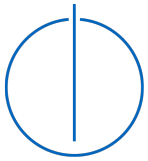
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

High-Level Synthesis for Microfluidic Large Scale Integration Considering Hybrid-Scheduling

Fangda Zuo



DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

High-Level Synthesis for Microfluidic Large Scale Integration Considering Hybrid-Scheduling

High-Level Synthesis für Microfluidic Large Scale Integration angesichts Hybrid-Scheduling

Author:	Fangda Zuo
Supervisor:	Prof. Dr. Andreas Herkersdorf
Advisor:	Dr.-Ing. Tsun-Ming Tseng
Submission Date:	15.01.2019

I confirm that this master's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.01.2019 Fangda Zuo

Acknowledgments

I would like to give all of them, who supported me during this work, my special thanks.

Primarily, I would like to give my advisor Dr.-Ing. Tsun-Ming Tseng and Prof. Dr.-Ing. Ulf Schlichtmann warm thanks for providing me this exciting and expressive topic as my master's thesis. With the valuable support of my advisor, I was able to finish this thesis without hassle. I sincerely thank my friends, Mengchu Li, and Ruichen Liu, for helping me improve the quality of my thesis. Besides, I would like to thank Prof.Dr. sc.techn. Andreas Herkersdorf very much for being my supervisor. Without him, I would not be able to do this fascinating topic at the Institute of Electronic Design Automation.

Moreover, I would additionally thank my girlfriend, Wenting Yu, for keeping me motivated for the entire time. It was a dark period for me. I could not overcome difficult situations and accomplish anything without her selfless help.

Abstract

The rapid evolution of microfluidic design development requires matching automated synthesis methods. As a frontier in microfluidic-based biochip design, a top-down approach simplifies the design of integrated microfluidic-based biochip by providing a library of high-level microfluidic components consisting of basic on-chip units with specialized functionalities. More recently, a component-oriented device and operation concept is proposed to enable flexible operation-device mapping and therefore to improve on-chip resource utilization. As the major contribution of my thesis, I present a method to solve binding and scheduling problems in one step considering storage usage and all execution limitations. Moreover, to improve the efficiency and robustness of my method, I propose an algorithm to process input and check its feasibility. Experiments show the feasibility of the method and the efficiency could be further improved by heuristics to handle large numbers of operations.

Contents

1 Introduction

With the increasing demand for massive parallelization, fast analysis and response time, and precise process control in modern synthesis, the miniaturized laboratory called lab-on-a-chip, also known as the biochip shown in figure 1.1, has been developed and evolved rapidly in the past two decades. As an alternative to conventional biochemical laboratories, the biochip is revolutionizing many applications, such as general single cell studies including growth [1, 2], genetic analysis [3], study of protein[4], and chemical synthesis [5].

Microfluidics refers to the science and technology of systems for manipulation of small amounts of fluids on the nanoliter scale and below to control the fluids precisely on a particular platform. The history of microfluidics dates back to the early 1950s when efforts to dispense small amounts of liquid in the nano- and sub-nanolitre ranges were made for providing the basics of today's ink-jet technology [6, 7].

Microfluidic large-scale integration (mLSI) is one of the most advanced microfluidic technologies. MLSI enables the parallel execution of hundreds of assays with multiple reagents in an automated manner [7].

Due to its characteristics, such as low fluid consumption and high selectivity, the microfluidics technology has become attractive for chemical synthesis in both industry and academia [8]. Computer-aided design automation approaches for continuous-flow microfluidics [9] have been proposed in recent years to make this technology scalable and therefore to transfer the technology from an academic environment to

[1] source: Maggie Bartlett, NHGRI

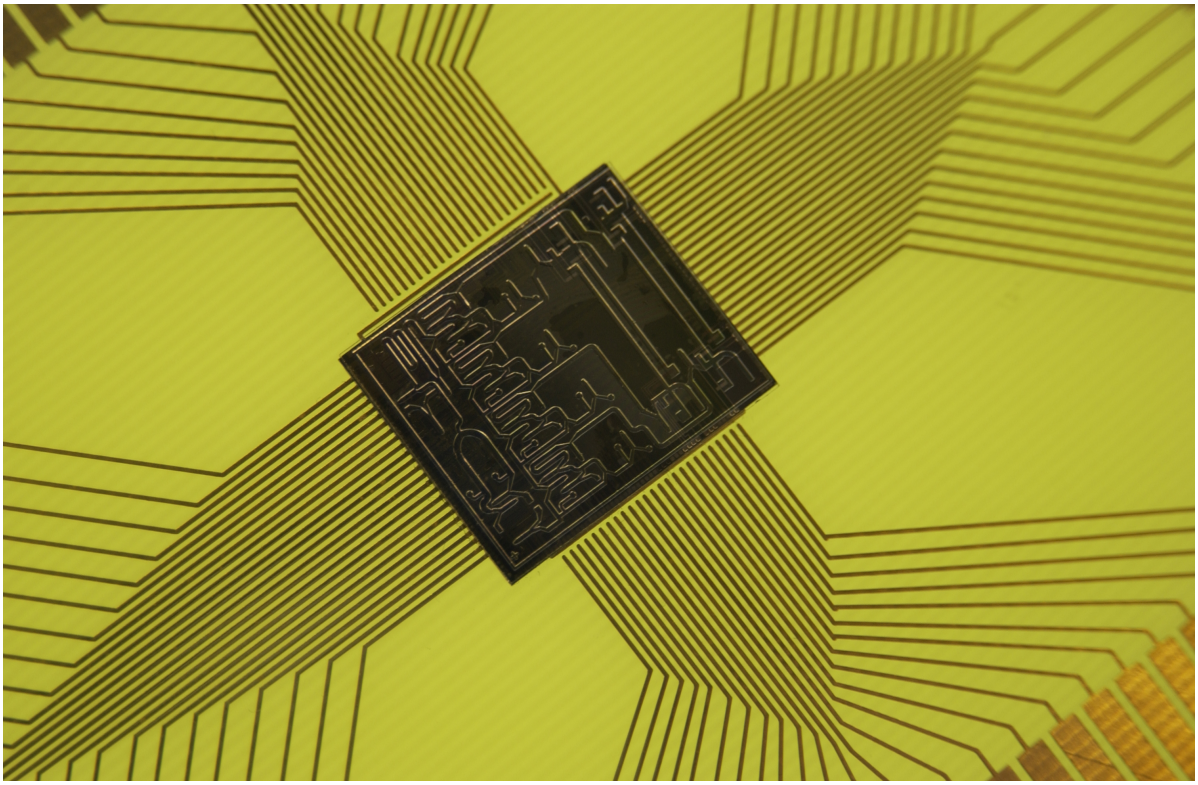


Figure 1.1: Microelectromechanical systems chip, sometimes called "lab on a chip" [1]

an industrial realm which requires both low human resources requirements and low initial investment [10].

With the rapid evolution of microfluidics technology evolved rapidly, more and more synthesis designs that can be processed on the biochip and the corresponding requirements have been proposed, which results in a gap between the biochip concepts and design automation.

First of all, as more and more types of chemical reactions are enabled on biochip due to the evolution of device and component integration, there are several execution limitations of bioassays. For instance, the latest start time of sequencing operations or parallel execution of related operations, are proposed and should be integrated into the design automation concept. As to this point, I will elaborate on these execution limitations later in the following chapter.

Besides, though several types of bioassay operations, for example mixing, detecting, heating as well as filtering, are integrated into automated synthesis, several types, such as storing, are neglected in previous design automation. However, to realize the biochip from the possible needs of enabled bioassay to feasible chip design, it should involve all existing types of operation and support types that may arise in the future with high expandability.

In this work, I present a mathematical modeling method to solve the binding and hybrid-scheduling problems of microfluidics design automation based on component-oriented general device concept that enables functionality- and volume-mapping to operations with or without execution limitations.

Chapter 2 will introduce the background of microfluidic design automation briefly.

Chapter 3 presents my approach to the solution of binding and scheduling problem using the component-oriented synthesis concept.

Chapter 4 shows the experimental results of my method and discusses the reason for the experimental results.

Chapter 5 concludes my attempts and proposes a vision for the future works.

2 Background

A biochemical assay consists of a series of operations. The microfluidic platform comprises an easily combinable set of microfluidic unit-operations that allows assay miniaturization within a consistent fabrication technology [11]. The unit where an operation is executed is called a device, which may be made up of several components with different specialized functionalities. As a base-element in such design automation problems, a device, which is used to perform an operation, should provide all functionalities required by the operation.

In this section, the background of biochip design automation will be introduced, and then the motivation of my work will be further discussed, the problem formulation of this work will be defined at the end of this section.

2.1 High-Level Components

As hundreds or thousands of components are integrated on a single chip, the challenge to design an integrated microfluidic system becomes more and more complex. As an exciting frontier in microfluidic-based biochip design, the top-down approach simplifies the design of integrated microfluidic-based biochip by providing a library of microfluidic components [7]. In order to have a comprehensive understanding of biochip design, different types of components that are integrated on the chip will be briefly reviewed.

2.1.1 Micromechanical Valve

A valve is the basic unit of microfluidics to handle the fluid. Early micromechanical valves were based on silicon microelectromechanical systems (MEMES) technology. However, due to the complexity of highly integrated devices, the micromechanical valves based on this technology is difficult to produce [12].

