Relative-Scheduling-Based High-Level Synthesis for Flow-Based Microfluidic Biochips

Fangda Zuo[‡], Mengchu Li[‡], Tsun-Ming Tseng[‡], Tsung-Yi Ho[⋆], and Ulf Schlichtmann[‡]
[‡]Institute for Electronic Design Automation, Technische Universität München, Arcisstrasse 21, D-80333 Munich, Germany
[⋆]Department of Computer Science, National Tsing Hua University, No. 101, Section 2, Kuang-Fu Road, 30013 Hsinchu, Taiwan
mylxzzfd@gmail.com, {mengchu.li,tsun-ming.tseng, ulf.schlichtmann}@tum.de, tyho@cs.nthu.edu.tw

***Abstract*—The rapid development of microfluidic biochips requires matching automated synthesis methods. In particular, high-level synthesis methods for microfluidic chips need to consider various bio-constraints regarding time and device occupancy. Recent biochemical applications show that the duration of some bio-operations cannot be predicted in advance, which increases the likelihood that current synthesis methods would waste on-chip resources or even violate given bio-constraints. In this work, we present a relative-scheduling-based high-level synthesis method to optimize the bio-assay schedules and the usage of on-chip devices considering bio-operations with indeterminate durations. Experimental results show that our method significantly reduces the total execution time of bioassays without violating bio-constraints when using the same on-chip resources.**

I. Introduction

A flow-based microfluidic biochip is a lab-on-a-chip (LoC) that integrates multiple miniaturized devices into a coin-sized fluidic circuit. It enables different fluidic operations to be performed in parallel with different reagents at submillimeter scale [1], and is used for numerous high-throughput applications covering in situ click chemistry [2], synthetic biology [3], cell culturing [4] and protein analysis [5].

High-level synthesis (HLS) of flow-based microfluidic biochips from bio-assay protocols has been known as an efficient design methodology [6]. Similar to HLS of electronic circuits, *scheduling* and *binding* are the two primary tasks in HLS of flow-based biochips. The goal of scheduling and binding in flow-based biochip design is to improve the efficiency of the target assay and the biochip in terms of assay duration and the number of on-chip devices. It is worth noting that the scheduling and binding results need to strictly satisfy potential bio-constraints.

Traditional HLS methods mostly assume that for a given bio-assay, the duration of each fluidic operation is fixed and can be known prior to the execution of the assay. This assumption, however, does not always hold. For example, some fluidic operations such as single-cell isolation have a certain chance to fail. In order to achieve the target products, the operations may need to be repeated multiple times and their exact duration remains unknown at design time [7]. The unknown duration does not only bring indeterminacy to scheduling, but also increases the difficulty in binding, as it is not clear to the designer when the devices occupied by these operations can be available for binding again.

What makes this indeterminacy more crucial is that there are sometimes strict time-constraints on the delay between two dependent operations. For example, some operations involve reagents that are prone to over-reaction, and therefore require their succeeding operations to start within short time after the reagents have been treated [8]. Without modelling the indeterminacy in the scheduling methods, the synthesized results are likely to violate the time-constraints.

We present in this paper a high-level synthesis method based on relative scheduling to prevent the potential conflict caused by indeterminate duration and time constraints. Given the sequencing graph of a bio-assay, our method automatically synthesizes a high-level description of the required on-chip devices, schedules the fluidic operations, and binds the operations to the synthesized devices. We formally model the time dependency between operations with indeterminate duration and their succeeding operations, so that we can integrate the indeterminacy into a comprehensive mixed-integer linear programming (MILP) model to optimize the time- and resource efficiency of the target application. We investigate the performance of our HLS method with real-world test cases and show that our method significantly reduces the total duration of bio-assays without violating time- and device-constraints.

II. Related work

Minhass et al. proposed a series of early architectural synthesis works [9], [10], [11] for flow-based microfluidic biochips that set many standards for later HLS research. These works assume that a bioassay and a library of on-chip devices are given and perform resource constrained list-scheduling and binding. To this end, each operation was marked with an explicit duration and assigned to a fixed slot in the schedule. Later HLS research mostly adopted this problem formulation and added several practical constraints to the synthesis process. Tseng et al. [12] and Liu et al. [13] investigated the storage of intermediate assay products. Li et al. [14] modelled *immediate execution*, which requires two dependent operations to be scheduled one-after-the-other without delay. Our method considers these constraints and in particular, we generalize the immediate execution constraint to a time constraint that specifies a given upper bound on the delay between two dependent operations.

Li et al. [15] proposed to model on-chip devices with a combinable device library to enable flexible binding options. Besides, [15] considered operations with *indeterminate duration* for the first time. Two heuristic algorithms were applied to divide bio-assays that involve such operations into multiple sub-assays, and thereby operations with indeterminate duration can be scheduled to the end of each sub-assay without competing for resources with other operations. However, this method cannot correctly model the storage- and time-constraints, and the division of the assay leads to suboptimal synthesis results. In this work, we adopt the combinable device

ICCAD 2021, DOI: 10.1109/ICCAD51958.2021.9643477
https://ieeexplore.ieee.org/document/9643477
https://dl.acm.org/doi/10.1109/ICCAD51958.2021.9643477

library from [15]. Instead of dividing the synthesis problem, we introduced a time property called *nominal schedule* to model the indeterminate delay between dependent operations and solve the HLS problem as a whole.

Ku et al. [16] proposed a *relative scheduling* method to schedule operations with unbounded execution delays for digital application-specific integrated circuits (ASICs). Given a sequencing graph of dependent operations and time constraints, [16] introduced a property called *well-posedness* to detect and resolve potential conflicts caused by the unbounded execution delay. In this work, we adapt the *well-posedness* check proposed in [16] to microfluidic scenario as a preprocessing step before HLS.

