c_s, c_t\}$$

$$\sum_{d_j \in D} o\_d_{i,j} = 1, \tag{5}$$

$$d_{j,x} - o\_d_{i,j} + 1 \geq o_{i,x}, \tag{6}$$

$$d_{j,y} - o\_d_{i,j} + 1 \geq o_{i,y}, \tag{7}$$

$$d_{j,z} - o\_d_{i,j} + 1 \geq o_{i,z}. \tag{8}$$

Operation Dependency: since a child operation can only start after collecting all its inputs from its parent operations, we introduce the following constraint:

$$\forall o_p \in O,\ o_c \in \{\text{child operations of } o_p\},$$

$$o_{c,st} \geq o_{p,st} + o_{p,dur} + t_p, \tag{9}$$

where t_p indicates the transportation time of o_p. Note that indeterminate operations have no child operation in the same layer, thus cannot be specified as o_p.

Device Conflicts Prevention: operations with overlapping execution time must be bound to different devices, since an operation needs to monopolize a certain device during its execution. We prevent device conflicts by applying the following constraints:

$$\forall o_a, o_b \in O, d_j \in D:$$

$$o_{a,st} + q_0 \cdot M \geq o_{b,st} + o_{b,dur} + t_b, \tag{10}$$

$$o_{a,st} + o_{a,dur} + t_a - q_1 \cdot M \leq o_{b,st}, \tag{11}$$

$$o\_d_{a,j} + o\_d_{b,j} - q_2 \leq 1, \tag{12}$$

$$q_0 + q_1 + q_2 \leq 2, \tag{13}$$

where M is an extremely large auxiliary constant, $\{q_0, q_1, q_2\}$ are auxiliary variables, at least one of which has to be set to 0 according to (13).

Table 1: Notation of Variables

Binary Variables			Operation o_i-related	Device d_j-related
Container Specification	ring (r)		$o_{i,r}$	$d_{j,r}$
	chamber (ch)		$o_{i,ch}$	$d_{j,ch}$
	Capacity	large (l)	o_{i,c_l}	d_{j,c_l}
		medium (m)	o_{i,c_m}	d_{j,c_m}
		small (s)	o_{i,c_s}	d_{j,c_s}
		tiny (t)	o_{i,c_t}	d_{j,c_t}
Accessory Specification	pump (p)		$o_{i,p}$	$d_{j,p}$
	heating pad (h)		$o_{i,h}$	$d_{j,h}$
	optical system (o)		$o_{i,o}$	$d_{j,o}$
	sieve valve (s)		$o_{i,s}$	$d_{j,s}$
	cell trap (c)		$o_{i,c}$	$d_{j,c}$
Transportation Path (between d and d')			$p_{d,d'}$	
Operation-Device-Mapping			$o\_d_{i,j}$	
Auxiliary Variable			q_0, q_1, q_2	
Integer Variables				
Operation Start Time (st)			$o_{i,st}$	
Operation Duration (dur)			$o_{i,dur}$	
Results Summation	toal execution duration		sum_t	
	area cost	ring (r)	$sum_{a,r}$	
		chamber (ch)	$sum_{a,ch}$	
	toal area cost		sum_a	
	processing cost	container (con)	$sum_{pr,con}$	
		accessory (acc)	$sum_{m,acc}$	
	total processing cost		sum_{pr}	
	total transportation paths		sum_p	

Indeterminate Execution: for each indeterminate operation o_i, we introduce the following constraint to ensure that it is allocated to the end of the corresponding sub-schedule:

$$\forall o_a \in O, \quad o_{a,st} \leq o_{i,st} + o_{i,dur}, \tag{14}$$

where $o_{i,dur}$ represents the minimum duration of o_i.

4.3 Objective Configuration

The objective of our modeling method is the minimization of total assay execution time, chip area cost, chip processing cost, and the number of transportation paths.

Total Assay Execution Time is decided by the last completed operation in this assay, which can be formulated as follows:

$$\forall o_i \in O, \quad sum_t \geq o_{i,st} + o_{i,dur}. \tag{15}$$

Chip Area Cost is decided by the integration of containers. We introduce the following constraints to model the area cost of different containers:

$$sum_{a,r} = \sum_{d_j \in D, d_{j,r}=1} \ \sum_{x \in \{c_l, c_m, c_s\}} A_x \cdot d_{j,x} \tag{16}$$

$$sum_{a,ch} = \sum_{d'_j \in D, d'_{j,ch}=1} \ \sum_{y \in \{c_m, c_s, c_t\}} A'_y \cdot d_{j',y}, \tag{17}$$

$$sum_a = sum_{a,r} + sum_{a,ch}, \tag{18}$$

where $A_{x \in \{c_l, c_m, c_s\}}$ and $A'_{y \in \{c_m, c_s, c_t\}}$ are constants indicating the area cost of a ring and a chamber with different capacity.