More recently, the polydimethylsiloxane (PDMS) pneumatic micro-valves [13] are developed by using the soft lithography techniques [14]. Also, the first mLSI was realized by using the multilayer soft lithography (MSL) to produce the monolithic membrane valves. Since then, microfluidic biochip has proliferated, and it is now even possible to integrate one million valves on a single chip [15].

Basically, a valve consists of two types of channels: the control channel and the flow channel. These two channels intersect orthogonally, between which a membrane is formed as a demarcation line of their corresponding layers. As shown in figure 2.1(a), if the control channel layer is on the top of the flow channel layer, a push-down valve is formed. Alternatively, a push-up valve shown in figure 2.1(b) will be formed.

Also, as shown in figure 2.1(c), a three-layer device with both push-up and push-down valves can be formed by combining the valves to increase valve density and simplify the channel routing.

2.1.2 Pump

A pump refers to a chip component consisting of a linear array of valves. As shown in figure 2.1(d), the valves actuate in sequence to provide the pressure for fluid movement. Suppose 0 and 1 represent the opening and the closing of the valve respectively, three valves are actuated in the pattern 101, 100, 110, 010, 011, 001. In this manner, fluid in the flow channel can be push from one side to the other side.

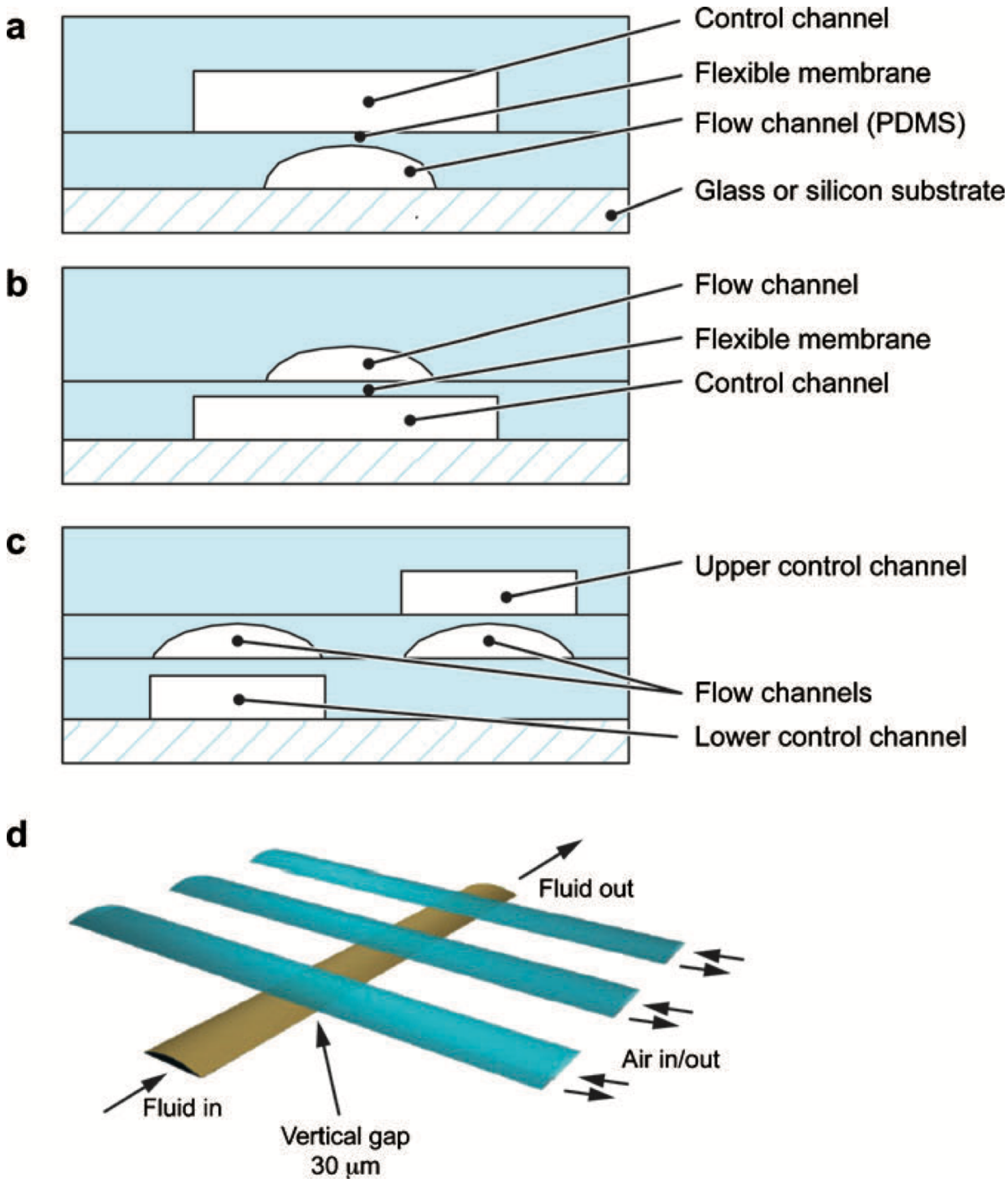


Figure 2.1: Monolithic micromechanical valves [7]: (a) Two-layer polydimethylsiloxane (PDMS) Push-Up Valve. (b) Two-layer PDMS Push-Down Valve. (c) Three-layer device with both push-up and push-down valves. (d) Schematic of a linear peristaltic pump using three valves in a series.

2.1.3 Sieve Valve

A sieve valve is a particular component based on PDMS technology for liquid transportation. Unlike the conventional push-up valve shown in figure 2.2(a) (marked by blue color), which should be entirely closed by the control valves, a sieve valve shown in figure 2.2(b) (marked by blue color) can be partially closed. By closing the sieve valve, a gap in the valve will be formed, which is small enough to filter larger particles, but at the same time, smaller particles and liquid can still pass through.

As shown in figure 2.2(c)(d), the anion exchange beads are stacked by the closed sieve valve shown in inset (b). All the valves marked with X are closed. Since the conventional valve at the bottom is open, the liquid can still flow through; thus, the target is concentrated from a 40ml volume into less than 400nl [16] while retaining the same amount of cells. The beads can be easily transported by opening the sieve valve and pushing them with a new liquid. In this manner, the sieve valve can be used to produce highly-concentrated target particles, which is important for cell-based operations.

2.1.4 Mixer

A mixer is one of the most critical microfluidic devices since the mixing of reagents is one of the fundamental requirements of the bioassay. One of the conventional mixer design is to combine a rotary geometry with the pumps [18]. As shown in figure 2.3, the mixing loop is sealed after the reagents are loaded into the device. When the pump is activated, the closer the part of the liquid is to the center of the ring, the faster it flows. As a result, the interface between two reagents is increased, and consequently, the reagents can be mixed efficiently.

However, the mixers are not necessary for mixing operations due to the rapid evolution of component designs. For example, a sieve valve mentioned above could be used for mixing operations. The figure 2.4(a) shows the micrograph of a chip

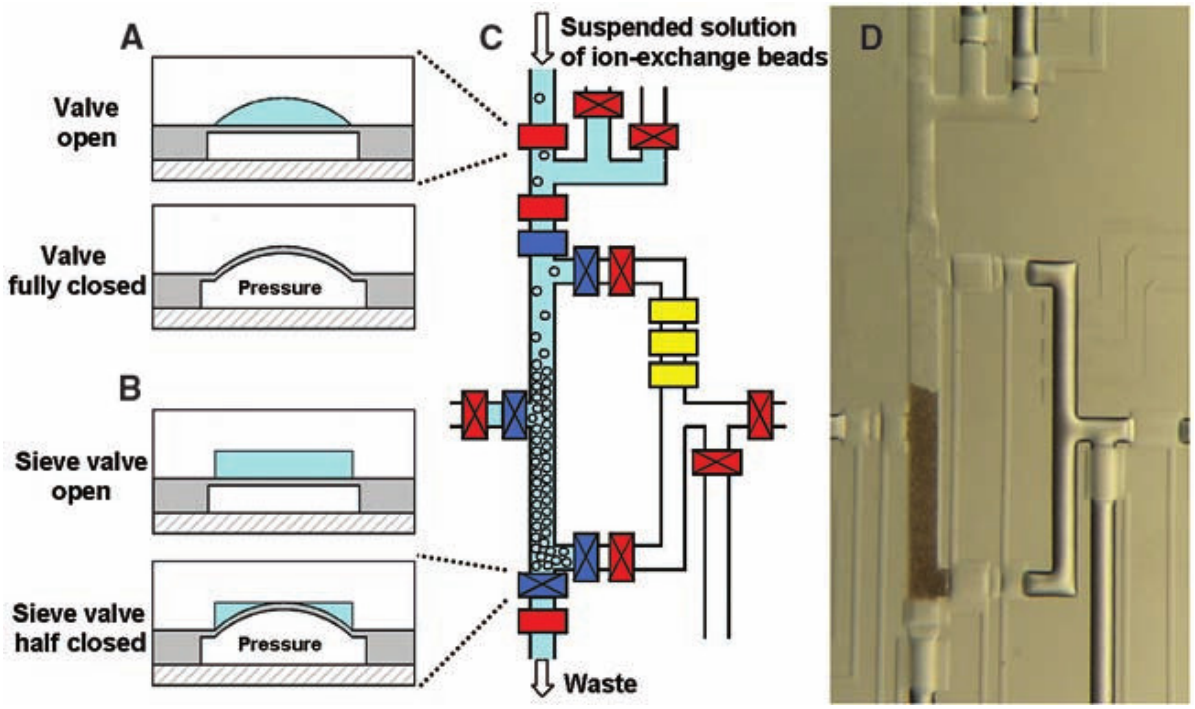


Figure 2.2: Sieve valve function [17]: (a) Regular valve. (b) Sieve valve. (c) Stacked beads. (d) Snapshot of (c).