Taking the preprocessed sequencing graph as an input, we propose to perform scheduling and binding simultaneously to optimize the time and resource efficiency of a flow-based biochip application. In particular, we formally model the time and device dependency between sequential operations with an integer linear programming method to synthesize conflict-free scheduling and binding results. To improve the efficiency of the synthesis, we propose two heuristic methods to reduce the search space of our model.

III. Problem Statement

We model a bio-assay as a sequencing graph, in which vertices represent fluidic operations and edges represent the dependency between operations. Specifically, there are two types of operations: *operations with fixed duration*, which are represented as single-line vertices, and *operations with indeterminate duration*, which are represented as double-line vertices. Besides, if there are time constraints on the delay between two dependent operations, we will integrate the constraints into the sequencing graph by adding a dotted edge from the child vertex to the parent vertex.

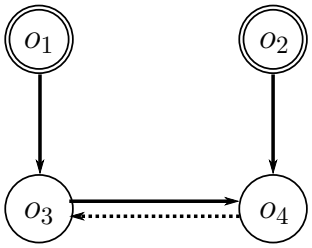

Figure 1: An example of the input sequencing graph

Figure 1(a) shows an example of the input sequencing graph. The target bio-assay consists of four operations, among which o_1 and o_2 are operations with indeterminate duration, and o_3 and o_4 are operations with fixed duration. There is a time constraint on the delay between o_3 and o_4, which requires that the start time of o_4 must be no later than the end time of o_3 plus a given delay upper bound. Note that the duration of each operation and the delay upper bound are also given as input of the HLS problem. For operations with indeterminate duration, their duration is modelled as the sum of their minimum duration and an indeterminate delay.

Besides time constraints, we assume that the input also specifies for each operation its device requirement, including the size, shape, and integrated components of the required on-chip device. This requirement can be modelled with the component-oriented device library proposed in [15].

The goal of our HLS approach is to synthesize:

- a logic description of the required flow-based microfluidic biochip, including the number and the specifications of the on-chip devices and their logic connections;
- a binding solution that specifies for each operation an on-chip device to execute the operation;
- a schedule of the bio-assay that specifies the start time of each operation. If the target assay involves operations with indeterminate duration, the start time should be specified relative to the indeterminate duration.

The synthesized scheduling and binding results must satisfy all device- and time-constraints. Besides, the number of on-chip devices and the total duration of the assay should be minimized.

IV. Preprocessing

A time constraint on dependent operations may not be satisfiable if the start time of the child operation depends on the completion of an operation with indeterminate duration.

For example, in the sequencing graph shown in Figure 1, if we denote the start time of o_3 and o_4 as t_3 and t_4, the duration of o_3 as τ_3, and the upper bound of the delay between the end time of o_3 and the start time of o_4 as c, the time constraint represented by the dotted line can be formulated as:

$$t_4 \leq t_3 + \tau_3 + c. \tag{1}$$

However, as o_4 is also the child operation of o_2, it can only start after o_2 ends. If we denote the start time of o_2 as t_2 and the duration of o_2 as $\tau_2 + \delta_2$, in which τ_2 is the minimum duration of o_2 and δ_2 is the indeterminate delay of o_2, the dependency between o_2 and o_4 can be formulated as:

$$t_2 + \tau_2 + \delta_2 \leq t_4. \tag{2}$$

Thus, whether the time constraint is satisfiable depends on whether the following inequation can be satisfied:

$$t_2 + \tau_2 + \delta_2 \leq t_3 + \tau_3 + c. \tag{3}$$

As the left-hand side of the inequation contains an indeterminate delay δ_2, and a scheduling method can only manage t_2 and t_3, no scheduling method can guarantee the satisfiability of this constraint.

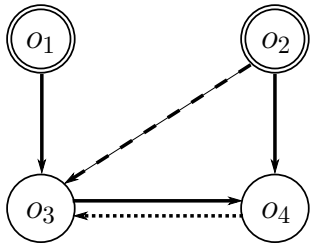

Figure 2: A revised sequencing graph.

In order to synthesize feasible scheduling results despite the indeterminacy, we revise the sequencing graph by adding an auxiliary dependency between o_2 and o_3, as shown in Figure 2, in which the auxiliary dependency is represented by a dashed

directed edge. By adding this dependency, we require that o_3 can only start after the completion of o_2, i.e.

$$t_2+\tau_2+\delta_2 \leq t_3, \tag{4}$$

and thus make sure that constraint (3) is always satisfied, which further ensures that the indeterminate duration will not harm the satisfiability of the time constraint.

The above described approach was adopted from the approach proposed in [16] for resolving similar conflicts in electronic circuit design. Specifically, given an acyclic graph $G=(V,E)$ and a set $A \subseteq V$ of vertices representing operations with indeterminate duration[1], [16] introduced a property called *well-posedness*. A directed edge (i,j) is considered *well-posed* if there is no vertex $v \in A$ that is a predecessor of i but not a predecessor of j. If all edges in the graph are well-posed, the graph is free from satisfiability concerns caused by the indeterminacy.

For example, in Figure 1, edge (o_4,o_3) is not well-posed, because o_2, as an operation with indeterminate duration, is a predecessor of o_4 but not a predecessor of o_3. This indicates that the satisfiability of the time constraint modelled by (o_4,o_3) is dependent on the indeterminate duration of o_2. By adding an extra dependency from o_2 to o_3, as shown in Figure 2, (o_4,o_3) becomes well-posed and the concern is thus resolved.

We implement this *well-posedness* check to revise the input sequencing graph before starting the HLS process. With the revised graph as the new input, we propose a mathematical modelling method to synthesize the scheduling and binding solutions.

V. Method

We model the HLS problem for flow-based microfluidic biochips with mixed-integer linear programming. For the sake of simplicity, we refer to *operations with indeterminate duration* as *indeterminate operations*.