Chip Processing Cost is decided by the integration of containers and accessories. The processing cost of containers $sum_{pr,con}$ are decided by their types and capacities in a similar manner as above, and the processing cost of accessories can be formulated as:

$$sum_{pr,acc} = \sum_{d_j \in D} \ \sum_{z \in \{p,h,o,s,c\}} Pr_z \cdot d_{j,z}, \tag{19}$$

where $Pr_{z \in \{p,h,o,s,c\}}$ are constants indicating the processing cost of different accessories. The total processing cost can therefore be formulated as:

$$sum_{pr} = sum_{pr,con} + sum_{pr,acc}, \tag{20}$$

Table 2: Synthesis Results for Bioassays.

Testcase				Exe. Time	#D.	#P.	Runtime
1	#Op	16	Conv.	225m	3	3	5.531s
[10]	#Ind.Op	0	Our	220m	2	2	8.412s
2	#Op	70	Conv.	277m+I_1	24	82	5m12s
[7]	#Ind.Op	10	Our	244m+I_1	21	33	5m10s
3	#Op	120	Conv.	603m+I_1+I_2	24	95	10m1s
[17]	#Ind.Op	20	Our	492m+I_1+I_2	24	85	10m5s

Exe.Time column shows the execution duration of the assay. #D, #P columns show the number of applied devices and paths. Runtime column shows program runtime. Conv. and Our. rows show the results applying conventional synthesis method and our method, respectively. I_1, I_2 indicate the extra execution time of indeterminate operations in Layer 1 and Layer 2, the maximal number of devices ($|D|$) is set to 25, the maximal number of indeterminate operations in each layer is set to 10. m and s are time units representing minute (m) and second (s).

Transportation Paths can be counted by the following constraints:

$$\forall o_i \in O, o_j \in \{\text{child operations of } o_i\}, d, d' \in D,$$
$$o\_d_{i,d} + o\_d_{j,d'} - p_{d,d'} \leq 1, \quad sum_p = \sum_{d,d' \in D} p_{d,d'}. \quad (21)$$

Objective Configuration: with the above configurations, we can formulate the objective of our model as the following:

$$\text{Minimize:} \quad C_t \cdot sum_t + C_a \cdot sum_a + C_{pr} \cdot sum_{pr} + C_p \cdot sum_p,$$

where C_t, C_a, C_{pr}, and C_p are adjustable weight coefficients that can be defined by users.

5 Experimental Results

We use C++ to implement our synthesis method and solve our ILP model with the ILP solver Gurobi [24] on a computer with a 2.67GHz CPU.

We generate scheduling and binding solutions for three assays from [7, 10, 17]. To demonstrate the ability of our method to handle large cases, we introduce replicated operations with the same protocol of the original assay, so that the number of operations in each assay are 16, 70, and 120, respectively. Our solutions are compared with the solutions generated by a modified conventional synthesis method.

Since the conventional synthesis method cannot support many up-to-date applications, we modify it by classifying operations and devices according to their component requirements instead of functionality. To enable the conventional scheduling method to support indeterminate operations, we integrate our layering algorithm and progressive re-synthesis process to the conventional method, too.

As shown in Table 2, our solutions show advantages in execution time reduction, device reduction and transportation path reduction. Our method avoids unnecessary device integration and balances the usage of chip resources, so that more operations can be executed in parallel. In case 1 and case 2, we achieve the time reduction even with fewer devices, since the reduction of transportation paths also results in the reduction of transportation time. In case 3, compared with the solutions generated by the modified conventional method, we reduce the assay execution time to 81.7% without increasing the number of devices.

Case 2 and case 3 include operations with indeterminate execution duration, we thus apply our layering algorithm to synthesize hybrid-scheduling solutions for them. The initial results are further refined by our progressive re-synthesis process. If the improvement compared with former results is larger than 10%, we will run another iteration. Table 3 shows the effectiveness of our progressive re-synthesis process: without increasing the usage of devices, we achieve circa 20% improvement in assay execution time.

Table 3: Improvement from Progressive Re-Synthesis.

Testcase		Initial	1_{st} Ite.	Improve	2_{nd} Ite.	Improve
2	Exe.Time	295m	247m	16.27%	244m	1.21%
[7]	#D.	21	21	0%	21	0%
3	Exe.Time	641m	530m	17.32%	492m	7.17%
[17]	#D.	24	24	0%	24	0%

6 Conclusion

Continuous-flow microfluidics evolves rapidly. The innovation of microfluidic components and bioassay protocols challenges conventional design automation approaches. In this work, we take a closer look at the characteristics and requirements of devices and operations, based on which we propose a component-oriented high-level synthesis method, which improves the utilization of chip resources and better adapts to technological developments. For assays involving indeterminate operations, we propose a layering algorithm and a mathematical modeling method to generate binding and hybrid-scheduling solutions, and thus turn a new page on high-level synthesis of continuous-flow microfluidics.

7 References

[1] D. Mark, S. Haeberle, G. Roth, F. v. Stetten, and R. Zengerle, "Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications," *Chem. Soc. Rev.*, vol. 39, pp. 1153–1182, 2010.