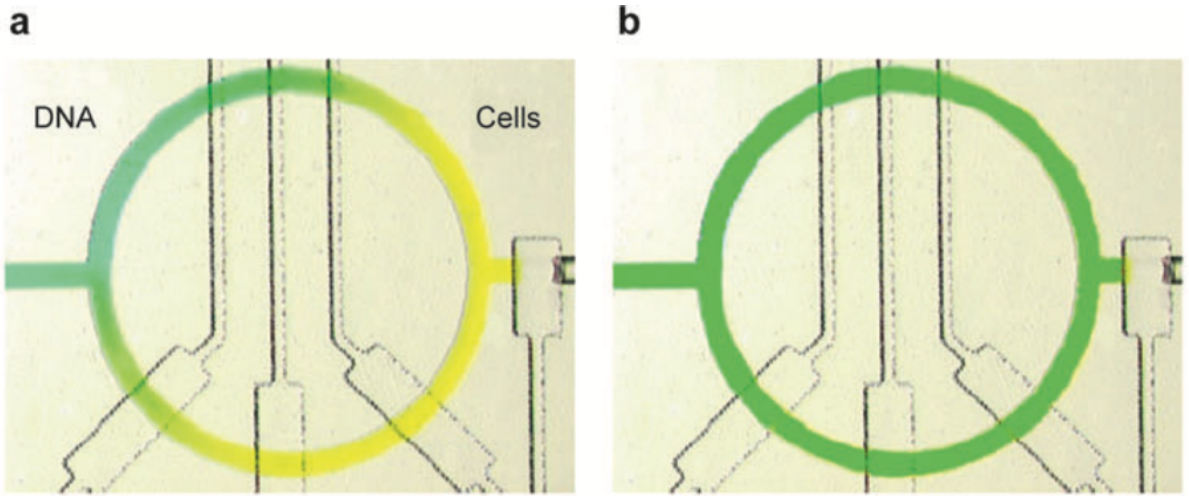


Figure 2.3: [7] (a) Rotary micromixer, DNA liquid within the mixing loop is marked by green color and cells liquid is marked by yellow color. (b) After actuating the pump two colored liquids are completely mixed.

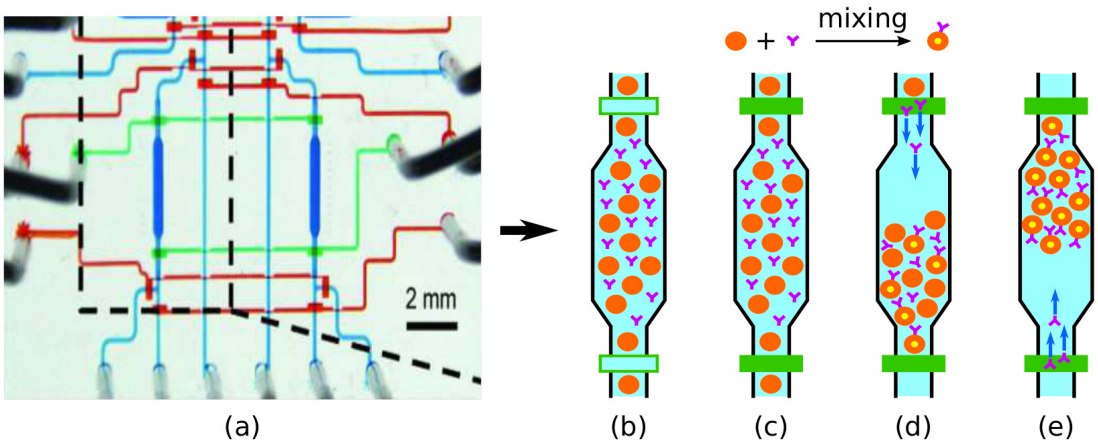


Figure 2.4: Sieve valve function [19]: (a) Micrograph of a chip designed for a kinase activity assay. (b)-(e) Flow reversal protocol enables liquid samples to pass through the bead column in both directions alternatively.

designed for a kinase activity assay. The flow channels marked by blue color are controlled by the sieve valves marked by green color. Since the sieve valves can be partially closed to filter larger particles while allowing smaller particles and liquid passing through, as shown in figure 2.4(b)-(e), the beads are pushed by input samples in both directions iteratively. Consequently, the mixing of beads and samples are performed without a conventional mixer. Moreover, differing from the conventional mixer which is difficult to support the mixing of large volumes of input samples effectively, the sieve valves are used to produce highly-concentrated particles, and thus, the mixing of large volumes can be executed by using sieve valves.

Besides the particular example of chip design, current designs demonstrate that the mixing operations can also be performed in a mixer that is integrated with several other microfluidic components which can be used for other operations to utilize the use of chip resources, such as detecting [20], heating [21], and washing [17]. As shown in figure 2.5, the mixers are integrated with cell-separation modules, which as shown in Insert 2 are U-shaped (marked by blue color). They are separated from the ring-shaped component (*cell-separation module*: blue, *the rest of mixer*: yellow) by the

portion valves marked by black color. When the portion valves are opened, the blue U-shaped cell-separation modules and the yellow U-shaped parts are combined to form the complete conventional ring-shaped mixers for mixing operations. However, after the portion valves are closed, the blue U-shaped parts will be separated from the mixers and serve as cell-separation modules for cell-isolation operations.

2.1.5 Cell Trap

There are two major methods for single cell isolation. One is to use cell-separation modules as shown in figure 2.5. To capture single-cells, the distance between two floating cells is adjusted by changing the cell concentration and flow rate. Using this method, the float can be captured separately by using cell-separation modules.

The other method is called cell traps. The cell traps vary in shapes and sizes to fit and hold cells; some U-shaped PDMS traps are shown in figure 2.6(a)-(d). The development of cell traps enables a large number of cell isolation operations which can be executed in a parallel way. However, as shown in figure (e), the probability that a cell trap captures exactly one cell is about 53%. Therefore the number of times that the single-cell capturing operation needs to be executed cannot be determinate.

2.1.6 Heating Pad

As shown in figure 2.7(b), the component for heating operations consists of a two-layer fluid-handling part and a heating circuit. The ring-shaped fluid channel is sealed after the sample liquid is loaded. With different actuation pressure and pumping frequency, the sample liquid is transported around the loop and heated.

2.1.7 Optical System

An optical system is a general component used to take images, which help analyze operation results. The system usually consists of a light source and a detector.