A. Nominal Schedule

For bio-assays that involve indeterminate operations, we introduce a time property called *nominal schedule* to exclude the indeterminacy in the model. Specifically, we consider a bio-assay as a combination of sequential determinate execution stages. Between two neighboring determinate execution stages, there is an indeterminate gap which refers to the indeterminate delay of the operations that start in the earlier determinate execution stage.

For example, Figure 3 illustrates a possible schedule for the input sequencing graph shown in Figure 2. The assay consists of two determinate execution stages: operations o_1 and o_2 start in stage 1; operations o_3 and o_4 start in stage 2. Between stage 1 and stage 2, there is an indeterminate gap Δ_1, which represents the potential delay of o_1 and o_2. In other words, we model the actual completion time of o_1 and o_2 as the duration of stage 1 added with Δ_1.

By introducing the indeterminate gap, we are able to model the start time of operations that are the successors of indeterminate operations. For example, the start time of o_3 can be modelled as $\tau_2+\Delta_1$, in which τ_2 is the duration as well as the completion time of stage 1, and Δ_1 is the indeterminate gap

[1]originally formulated as "unbounded execution delay" in [16].

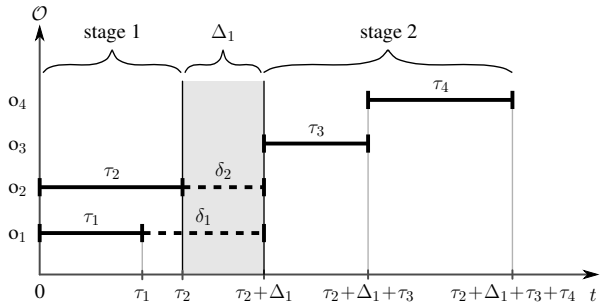

Figure 3: An example of the input sequencing graph

between stage 1 and stage 2. Similarly, the start time of o_4 can be modelled as $\tau_2+\Delta_1+\tau_3$, in which τ_3 is the duration of o_3. The completion time of both stages τ_2 and $\tau_2+\Delta_1+\tau_3+\tau_4$ are determined by the end time of the latest operation in each determinate execution stage.

In this manner, we can model the actual duration of an assay as the sum of the duration of all determinate execution stages and all the indeterminate gaps between stages. For example, the total duration of the assay shown in Figure 3 is modelled as $\tau_2+\Delta_1+\tau_3+\tau_4$, in which τ_2 is the duration of stage 1; Δ_1 is the indeterminate gap, and $\tau_3+\tau_4$ is the duration of stage 2. This is also the completion time of stage 2 in this example. We notice that in this model, if we put aside the indeterminate part that we cannot control, the duration of the assay depends only on the duration of the determinate execution stages. Thus, the optimization of the actual schedule can be simplified as optimizing the total duration of the determinate execution stages.

In contrast to the actual schedule of an assay, we introduce a *nominal schedule* which puts aside the indeterminate gaps and only models the determinate execution stages. For example, in contrast to the actual completion time of o_1, which is $\tau_1+\delta_1$, the nominal completion time of o_1 is τ_1. Similarly, the nominal completion time of o_2, o_3 and o_4 is τ_2, $\tau_2+\tau_3$, and $\tau_2+\tau_3+\tau_4$, respectively. Thus, the nominal duration of the whole assay is $\tau_2+\tau_3+\tau_4$, which is the completion time of the latest operation. In order to get the actual schedule of an operation, we just need to add the indeterminate gaps back to the nominal schedule. For example, in Figure 3, since there is an indeterminate gap Δ_1 before stage 2, the actual completion time of o_3 in stage 2 is $\tau_2+\tau_3+\Delta_1$.

As a conclusion, when we add the indeterminate gaps back to the nominal schedule, we get exactly the actual schedule of the assay. Thus, we can focus on optimizing the nominal schedule despite the indeterminacy of the assay.

B. Mathematical Model

We introduce a set $\mathcal{O}$ for all operations specified in the given assay, and a subset $\mathbb{O} \subseteq \mathcal{O}$ for all indeterminate operations specified in the assay. The notations of variables and constants in our model are listed in Table I. In particular, we represent binary variables with b and integer variables with v.

1) Assignment of the determinate execution stages

Since determinate execution stages are separated by indeterminate gaps, which is shaped by at least one indeterminate

Table I: Notation of Variables and Constants

Binary Variables	
operation-stage-mapping	$b_{o,k}^{stage}$
operation-device-mapping	$b_{o,j}^{device}$
same-device-indicator	$b_{o_1,o_2}^{same\_device}$
auxiliary variables	$q_1, q_2, ...$
Integer Variables	
operation-stage-index	$v_o^{s\_id}$
stage completion time	v_k^T
operation start time	v_o^t
occupation start time	$v_o^{d.start}$
occupation end time	$v_o^{d.end}$
product transportation time	$v_{p,c}^{trans}$
Constants	
operation duration	τ_o
product exportation time	θ
very large integer	M

operation, an assay can have at most $|\mathbb{O}|+1$ determinate execution stages, where $|\mathbb{O}|$ is the number of indeterminate operations in the assay.

To model the stage assignment, we introduce $|\mathbb{O}|+1$ binary variables $b_{o,k}^{\text{stage}}$ for each operation $o \in \mathcal{O}$ with $1 \le k \le |\mathbb{O}|+1$. $b_{o,k}^{\text{stage}} = 1$ indicates that operation o is assigned to stage k.