[2] A. M. Amin, M. Thottethodi, T. Vijaykumar, S. Werely, and S. C. Jacobson, "Aquacore: a programmable architecture for microfluidics," in *Proc. Int. Symp. on Comput. Archi.*, pp. 254–265, 2007.

[3] W. H. Minhass, P. Pop, and J. Madsen, "System-level modeling and synthesis of flow-based microfluidic biochips," in *Proc. Int. Conf. Compil., Arch. and Syn. Embed. Sys.*, pp. 225–234, 2011.

[4] K.-H. Tseng, S.-C. You, J.-Y. Liou, and T.-Y. Ho, "A top-down synthesis methodology for flow-based microfluidic biochips considering valve-switching minimization," in *Proc. Int. Symp. Phy. Des.*, pp. 123–129, 2013.

[5] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga, "Architectural synthesis of flow-based microfluidic large-scale integration biochips," in *Proc. Int. Conf. Compil., Arch. and Syn. Embed. Sys.*, pp. 181–190, 2012.

[6] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proc. Design Autom. Conf.*, pp. 141:1–141:6, 2015.

[7] J. F. Zhong, Y. Chen, J. S. Marcus, A. Scherer, S. R. Quake, C. R. Taylor, and L. P. Weiner, "A microfluidic processor for gene expression profiling of single human embryonic stem cells," *Lab on a Chip*, vol. 8, no. 1, pp. 68–74, 2008.

[8] J. Liu, M. Enzelberger, and S. R. Quake, "A nanoliter rotary device for polymerase chain reaction," *Electrophoresis*, vol. 23, pp. 1531–1536, 2002.

[9] M. Li, T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations," in *Proc. Design, Automation, and Test Europe Conf.*, 2016.

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

[11] D. D. Carlo, N. Aghdam, and L. P. Lee, "Single-cell enzyme concentrations, kinetics, and inhibition analysis using high-density hydrodynamic cell isolation arrays," *Anal. Chem.*, vol. 78, pp. 4925–4930, 2006.

[12] Y. Marcy, T. Ishoey, R. S. Lasken, T. B. Stockwell, B. P. Walenz, A. L. Halpern, K. Y. Beeson, S. M. D. Goldberg, and S. R. Quake, "Nanoliter reactors improve multiple displacement amplification of genomes from single cells," *PLoS Genet*, vol. 9, no. 3, p. e155, 2007.

[13] M. Ibrahim, K. Chakrabarty, and K. Scott, "Integrated and real-time quantitative analysis using cyberphysical digital-microfluidic biochips," in *Proc. Design, Automation, and Test Europe Conf.*, pp. 630–635, 2016.

[14] A. R. Wu, J. B. Hiatt, R. Lu, J. L. Attema, N. A. Lobo, I. L. Weissman, M. F. Clarke, and S. R. Quake, "Automated microfluidic chromatin immunoprecipitation from 2,000 cells," *Lab on a Chip*, vol. 9, pp. 1365–1370, 2009.

[15] T.-M. Tseng, M. Li, B. Li, T.-Y. Ho, and U. Schlichtmann, "Columba: Co-layout synthesis for continuous-flow microfluidic biochips," in *Proc. Design Autom. Conf.*, 2016.

[16] H. Yao, Q. Wang, Y. Ru, T.-Y. Ho, and Y. Cai, "Integrated flow-control co-design methodology for flow-based microfluidic biochips," *IEEE Des. Test. Comput.*, 2015.

[17] A. K. White, M. VanInsberghe, O. I. Petriv, M. Hamidi, D. Sikorski, M. A. Marra, J. Piret, S. Aparicio, and C. L. Hansen, "High-throughput microfluidic single-cell RT-qPCR," *Proc. Natl. Acad. Sci.*, vol. 108, no. 34, pp. 13 999–14 004, 2011.

[18] J. Wang, H. C. Fan, B. Behr, and S. R. Quake, "Genome-wide single-cell analysis of recombination activity and *de novo* mutation rates in human sperm," *Cell*, vol. 150, no. 2, pp. 402–412, 2012.

[19] R. Gomez-Sjoeberg, A. A. Leyrat, D. M. Pirone, C. S. Chen, and S. R. Quake, "Versatile, fully automated, microfluidic cell culture system," *Anal. Chem*, vol. 79, pp. 8557–8563, 2007.

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

[21] A. Gross, J. Schoendube, S. Zimmermann, M. Steeb, R. Zengerle, and P. Koltay, "Technologies for single-cell isolation," *Int. J. Mol. Sci.*, vol. 16, no. 8, pp. 16 897–16 919, 2015.

[22] K. Gupta, D.-H. Kim, D. Ellison, C. Smith, A. Kundu, J. Tuan, K.-Y. Suh, and A. Levchenko, "Lab-on-a-chip devices as an emerging platform for stem cell biology," *Lab on a Chip*, vol. 10, pp. 2019–2031, 2010.

[23] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Section 26.2: The Ford-Fulkerson method, Introduction to Algorithms (Second ed.).* MIT Press and McGraw-Hill.

[24] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual.* http://www.gurobi.com.