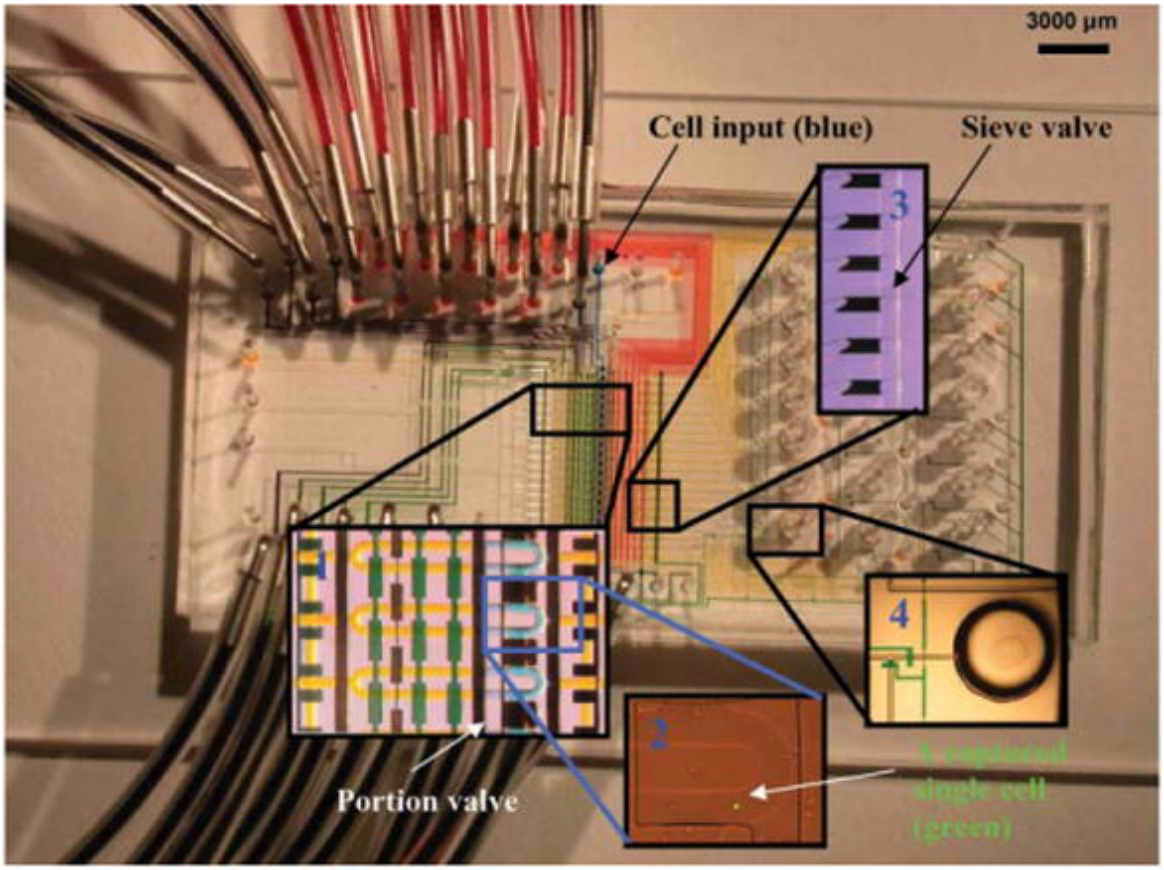


Figure 2.5: Single-cell mRNA extraction microfluidic device. (Insert 1) Micrograph of mixer disign integrated with cell-separation modules by using portion valve. (Insert 2) Cell-separation module with captured single-cell. (Insert 3) Bead columns next to the sieve valve to synthesize cDNA. (Insert 4) Collection wells to store beads with attached cDNA. [20].

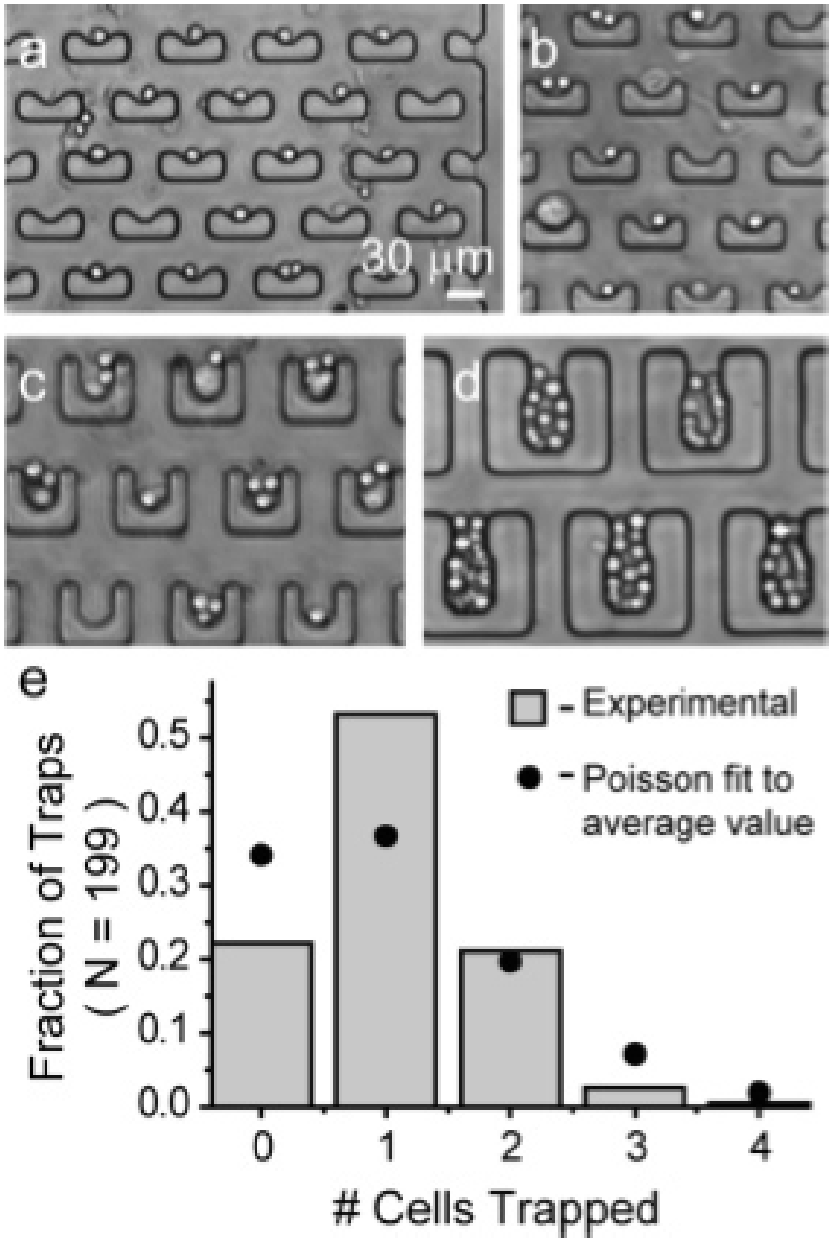


Figure 2.6: Single-cell isolation using cell trap [22]. (a)-(d) Varying geometry of cell traps, the depth varied as 10, 15, 30, and 60 μm. (e) The distribution of trapped cells for the geometry shown in (a).

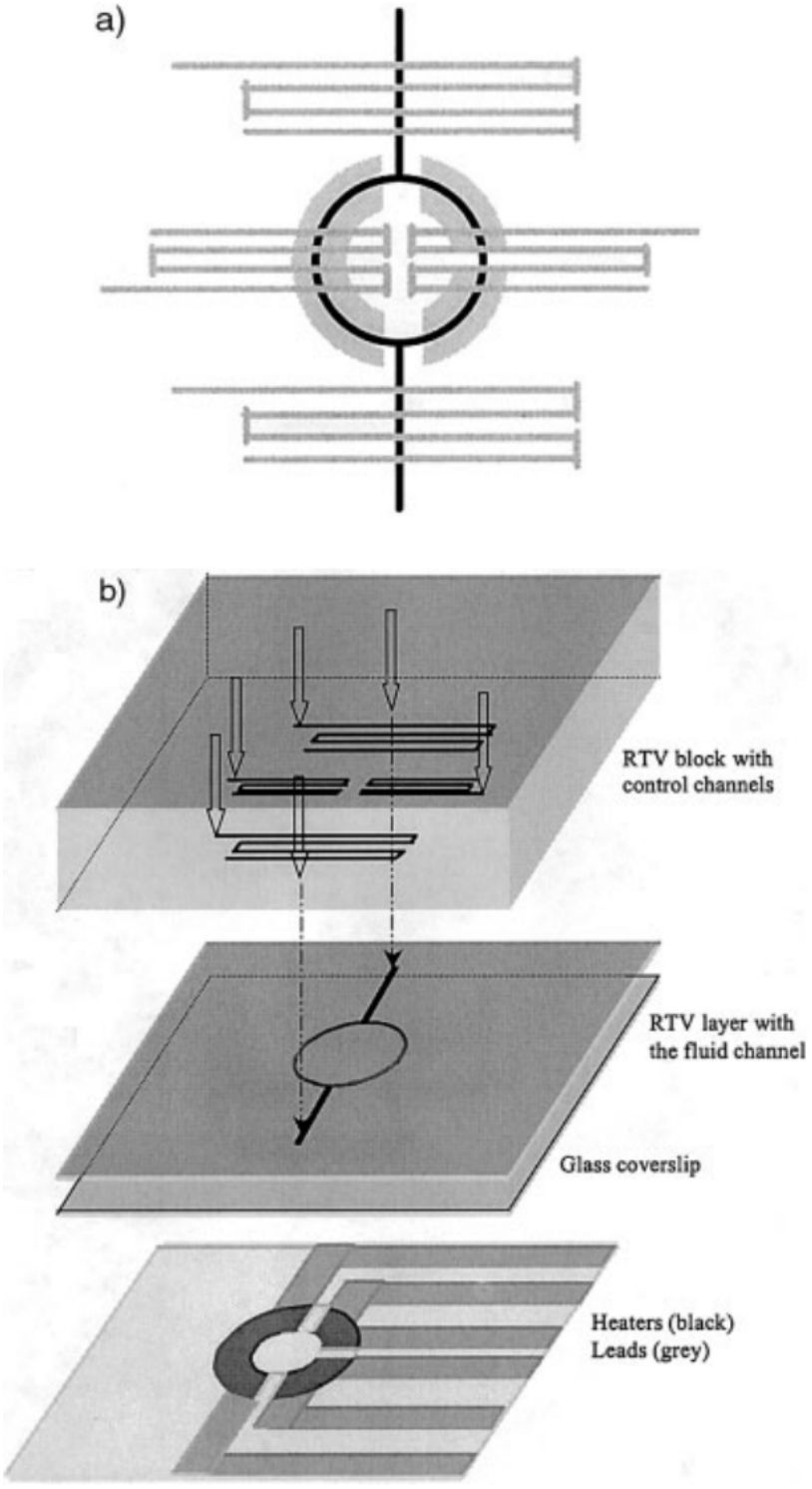


Figure 2.7: Heating pad [21]. (a) Schematic of the heating pad (top view). (b) Assembly of the rotary microchip and the heating pad.