To model that each operation must be assigned to exactly one determinate execution stage, we introduce the following constraint:

$$\forall o \in \mathcal{O}: \sum_{1 \le k \le |\mathbb{O}|+1} b_{o,k}^{\text{stage}} = 1. \tag{5}$$

Besides, we introduce an integer variable $v_o^{\text{s\_id}}$ to model the exact stage index of operation $o \in \mathcal{O}$ with the following constraint:

$$v_o^{\text{s\_id}} = \sum_{1 \le k \le |\mathbb{O}|+1} k \cdot b_{o,k}^{\text{stage}}. \tag{6}$$

For example, if operation o_3 is assigned to stage 2, $b_{o,2}^{\text{stage}}$ will be set to 1, while $b_{o,1}^{\text{stage}}$, $b_{o,3}^{\text{stage}}$, $\cdots$, $b_{o,|\mathbb{O}|+1}^{\text{stage}}$ will be set to 0. Thus, $v_o^{\text{s\_id}}$ will be calculated as $0 \cdot 1 + 1 \cdot 2 + 0 \cdot 3 + \cdots + 0 \cdot (|\mathbb{O}|+1) = 2$.

To model the completion time of each determinate execution stage, we introduce $|\mathbb{O}|+2$ integer variables v_k^T for $0 \le k \le |\mathbb{O}|+1$. In particular, v_0^T is introduced as an auxiliary variable and its value is constantly set to 0.

We introduce the following constraint to model that the determinate exectuion stages are sequentially ordered:

$$\forall 1 \le k \le |\mathbb{O}|+1: \quad v_{k-1}^T \le v_k^T. \tag{7}$$

To model the start time of each operation $o \in \mathcal{O}$, we introduce an integer variable v_o^t. We introduce the following constraints to model that if an operation is assigned to stage k, its nominal start time and completion time must be within the range of $[v_{k-1}^T, v_k^T]$:

$$\forall 1 \le k \le |\mathbb{O}|+1: \quad v_{k-1}^T \le v_o^t + (1 - b_{o,k}^{\text{stage}}) \cdot M, \tag{8}$$

$$v_k^T \ge v_o^t + \tau_o - (1 - b_{o,k}^{\text{stage}}) \cdot M, \tag{9}$$

in which τ_o is a constant representing the given duration operation o and M is an extremely large auxiliary constant. If an operation o is assigned to stage k, $b_{o,k}^{\text{stage}}$ will be set to 1, thus, constraints (8) and (9) become:

$$v_{k-1}^T \le v_o^t, \qquad v_k^T \ge v_o^t + \tau_o, \tag{10}$$

which confine the range of the start and the completion time of o. On the other hand, if operation o is not assigned to stage k, i.e. $b_{o,k}^{\text{stage}} = 0$, constraints (8) and (9) will become:

$$v_{k-1}^T \le v_o^t + M, \qquad v_k^T \ge v_o^t + \tau_o - M. \tag{11}$$

Since M is an extremely large constant, these constraints are always satisfied and thus do not confine the nominal schedule of o.

Since the duration of indeterminate operations cannot be predicted, we introduce the following constraint to assume that an indeterminate operation will be in progress till the end of the corresponding execution stage:

$$\forall o \in \mathbb{O} \ \forall 1 \le k \le |\mathbb{O}|+1: v_o^t + \tau_o + (1 - b_{o,k}^{\text{stage}}) \cdot M \ge v_k^T. \tag{12}$$

If indeterminate operation o is assigned to stage k, i.e. $b_{o,k}^{\text{stage}} = 1$, constraint (12) is equivalent to:

$$v_o^t + \tau_o \ge v_k^T. \tag{13}$$

As the completion time of o cannot be larger than the completion time of stage k owing to constraint (9), this constraint means that the completion time of o is equal to the completion time of stage k. On the other hand, if operation o is not assigned to stage k, i.e. $b_{o,k}^{\text{stage}} = 0$, constraint (12) is equivalent to:

$$v_o^t + \tau_o + M \ge v_k^T,$$

which is a tautology since M is an extremely large constant.

Additionally, since the potential delay of indeterminate operations is unpredictable but non-negligible, if an indeterminate operation is assigned to a device, the device must be regarded as occupied till the end of the stage. Take Figure 1 again as an example. o_1 and o_2 are both indeterminate operations and both assigned to stage 1 with the same scheduled start time at 0. The completion time of stage 1 is τ_2, dominated by the nominal completion time of o_2. Though the nominal completion time of o_1 is τ_1 and $\tau_1 < \tau_2$, the device bound by o_1 cannot be regarded as free from the time τ_1 to τ_2 due to the indeterminate delay. Consequently, aligning the completion time of indeterminate operations to the end of their stage using constraint (12) will not harm the optimality of the stage completion time in our model. Furthermore, the device bound to o_1 can be released before the start of o_1 for other usage.

Moreover, since the end time of an indeterminate operation is considered identical to the completion time of its execution stage, if an indeterminate operation p has a child operation c, c must be executed in a subsequent stage. We formulate the constraint as:

$$v_p^{s\_id} < v_c^{s\_id}. \tag{14}$$

2) Operation dependency and storage of intermediate products

For sequential operations, the products of an earlier operation may need to be temporarily stored in its current device if the device of the later operation is not ready for use. In other words, an operation may occupy its device even after its completion. For example, suppose that there are three operations o_1, o_2 and o_3 and two devices A and B: o_1 is bound to device A; o_2 and o_3 are bound to device B; o_1 is the parent operation of o_2; and o_3 starts before o_2. Suppose that when o_1 is completed, o_3 is still in progress and occupies device B. In this case, the product of o_1 cannot be delivered to device B for o_2 to start. Instead, it needs to be temporarily stored in device A to wait for the completion of o_3 despite the completion of o_1.