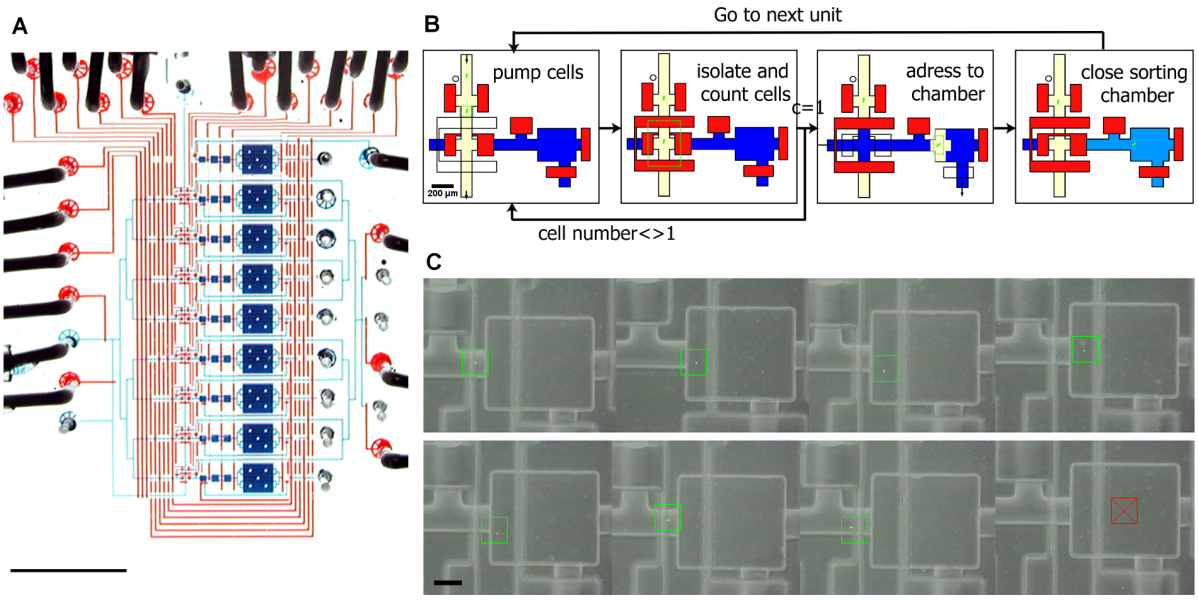


Figure 2.8: Photograph of a Single-Cell Isolation [23]: (a) micrograph of chip architecture (scale bar 5 mm). (b) Schematic diagram of the automated sorting procedure. (c) A color combination of a phase contrast image (gray) and a fluorescence image (green). A red crossed square indicates the absence of cell (scale bar 100 μm).

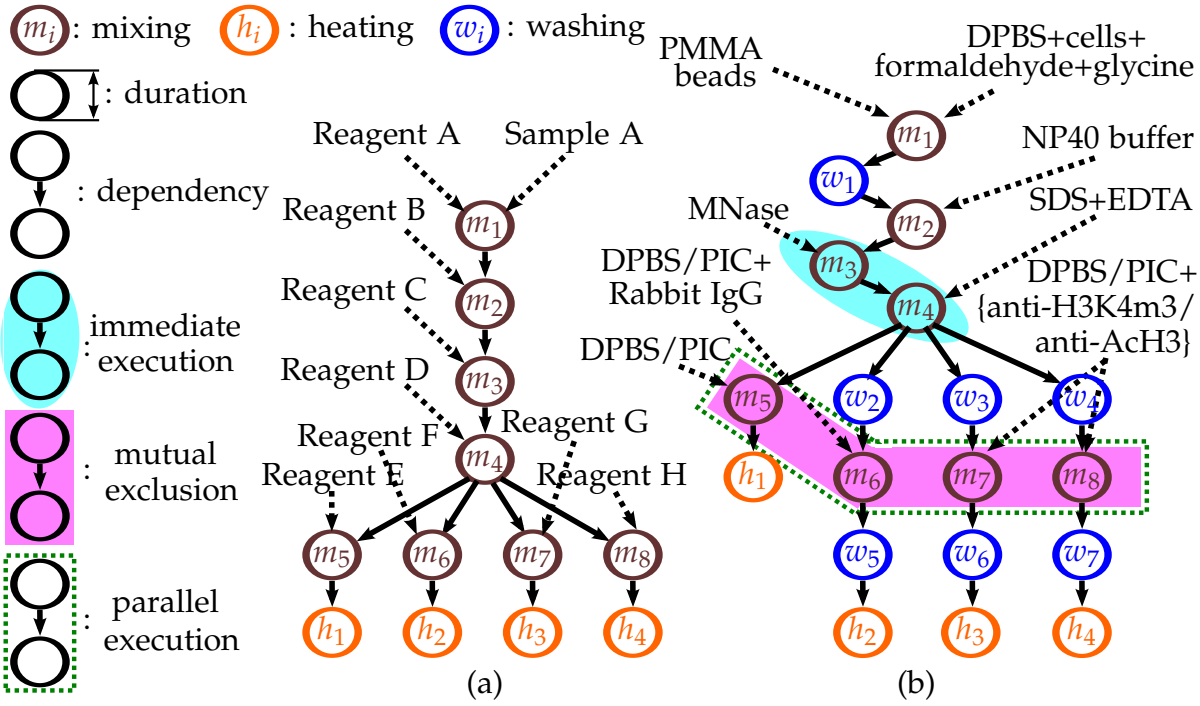


Figure 2.9: Sequencing graphs of Chromatin Immunoprecipitation (ChIP) [17]: (a) Simplified graph. (b) Realistic graph

As shown in figure 2.8(c), right after the detection of a fluorescent signal, an image is taken and analyzed to count the number of cells.

2.2 Operation properties and execution limitations

ChIP (Chromatin Immunoprecipitation) [24] is an assay used to study interactions between protein and DNA. I take the ChIP assay as an example (figure 2.9) to introduce the operation properties and execution limitations.

2.2.1 Operation properties

The operation means the manipulation of biochemical inputs. On continuous-flow based microfluidic biochips, each operation has the following basic properties:

1. **Operation type**: the type of operations, such as mixing, heating, washing, as well as detecting. The nodes marked by different colors shown in figure 2.9(b) represent different types of operation. The components mentioned above are designed to perform these operations.

2. **Input volume**: the volume of the input sample to get the required synthesis result. Since the different component designs have their own capacities to execute operations with a certain volume of input sample, different devices may be needed to execute the operations, which share the same operation type but different input volumes.

3. **Execution duration**: the time that an operation is once fully executed, whether it is successful or not. Some operation types, for instance single-cell-capturing operation mentioned in section 2.1.5, are not necessarily successful.

4. **Dependency**: the relationship when an operation o_c takes the output sample of another operation o_p as its input sample. Similarly as that in [19], the operation o_c is defined as the child operation of operation o_p, and operation o_p as the parent operation of operation o_c. As shown in figure 2.9(b), each solid edge represents a dependency relationship between its source and sink node. Additionally, the successful execution of a parent operation limits the earliest start time of its child operations.

2.2.2 Execution limitations

1. **Latest start time**: as shown in figure 2.9(b), the node m_3 specifies the upper bound of the start time of m_4, and I define the limitation as the latest start time. After the Micrococcal Nuclease (MNase) sample is loaded to execute the operation m_3, the DNA is digested from cells into fragments. To prevent over-digestion of DNA and lyse cells completely, the operation m_4 is performed

with the Sodium Dodecyl Sulfate/Ethylenediaminetetraacetic acid (SDS/EDTA) buffer. In case the DNA is over-digested, the DNA ends would be harmed, and the required dinucleosome signal would disappear [25]. Thus, the start time of operation m_4 is closely related to the end time of operation m_3 and cannot be arbitrarily postponed.

2. **Parallel execution**: as shown in figure 2.9(b), m_4 is the final operation of the chromatin (Ch) process, and its output sample is used to perform the immunoprecipitation (IP) process. The operations m_7 and m_8 take Dulbecco's phosphate-buffered saline/Protease Inhibitor Cocktail (DPBS/PIC) buffer with target antibodies while the operation m_5 as a reference operation takes no antibody and the operation m_6 as negative control takes antibodies that specify no target proteins. Thus, to compare the outputs of these operations fairly, they need to be executed in parallel [24, 26].

3. **Indeterminate execution duration**: some operations are not necessarily successful, and therefore the execution duration of these operations to produce the desired result cannot be confirmed. For example, as mentioned in section 2.1.5 and section 2.1.7, the cell traps are used for single cell isolation and the optical systems are used to detect whether the cell traps have captured exactly one cell or not. If the number of captured cells does not equal one, the operation needs to be executed once again. The experimental result 2.6 shows that the cell trap only succeeds in capturing precisely one cell with a probability of 53%. Thus, the operation could be executed repeatedly, and the execution duration of the operation become indeterminate. For short, I call the operations with indeterminate execution duration *indeterminate operation* in the same way as in [19].

2.3 Component-oriented Synthesis Concept

In previous works, the operations and devices were classified into different types by their functionality, and an operation could be assigned to a device only if their types matched. As a accepted standard, these specifications simplified the scheduling and binding problems.

However, the demarcation lines in the conventional type-classification standards have become increasingly blurred as the component design evolves. For example, as mentioned in figure 2.5(Insert 1), a ring-shaped mixer can be disassembled into two U-shaped parts, one of which can be used as a cell-separation module. Thus, a mixer in this design can support multiple types of operations. Consequently, more flexible operation-device mapping methods are needed.

A component-oriented synthesis concept [19] is proposed to enable the precise description of operations and microfluidic devices. Unlike the type-classification standard mentioned above, which contained assumptions that simplified the problem, the general device concept improves the utilization of on-chip resources and adaption of technological updates by allowing flexible operation-device mapping.

2.3.1 Microfluidic Components

Both container and accessory are the microfluidic components. They are classified by different area cost and processing cost when integrated into a chip.

1. **Container**: refers to a part of a flow channel which enables separate liquid storage and handling. The integration of containers requires both area cost and processing costs:

    a) **Chamber**: a segment of a flow channel that is separated by several valves. The segment is usually separated by two valves. However, as shown in figure 2.2(c), five valves here separate the segment of the flow channel

filled with stacked beads. The flow channels can vary in length and width according to different operating protocols.

b) **Ring**: a ring-shaped flow channel connected end to end to enable circulation flow. According to the definition of chamber, a *ring* is a specialized *chamber*.

2. **Accessories**: refers to a library of component units with a particular function. The accessories require no additional area cost because of the fact that they can be integrated into containers.

   a) **Pump**: a group of valves as mentioned in section 2.1.2 which provide actuation pressures for liquid movement.

   b) **Heating pad**: a component shown in figure 2.7 which supports heating operations.

   c) **Optical system**: the optical components for detecting operations.

   d) **Sieve valve**: a particular valve shown in figure 2.2 which can be partially closed for filtering of particles.

   e) **Cell trap**: a passive component shown in figure 2.6 for cell-capturing.

2.3.2 General Device and Component-oriented operation Definition

Consisting of one container and several accessories, a *general device*, serves as a general platform for operation execution. All high-level components mentioned in section 2.1 can also be described well based on this concept. For instance, a conventional mixer shown in figure 2.3 can be defined as a general device with a ring as the container and a group of pumps as its accessories; the same device is shown in figure 2.5 with an additional separation valve as its accessory serves as a platform for both mixing and cell-separation operation; moreover, as shown in figure 2.2, a

mixing operation can also be performed in a device consisting of a chamber as the container and two sieve valves as its accessories.

Component-oriented operation is likewise proposed to accommodate the general device concept. A *component-oriented operation* includes the following attributes:

1. *container and accessories* required for operation execution;
2. *volume of input sample* is used to select container with specified capacity;
3. *dependency* which indicates related operations;
4. *execution limitations* which specifies the execution time of this operation.

2.4 Problem formulation

The binding and scheduling problem is defined as follows:

- **Input**: specified information about a series of component-oriented operations to perform a particular bioassay.
- **Objective**: minimization of assay execution time and the costs incurred by integrating the required components on the chip.
- **Output**: the bioassay synthesis result specifying scheduling and binding results.

3 Method

In this chapter, a detailed description of my method will be given to solve the problem mentioned above. The method proposed consists of the following parts:

1. Inputs processing for method robust and efficiency,
2. Hybrid-Scheduling approach to solve the problem with the indeterminate operations,
3. Mathematical modeling and optimization of the problem, and
4. Reprocessing of the optimization result.

The flowchart of this method is shown in figure 3.1.

3.1 Input Processing

The input that describes an assay consisting of component-oriented operations is the data in the form of a text file that contains information about the properties and execution limitations of a series of operations. Abstractly, the protocols could be shown in a sequencing graph like the one mentioned in chapter 2. In previous works, the inputs were assumed to be correct and feasible. However, these assumptions might not be valid considering the indeterminate execution duration and other execution limitations. In this thesis, an "infeasible input" is defined as an input

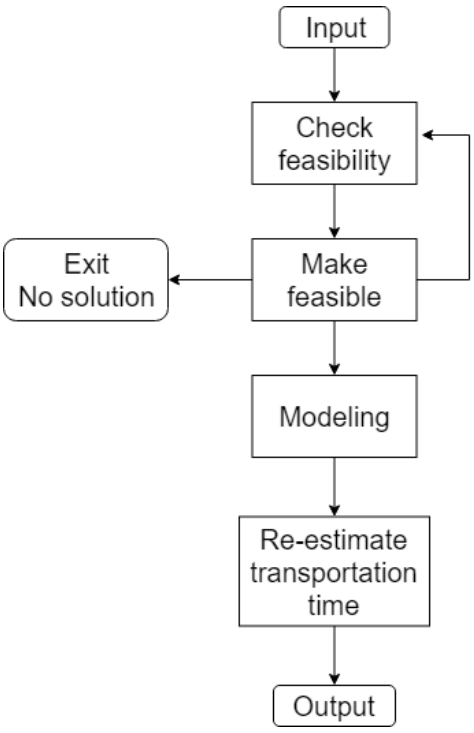


Figure 3.1: Block diagram of my method.

violating the execution limitations, and a "conditionally feasible input" as an input, whose feasibility cannot be assured.

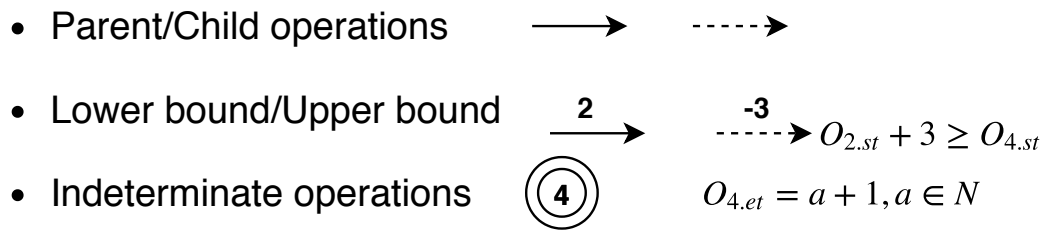
An infeasible input may occur due to a design error or a simple manual error by generating the input files. It is reasonable to check the feasibility of design input since the modern synthesis design can include up to hundred or thousand operations and could cause the waste of time if the optimization program could not find the feasible solutions after running for several hours. In this thesis, several infeasible and conditionally feasible cases will be discussed, and the methods to detect or solve them will be proposed. My goal is to improve the robustness and the efficiency of the method by checking the feasibility of inputs before the beginning of mathematical modeling.

3.1.1 Modified weighted sequencing graph

In this thesis, I propose modified sequencing graph to describe and visualize the input data and to illustrate upper bound limitation and indeterminate operation.

A modified weighted sequencing graph consists of the following elements:

- *Operation and Operation Sequence*: As shown in figure 3.2, in a modified weighted sequencing graph, same as the elements in the original sequencing graph, each node means an operation, and each edge represents the operation sequence or so-called dependency relationship.
- *Execution duration*: The positive weight labeled on solid edges, or forward edges, represents the execution duration of the parent operation.
- *Upper bound limitation*: The negative weight labeled on dotted edges, or backward edges, represents the upper bound of the waiting time of a child operation, which means that the child operation must be started not later than this given time after the end of its parent operation.
- *Indeterminate operation*: The node with a double circle means an indeterminate operation. The weight on the corresponding forward edge represents the time that the operation is executed once regardless of the result.
- *Source and Sink*: Additionally, I added a starting node (source), which is marked as indeterminate operation and has the outgoing flow to all sources in original sequencing graph, to show the execution duration of indeterminate operations in sequencing graph and represent sequencing graphs with multiple sources and sinks as one flow network. Moreover, an end node (sink) is also added, which can be considered as an indeterminate operation and has incoming flow from all sinks in the original sequencing graph.



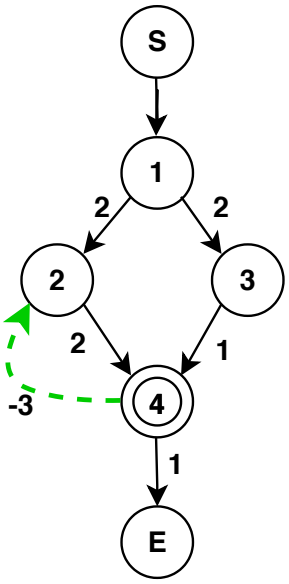


Figure 3.2: Modified weighted sequencing graph, S represents source and E represents sink.

The source and sink are marked as indeterminate since it is intuitive that the start and the end of the synthesis on the chip are triggered by the operator and therefore have no fixed time information. As a network flow, the modified weighted sequencing graph can be used for capability calculation or other graph theory related algorithms.

3.1.2 Cycle in sequencing graph

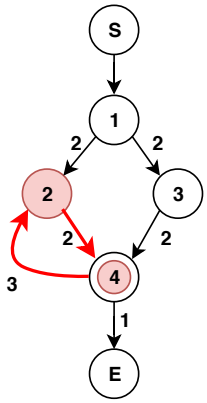


Figure 3.3: Example of cycle in sequencing graph

In graph theory, a cycle as shown in fig 3.3 consisting of node 2 and node 4 in a directed graph $G(V, E)$ can be represented as a subgraph consisting of a set of nodes and edges: $G'(V', E')$, $V' \subseteq V, E' \subseteq E$. A cycle means the predecessor of each node in the cycle is also its successor at the same time.