To accurately model the dynamic device usage during the assay execution process, we introduce two integer variables $v_o^{d.\text{start}}$ and $v_o^{d.\text{end}}$ for each operation $o \in \mathcal{O}$ to represent the time that o starts and ends to occupy a device, respectively. In particular, we introduce the following constraints to make sure that the nominal schedule of o is within the range $[v_o^{d.\text{start}}, v_o^{d.\text{end}}]$:

$$v_o^{d.\text{start}} \le v_o^t, \tag{15}$$

$$v_o^{d.\text{end}} \ge v_o^t + \tau_o. \tag{16}$$

Besides, if an operation has multiple child operations, different child operations may require the products of the parent operation at different times. The parent operation will occupy the device that it is bound to until its latest child operation is ready to receive the product. For example, suppose that there is an operation o_1 that has two child operations o_2 and o_3; o_1, o_2 and o_3 are bound to devices A, B and C, respectively. At the time when o_1 completes, device B is available for carrying out o_2 but device C is occupied by other operations and cannot carry out o_3. In this case, some product of o_1 will keep occupying device A until device C is ready.

Furthermore, we introduce an integer variable $v_{p,c}^{\text{trans}}$ for each pair of parent-child operations to model the time that a parent operation o_p starts to transport its product to a child operation o_c. Suppose that θ denotes a constant duration to transport a fluid from one device to another. $v_{p,c}^{\text{trans}} + \theta$ is the latest time that operation o_c starts to occupy its device. We therefore construct the following constraints:

$$v_{p,c}^{\text{trans}} \ge v_p^t + \tau_p, \tag{17}$$

$$v_{p,c}^{\text{trans}} \le v_p^{d.\text{end}}, \tag{18}$$

$$v_{p,c}^{\text{trans}} + \theta \cdot (1 - b_{p,c}^{same\_device}) \ge v_c^{d.\text{start}}, \tag{19}$$

$$v_{p,c}^{\text{trans}} + \theta \cdot (1 - b_{p,c}^{same\_device}) \le v_c^t. \tag{20}$$

Constraint (17) describes that the parent operation o_p can only send product to its child operation o_c when o_p itself is done, and constraint (18) describes that o_p can only send products to o_c when o_p still occupies the device. The binary variable $b_{p,c}^{same\_device}$ introduced in constraint (19) indicates whether o_p and o_c are bound to the same device. With $b_{p,c}^{same\_device}$, constraint (19) describes that if o_p and o_c are bound to different devices, o_c must start to occupy its device before the product of o_p arrive at $v_{p,c}^{\text{trans}} + \theta$. Constraint (20) describes that o_c can only start after it receives all products from its parent operations. If both o_p and o_c are bound to the same device, $b_{p,c}^{same\_device}$ will be set to 1, and therefore, constraint (19) and constraint (20) become:

$$v_{p,c}^{\text{trans}} \ge v_c^{d.\text{start}}, \tag{21}$$

$$v_{p,c}^{\text{trans}} \le v_c^t, \tag{22}$$

so that the fluid transportation time θ can be saved.

We introduce a set D for all initiated devices. The constraints used to confirm whether two operations are bound to the same device are as follows:

$$\forall d_j \in D: \quad q_j \le b_{p,j}^{device}, \tag{23}$$

$$q_j \le b_{c,j}^{device}, \tag{24}$$

$$b_{p,c}^{same\_device} = \sum_{\forall d_j \in D} q_j, \tag{25}$$

$$b_{p,j}^{device} + b_{c,j}^{device} - b_{p,c}^{same\_device} \le 1. \tag{26}$$

where the binary variable $b_{o,j}^{device} = 1$ indicates that operation o is bound to device j and q_j is an auxiliary variable. Constraints (23), (24) and (25) describes that if parent operation o_p and child operation o_c are bound to different devices, at least one of $b_{p,j}^{device}$ and $b_{c,j}^{device}$ must be 0. Consequently, all q_j and then $b_{p,c}^{same\_device}$ must be set to 0.

Nevertheless, when both operations o_p and o_c are bound to the same device d_j, the auxiliary variable q_j may still be set to 0 according constraints (23) and (24). To prevent this situation, constraint (26) is introduced. If both operations o_p and o_c are bound to the same device d_j, both variables $b_{p,j}^{device}$ and $b_{c,j}^{device}$, and consequently the variable $b_{p,c}^{same\_device}$, must be set to 1.

3) Upper bound of the delay between dependent operations

If an upper bound is specified for the delay between two dependent operations o_p and o_c, i.e. o_c must start within a given time π after the completion of o_p, we need to ensure that o_p and o_c are assigned to the same determinate execution stage to prevent that the upper bound is violated by the indeterminate gap. To note is that the duration of the parent operation o_p should be determinate. To this end, we introduce the following constraints:

$$v_p^{\text{s\_id}} = v_c^{\text{s\_id}}, \tag{27}$$

$$v_c^t \ge v_p^t + \tau_p + \pi. \tag{28}$$

4) Device conflicts and device consistency

Besides the above introduced constraints, we also need to introduce some device-related constraints to prevent operations that have overlapping schedules from being bound to the same device.

We prevent overlapping schedules by applying the following constraints:

$$\forall o_a, o_b \in \mathcal{O},$$

$$v_a^{d.start} + (1 - q_1) \cdot M \ge v_b^{d.end}, \tag{29}$$

$$v_b^{d.start} + (1 - q_2) \cdot M \ge v_a^{d.end}, \tag{30}$$

$$q_1 + q_2 = b_{a,b}^{same\_device}. \tag{31}$$

Algorithm 1: Device specification

$settledOperations$: set of settled operations;
$specDeviceOperation$: map of specified device-operation pairs;
O: set of all operations;
D: set of all initiated devices;
$isDeviceConflict(o_1,o_2)$: function returns binary value, TRUE if both operations cannot be executed in the same device due to their requirements;

```
init undirected graph G;
for o_i in O do
    G.addVertex(o_i)
for o_a,o_b in O do
    if isDeviceConflict(o_a,o_b) then
        G.addEdge(o_a,o_b)
settledOperations=findMaxClique(G);
deviceIndex=1;
for so in settledOperations do
    specDeviceOperation[deviceIndex]=so;
    deviceIndex+=1;
for o_i in O do
    if o_i is in settledOperations then
        b^device_{i,specDeviceOperation.find(i)} = 1;
        for d_j in D \ {specDeviceOperation.find(i)} do
            b^device_{i,j} = 0;
```