In this case, the start time of these operations represented by the nodes in the cycle cannot be decided, since the end time of O_a, as a predecessor of O_b, must be no

later than the start time of O_b, and meanwhile, the start time of O_a, as a successor of O_b, should be no earlier than the end time of O_b, causing that the end time of O_a should be no later than the start time of O_a, which is obviously infeasible.

As a solution, I could find the predecessors of each node by performing a Breadth-first search (BFS). The decision to chose BFS is driven by the need for collecting other information including execution duration, etc., in the input processing phase. Once a successor of a node occurs in the set of its predecessor, the entire process will be stopped immediately, and the input will be marked as infeasible.

3.1.3 Exceedance of upper bound

When assuming infinite time and chip area resource, an acyclic series of operations that do not have upper bound limitations is feasible, since one could always find a start time for the child operation greater than all the lower bound of its parents operations, and one could always find a device that is not occupied to execute the child operation. All that is needed is to find an optimized solution. However, this situation becomes different after considering upper bound limitation and the indeterminate execution duration.

As shown in figure 3.4, the start time of operation O_4 depends on the end time of an indeterminate operation O_3, and O_4 must be started not later than a given time after the end of operation O_2. Since the upper bound is a fixed number and the indeterminate operation could be executed repeatedly without the desired result, the upper bound has a probability of being exceeded. This relationship is called conditionally feasible, which means that the feasibility of the input depends on the entire execution duration of the indeterminate operation.

To solve this, I proposed a method inspired by relative scheduling [27]. Since ensuring the feasibility of the result has the highest priority, the start time of the related operations could be postponed as much as necessary to ensure that the upper

Algorithm 1 Input Processing Part I

```
1: feasiblity: the boolean value represents the input feasibilit
2:
3: procedure checkFeasibility()
4:     if not isFeasible() then
5:         feasibility ← makeFeasible()
6:     else
7:         feasibility ← true
8:     end if
9: end procedure
10:
11: procedure isFeasible()
12:     isFeasible ← true
13:     for all O_p as parent operation do
14:         for all O_c as child operation, O_c ∈ O_p.upper_bound do
15:             for all O_pre, O_pre ∈ O_c.indeterminate_predecessor do
16:                 if O_pre ∉ O_p.indeterminate_predecessor then
17:                     isFeasible ← false
18:                     mark O_pre
19:                 end if
20:             end for
21:         end for
22:     end for
23:     return isFeasible
24: end procedure
```

Algorithm 2 Input Processing Part II

```
1:  procedure MAKEFEASIBLE()
2:      isFeasible ← true
3:      for all marked O_pre do
4:          if not addEdge(O_pre, O_p) then
5:              isFeasible ← false
6:              break
7:          end if
8:      end for
9:      return isFeasible
10: end procedure
11:
12: procedure ADDEDGE(U,V)
13:     addedEdge ← true
14:     if O_u ∉ O_v.indeterminate_predecessor then
15:         if O_v ∈ O_u.predecessor then
16:             return false
17:         else
18:             add forward edge(u, v)
19:             add O_u into O_v.indeterminate_predecessor
20:             for all O_w, O_v ∈ O_w.upper_bound do
21:                 addedEdge ← addedEdge ∧ addEdge(O_u, O_w)
22:             end for
23:             return addedEdge
24:         end if
25:     else
26:         return true
27:     end if
28: end procedure
```

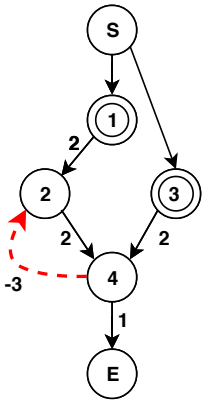


Figure 3.4: Example of upper bound exceedance

bounds will not be exceeded.

The exceedance case occurs between a pair of operations with an upper bound limitation. As proved in [27], assuming $G(V, E_f)$ is acyclic, a sequencing graph G is feasible for the pre-generated schedule only if all predecessors with the indeterminate delay of v belong to the set of all predecessors with the indeterminate delay of u for all edges $(u, v), u, v \in V$.

Hence, for each pair of operations with upper bound limitation, I could compare their predecessors which has indeterminate execution delay. As shown in 3.5 and the pseudo codes Algorithm 1 and Algorithm 2, if a cycle does not occur by doing so, the added edge between the parent operation and the specified indeterminate operation can avoid the exceedance case. Otherwise, the input will be marked as infeasible.

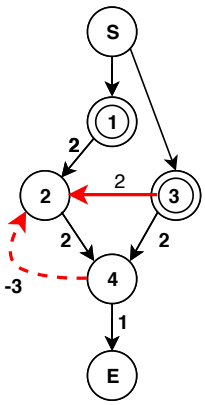


Figure 3.5: By adding edge, a conditionally feasible sequencing graph become feasible.

3.2 An Intuitive Approach

Considering solving the scheduling and binding problems separately, an intuitive approach for the scheduling problem is to set the execution time of the indeterminate operations as a fixed time, which is the execution time of the operation, plus a variable time, which represents the indeterminate execution delay.

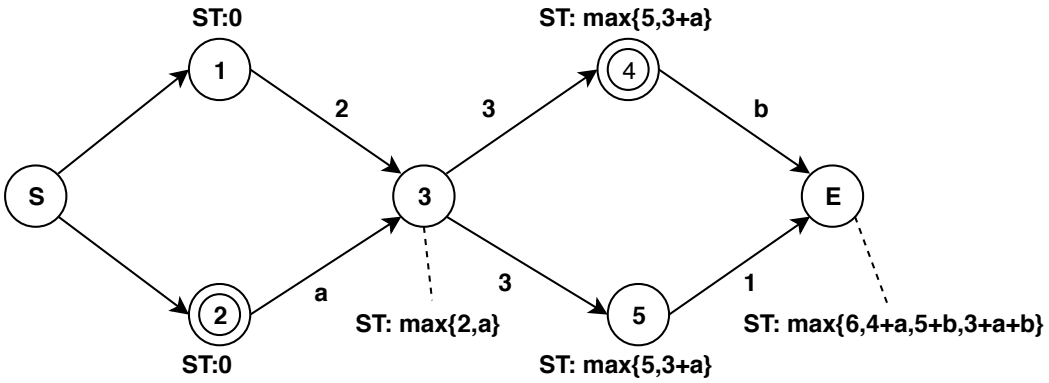


Figure 3.6: The intuitive approach.

The figure 3.6 shows the idea of how to solve the scheduling part separately. The most conventional way to solve the scheduling problem is to find the critical path as presented in the [28]. By calculating the earliest and latest start and end time of operation execution, a critical path to compact the scheduling plan could be found. As a result, the plan can be optimized.

However, the original critical path method is for the operations with fixed execution duration. To make the method work for operations with indeterminate execution delay as well, all execution time of indeterminate operations is set to constant execution time plus a variable represents execution delay.

For example, if an operation O_c has multiple parent operations and at least one of them is an indeterminate operation, the earliest start time of this operation O_c is set to the maximum of all earliest end time of its parent operations for path calculation. The mathematical expression is $O_{c.st.earliest} = \max\{O_{p.et.earliest}\}, \forall O_p \subseteq O_{c.parent}$.

In this intuitive approach, the number of elements in the set $\max\{O_{p.et.earliest}\}$ explodes as the number of operations increases. Because this growth is not linear, this method is not scalable for scheduling of a very large number of operations. Therefore, a method to solve the problem with a large number of operations and meanwhile, solve both scheduling and binding problem at the same time, is still needed.

3.3 Hybrid-Scheduling Approach

In a schedule, the indeterminate operation cannot be allocated to fixed time slots. However, due to the need for device reservation in many applications, the pre-generated schedule is indispensable. To solve this problem, a hybrid synthesis method that includes not only pre-generated schedule step, but also real-time decision step, is proposed. This method is an extended version of the method proposed in [19].

In this method, operations are assigned into several layers to make sub-pre-generated schedules with fixed time slots. In each sub-pre-generated schedule, the explicit start time of included operations is modified by real-time decisions. The operation-device binding problem is solved in these sub-schedules. The indeterminate operations are placed at the end of the sub-schedules, and they occupy the corresponding device until the sub-schedule is considered completed. Then the completion

of the indeterminate operations will trigger the start of the next sub-schedule.

In this manner, the synthesis problems can be solved comprehensively.

3.3.1 Visualization of pre-generated schedule and indeterminate gaps

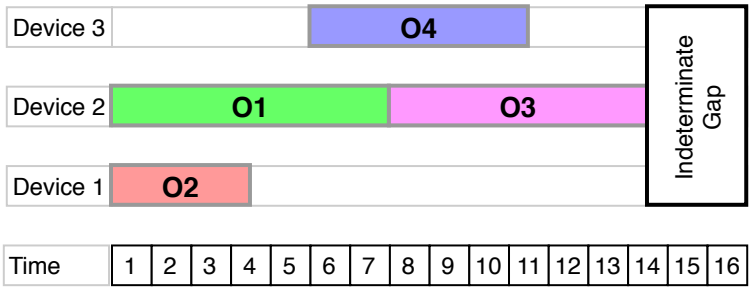


Figure 3.7: Visualization of a schedule output.