Algorithm 2: Minimum-stage calculation

O: set of all operations;
$o_i.modified\_predecessors$: set of modified predecessors of operation o_i. The predecessors can be defined by the bio-assay itself or the pre-processing;
$o_i.min\_stage$: variable indicating the minimum stage of operation o_i;

```
for o_i in O do
    o_i.min_stage = 1;
    set tempSet;
    for o_pre in o_i.modified_predecessors do
        if o_pre is indeterminate then
            tempSet.insert(o_pre.min_stage);
    if tempSet not empty then
        o_i.min_stage=
          max(o_i.min_stage,max(tempSet)+1);
    for s_id in range(1,o_i.min_stage) do
        b^stage_{i,s_id} = 0;
```

where M is an extremely large auxiliary constant. q_1 and q_2 are auxiliary variables, one of which has to be set to 1 according to (31) if two operations o_a and o_b are bound to the same device.

For example, suppose that operation a and operation b are bound to the same device. In this case, the auxiliary variables q_1 and q_2 will be set to 1 and 0, or 0 and 1, respectively, according to constraint (31). Constraints (29) and (30) will thus become

$$v_a^{d.start} \geq v_b^{d.end}, \tag{32}$$

$$v_b^{d.start} + M \geq v_a^{d.end}. \tag{33}$$

and prevent the overlap of time intervals $[v_a^{d.\text{start}}, v_a^{d.\text{end}}]$ and $[v_b^{d.\text{start}}, v_b^{d.\text{end}}]$.

Device consistency means that an operation can only be bound to a device that satisfies all its requirements. For example, some operations can only be performed in the device containing heating units, and some operations cannot be performed in the device that is too small due to the large volume of solution required. Related constraints are widely applied in many conventional HLS models [14], [15], [17], [18], and thus we omit an exhaustive description of them in this paper.

5) Modelling objective

We can model the total duration of the nominal schedule as the completion time of the last determinate execution stage, i.e. $v_{|\mathbb{O}|+1}^T$. It is noteworthy that if several indeterminate operations have been assigned to the same determinate execution stage, the number of actual stages may be smaller than $|\mathbb{O}|+1$. In this case, the duration of the redundant stages will be automatically set to 0 by the MILP solver, and $v_{|\mathbb{O}|+1}^T$ will be equal to the completion time of the last stage with non-zero duration.

Thus, we set the objective of our MILP model as follows:

$$\textbf{Minimize:} \qquad \alpha \cdot v_{|\mathbb{O}|+1}^T + \beta \cdot v_d, \tag{34}$$

in which v_d stands for the number of used on-chip devices and is modelled with the device-related constraints. α and β are adjustable weight coefficients which allow us to control the optimization priority.

C. Problem Space Reduction

Solving an MILP model is $\mathcal{NP}$-hard for an optimization problem. To ensure the efficiency of our method, we propose two approaches to trim the redundant problem space without harming the optimality of our solutions.

1) Device specification

We propose an algorithm to estimate the least number of required devices considering bio-assay-specified constraints. The core idea is to detect operations that must be performed in different devices. First, we detect operations with different device requirements, such as volumes, shapes, and accessories like heating units. In particular, operations with mutually exclusive device requirements must be assigned to different devices. For example, among all the operations, if there are an operation that needs a ring-shaped mixer as its container and another operation that needs a reaction chamber as its container, we need to prepare at least one ring-shaped mixer and one reaction chamber to perform the assay. Second, we detect operations that need to be performed with overlapping schedules, such as in parallel. These operations also need to

be assigned to different devices. Any operations that should be assigned to different devices are said to be *mutually exclusive*.

The detailed steps of our algorithm are shown in Algorithm 1. Specifically, we first create an undirected empty graph G and insert all operations as vertices into G. If two operations are mutually exclusive, we add an edge between the vertices representing the two operations. If G contains a clique consisting of n vertices, we need to prepare at least n devices to execute all the operations. The maximal cliques can be found by several algorithms such as *Enumeration algorithm* and *Bron–Kerbosch algorithm* [19].

In addition to estimating the least number of required devices, our algorithm can also be used for model reduction. Specifically, when we construct the n devices for the n operations in the maximal clique, we also determine their binding relation so that some operation-device-mapping variables $b_{o,j}^{device}$ are set to constant values. Note that this determination will not harm the optimality of our solutions, considering that each operation eventually has to be assigned to a device that meets all the requirements. Besides, if an operation is bound to a device, it means that the other devices will not be used to carry out this operation. We thereby set some more variables to constants.

2) Earliest-stage calculation

In our method, we create determinate execution stages and assign operations to the stages. Here, we propose an algorithm to calculate the earliest stage that an operation can be assigned to. The algorithm is described in Algorithm 2. Specifically, if a parent operation o_p is determinate, its child operation o_c can be assigned to the stage that o_p is assigned to, but if o_p is indeterminate, o_c can only be assigned to the next stage at the earliest. Based on this observation, we apply BFS to check all the operations in the sequencing graph. For an operation o_1, if all its predecessors are determinate operations, the earliest stage that o_1 can be assigned to is the first stage. If o_1 itself is an indeterminate operation and it has a child operation o_2, then the earliest stage that o_2 can be assigned to is the second stage. Similarly, if o_2 is an indeterminate operation and it has a child operation o_3, the earliest stage that o_3 can be assigned to is the third stage. By applying this algorithm, some operation-stage-mapping variables $b_{o,k}^{stage}$ and other related variables can be set to constants.

VI. Experimental Results

We implemented our algorithm in C++ and run it on a computer with 2.9GHz CPU. The mathematical model was solved by the MILP solver Gurobi.

We compared our method with the state of the art HLS method proposed in [15], which was also the only method that could model operations with indeterminate duration. In addition, we compared our method with the representative HLS method proposed in [9], which however could not support indeterminate operations. To enable the method of [9] to support indeterminate operations, we integrated the modification algorithm in [15] to the method. We applied our input preprocessing approach to revise the sequencing graph so that the methods could deliver feasible scheduling results. Since the state of the art methods did not consider the storage of indeterminate products, there were potential risks that the binding results synthesized by the state of the art method might not be feasible. We ignored this concern regarding the state of the art method in our following comparisons.

Table II: Three Real-World Flow-Based Biochip Applications

name of application	# of operations	# of indeterminate operations
kinase [20]	8	0
cDNA [21]	6	1
RTqPCR [22]	6	1

As shown in table II, we tested the HLS methods with three real-world flow-based biochip applications: kinase [20], which consists of 8 operations and no indeterminate operation; cDNA [21], which consists of 6 operations including one indeterminate operation; and RTqPCR [22], which consists of 6 operations including one indeterminate operation. We replicated each assay multiple times to investigate the performance of the HLS methods at different scales. In our experiments, each synthesis was done within two hours, and thus we omit the program runtime information in the tables and figures for simplicity. Figure 4 shows the comparison results.

Specifically, we compare the number of used devices and the total duration of the assay under the same device constraints. In order to compare the performance of the methods in the case of a sufficient number of devices, we set the initial number of devices to be far greater than the minimum feasible device number. For the kinase assay which does not involve any indeterminate operation, our synthesis results are comparable to the state of the art method [15], while the method [9] needs more total duration. Our method uses slightly more devices or requires longer duration for some test cases, because we accurately model the storage of intermediate products, which is neglected by the state of the art method. For the other two test cases that involve indeterminate operations, our method achieves up to 40% improvement in minimizing the assay duration compared to the state of the art method [15]. Specifically, in the second case, cDNA, both methods [15] and [9] have exactly the same performance on both optimization metrics. Our method needs the same number of devices, but our total duration is much shorter than theirs. For the third test case, our method requires fewer devices when the original assay is replicated for six, seven, and eight times. Regarding the total duration, our duration is stable and the minimum, while the duration of the state of the art method [15] fluctuates. When the number of indeterminate operations is odd, the total duration of the state of the art method is greater than when the number is even. This is because our nominal scheduling method allows us to synthesize the HLS problem as a whole and thus we can find the global optimal solution, while the state of the art method partitions the HLS problem into several sub-problems and thus can only achieve a combination of local optimal solutions.

Next, we investigate the performance of the HLS methods with respect to different device constraints. We replicate the RTqPCR assay for four times and use the new assay as the test case. We set the maximum number of devices to 3, 4, 5, 6, 7, and 8, i.e. the number of devices used for the application cannot exceed these numbers. Figure 5 shows the comparison results.

Specifically, as our algorithm does not partition the HLS

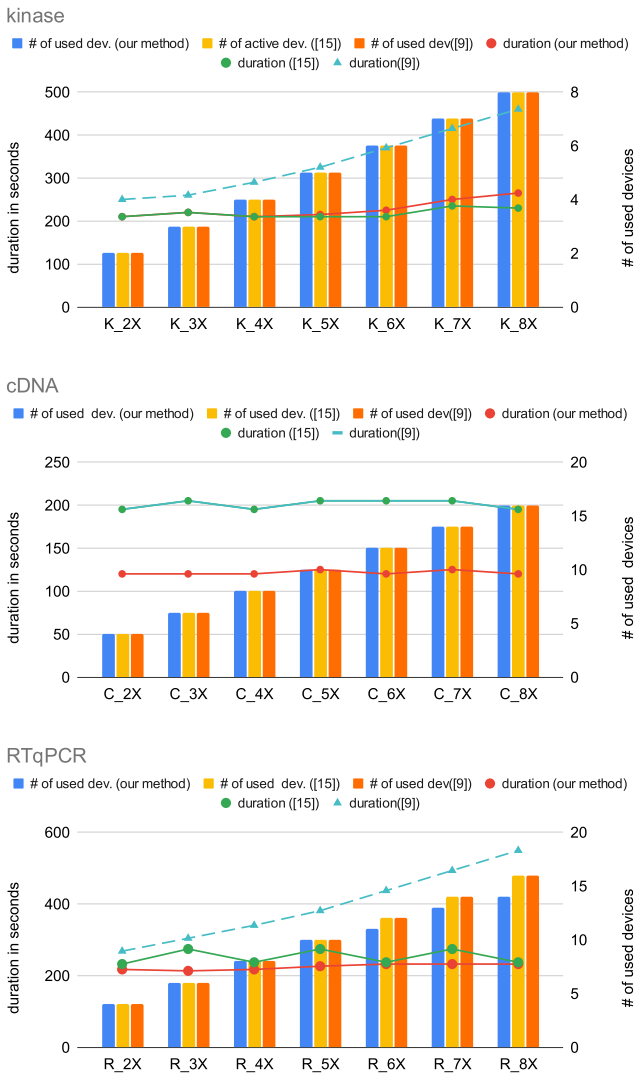

Figure 4: Comparison between our method and the state of the art methods for synthesizing each assay at different scales.

problem, we are able to synthesize the devices considering the whole assay. On the other hand, the state of the art methods can only synthesize the devices considering each sub-assay, which limits the maximum capacity of the devices that can be used for carrying out the assay. Thus, our method in genral achieves $20\% - 45\%$ improvement in minimizing the total duration compared to the state of the art method [15].

At last, we investigate the HLS methods using some synthesized test cases by combining different assays, as shown in Figure 6. As the operations of the synthesized test cases have weaker dependencies and diverge more in device requirements, the optimization space becomes larger. We compare the performance of the methods by limiting the number of devices to the minimum feasible number for test case. The minimum feasible number of devices is the least number of devices to carry out an assay regardless of the total duration. As shown in Figure 6, the minimum feasible numbers of devices are the same for different methods for each synthesized test case. Even

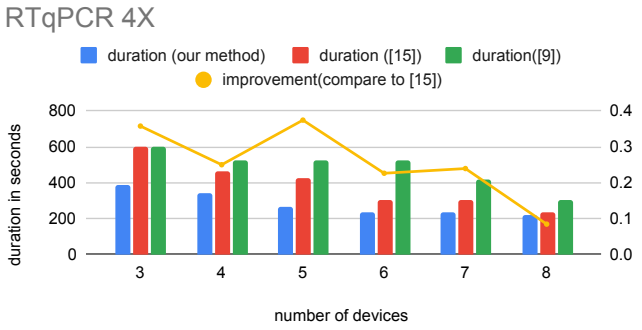

Figure 5: Comparison between our method and the state of the art methods for synthesized test cases.

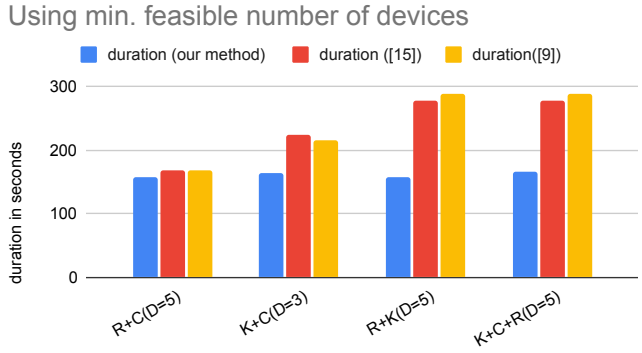

Figure 6: Comparison between our method and the state of the art methods for synthesizing each assay using minimum feasible number of devices.

with the same number of devices, our method outperforms the state of the art methods in terms of the total duration. This demonstrates that our method is generally applicable despite the variation in dependency and device requirements.

VII. Conclusion

In this work, we have proposed a novel high-level synthesis approach for flow-based microfluidic biochips. Specifically, we have proposed to revise the input sequencing graph based on relative scheduling to avoid potential conflicts caused by indeterminate operations and time constraints. Furthermore, we have proposed a nominal scheduling approach to model the high-level synthesis problem as a whole despite the indeterminacy of operation duration. We also proposed two model reduction approaches to improve the efficiency of the synthesis. Our method is guaranteed to achieve the global optimal solution of the scheduling and binding results. Experimental results have demonstrated that our method significantly improves the device usage and minimizes the total duration of the input applications by up to 40%.

VIII. Acknowledgment

This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Reseach Foundation) Project Number 456006534.

REFERENCES

[1] J. A. White and A. M. Streets, "Controller for microfluidic large-scale integration," *HardwareX*, vol. 3, pp. 135–145, 2018.

[2] Y. Wang *et al.*, "An integrated microfluidic device for large-scale in situ click chemistry screening," *Lab on a Chip*, vol. 9, no. 16, pp. 2281–2285, 2009.

[3] H. Huang and D. Densmore, "Integration of microfluidics into the synthetic biology design flow," *Lab on a Chip*, vol. 14, no. 18, pp. 3459–3474, 2014.

[4] R. Gomez-Sjoeberg *et al.*, "Versatile, fully automated, microfluidic cell culture system," *Anal. Chem*, vol. 79, pp. 8557–8563, 2007.

[5] 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.

[6] T.-Y. Ho, "Design automation and test for flow-based biochips: Past successes and future challenges," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 650–654.

[7] Y. Marcy *et al.*, "Nanoliter reactors improve multiple displacement amplification of genomes from single cells," *PLoS genetics*, vol. 3, no. 9, pp. 1702–1708, 2007.

[8] A. R. Wu *et al.*, "Automated microfluidic chromatin immunoprecipitation from 2,000 cells," *Lab on a Chip*, vol. 9, no. 10, pp. 1365–1370, 2009.

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

[10] W. H. Minhass *et al.*, "Architectural synthesis of flow-based microfluidic large-scale integration biochips," in *Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems*, 2012, pp. 181–190.

[11] ——, "Scheduling and fluid routing for flow-based microfluidic laboratories-on-a-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 615–628, 2017.

[12] T.-M. Tseng *et al.*, "Storage and caching: Synthesis of flow-based microfluidic biochips," *IEEE Design & Test*, vol. 32, no. 6, pp. 69–75, 2015.

[13] C. Liu *et al.*, "Transport or store? synthesizing flow-based microfluidic biochips using distributed channel storage," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[14] M. Li *et al.*, "Sieve-valve-aware synthesis of flow-based microfluidic biochips considering specific biological execution limitations," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 624–629.

[15] ——, "Component-oriented high-level synthesis for continuous-flow microfluidics considering hybrid-scheduling," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[16] D. C. Ku and G. De Micheli, "Relative scheduling under timing constraints: Algorithms for high-level synthesis of digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 6, pp. 696–718, 1992.

[17] T.-M. Tseng *et al.*, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[18] ——, "Reliability-aware synthesis with dynamic device mapping and fluid routing for flow-based microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 1981–1994, 2016.

[19] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[20] C. Fang *et al.*, "Integrated microfluidic and imaging platform for a kinase activity radioassay to analyze minute patient cancer samples," *Cancer research*, vol. 70, no. 21, pp. 8299–8308, 2010.

[21] J. F. Zhong *et al.*, "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.

[22] 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.