The fig 3.7 illustrate my desired output which is the pre-generated schedule. In the coordinate indicator system, the x-axis represents time, and the y-axis represents the device. Each solid block represents an operation-device mapping and the corresponding operation's time consumption.

Since the indeterminate operations have no fixed end time in the pre-generated schedule, their corresponding blocks represents only the first round of execution. Their end time is represented by indeterminate gaps. The left side of an indeterminate gap represents the latest end time of all operations including the first round of indeterminate operations in the current sub-schedule. It is the nominal end time of

this sub-schedule. The right side of an indeterminate gap means the real latest end time of all indeterminate operations in the current sub-schedule and is the real end time of this sub-schedule. The width of the indeterminate gap cannot be optimized since it depends on the number of rounds that the corresponding indeterminate operations are executed in real time.

3.3.2 Operation Layering

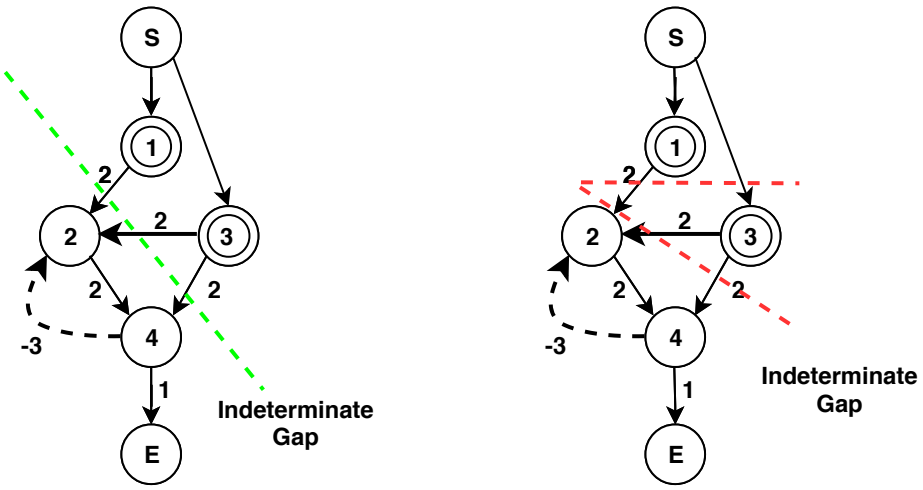


Figure 3.8: Example of layer cut, different colors represent different layering approaches.

In schedule visualization graph, a sub-schedule represents the set of operations in one layer. In the sequencing graph, the layer can be represented by a cut like the examples in fig 3.8. In fact, for real-time termination control, the start of operations in

the sequencing sub-schedule depends on the end of operations in the previous sub-schedule. The dependency results in additional edges the sequencing graph. In other words, the cut for operation layering is a division of the vertices of the sequencing network separating two parts. All operations in one part are the successors of each operation in the other part.

To avoid that the method traps solutions into local optima, it should explore all possible additional edges and the resulting operation layering.

Due to the fact that the width of the indeterminate gaps cannot be optimized, I neglect it and choose the nominal end time as one of my optimization objectives. Besides, the number of indeterminate gaps and layers are also included in my optimization objectives. Before the next sub-schedule starts, only the devices for indeterminate operation execution are active, and the rest of the devices are idle. By executing multiple indeterminate operations in the same layer, the capability to execute operations in parallel can be increased.

3.4 Storage Usage

As described in the background chapter, the storage usage is neglected in previous works. In the previous works, the products of the parent operation will be exported immediately after the end of the execution and then be imported immediately into the devices for child operations before their start. Between the end of parent operation and the start of child operations, all corresponding devices are considered to be free. This assumption simplifies the problem and can cause device usage conflict in real chip design.

In this section, several storage usages and reasonable transportation strategies to solve this problem are proposed.

3.4.1 Storage Usage Strategy

A biochip consists of several components connected by channels. It is intuitive to think of storing the operation products in a device or a channel. These have both advantages and disadvantages.

Device as storage: operation products will be placed in devices, and the devices will be marked as occupied when they are used as storage.

- Advantage:
  - Intuitive way to store the products
  - Easy to manage device usage
- Disadvantage:
  - Requires more device resources

Channel as storage: operation products will be exported into a channel after the end of the parent operation and then be stored in the channel.

- Advantage:
  - Save device resources
- Disadvantage:
  - No widely accepted standard for channel usage as storage
  - Storage capability depends on channel length

Mixed storage usage: Operation product storage depends on device usage and transportation time.

- Advantage:
  - Most efficient resource (both time and device) usage

- Disadvantage:
  - No widely accepted standard for channel usage as storage
  - Make problem more complicated

From the perspective of simplifying the problem and improving the compatibility of the method, the ***device as storage*** is chosen as my standard storage strategy.

3.4.2 Modified Schedule Visualization Graph

To describe device storage and product transportation strategy clearly, the schedule visualization graph for the representation of the device occupation is modified.

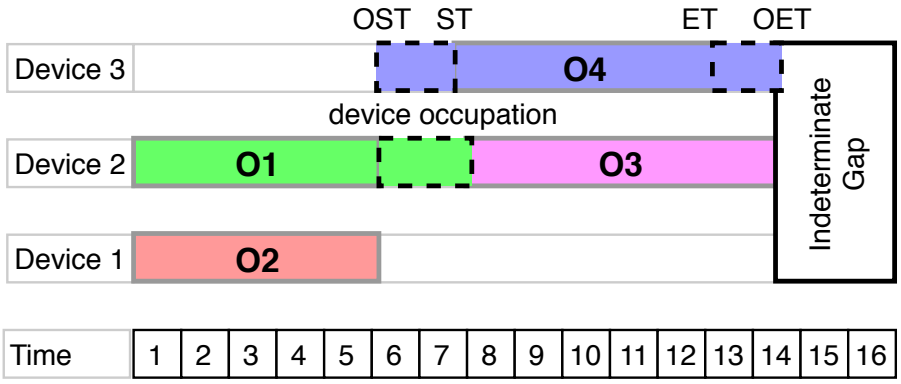


Figure 3.9: Modified schedule visualization, a device is occupied even the corresponding operation is not executed. A device can be occupied before or after the execution of the corresponding operation.

As shown in the figure 3.9, each block that represents the operation-device mapping consists now of three connected parts:

- the part (OST-ST) represents that the device is occupied by products from parent operations,

- the part (ST-ET) illustrates that the device is occupied for operation execution,
- and the part (ET-OET) serves as the device occupation for storage of operation product

The modified schedule visualization graph will be used later for my transportation strategy description and explanation of other approaches to solving the problem.

3.4.3 Transportation Strategy

The transportation strategy is to decide when and how to import and export the operation products to the designated device. In this method, I do not consider the preparation phase of an assay, and the main focus will be put on the transportation of the intermediate products.

In this section, two different transportation strategies for products transportation will be discussed.

Device for Both Parent and Child Operations: After the execution of parent operation, the operation products can be stored in both devices for the parent and the child operation, the decision depends on the device usage and other execution limitations.

Device for either Parent or Child Operation: After the execution of a parent operation, the operation products will be stored only in the device for the parent or the child operation.

- If the *device for parent operations* strategy is chosen, the device for parent operation will still be occupied after the execution until the product is completely exported to the corresponding devices due to the start of every child operations and their needs of the product of this parent operation.
- Alternatively, if the *device for child operation* strategy is chosen, the operation product will be exported immediately after the execution to every device for

child operations and these devices will be considered occupied after receiving the product.

The ***device for both parent and child operations*** strategy is chosen as my standard transportation strategy to avoid being trapped into the local optima.

3.5 Mathematical Modeling Method

I propose an integer-linear-programming (ILP) model to synthesize scheduling and binding solutions for the entire problem.

The inputs of my model include:

- **I.** a set O representing a collection of component-oriented definitions of biological operations in a biological assay. The notation IO serves as a set of indeterminate operations;
- **II.** a set D indicating the usage of general devices. The maximal number of the devices allowed to be integrated on the chip is given by the user.

The notations of the variables in my model are listed in table 3.1

3.5.1 Model Construction

Operation Layering: each operation has exactly one layer. The layer index starts from one, the start time of operations in the layer with a bigger index has to be not earlier than the end time of operations in a layer with a smaller index. In other words, the layer index of each operation is not smaller than that of its predecessor. Since the indeterminate operations should be placed at the end of the sub-schedules, the layer index of its successors should be bigger. Besides, in order to ensure that parallel operations can start at the same time, their layer index must be the same.

$$\forall o_a \in O, o_b \in o_{a.parallel\_operation}, \quad L_a = L_b \tag{1}$$

Table 3.1: 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')			$P_{d,d'}$	
Same Device (mapped to operation a and operation b)			$SD_{a,b}$	
Original Operation Dependency (between parent p and child c)			$OOD_{p,c}$	
Operation-Device-Mapping			$o\_d_{i,j}$	
Auxiliary Variable			$q_0, q_1, q_2 \ldots$	
Integer Variables				
Operation Occupation Start Time (ost)			$o_{i.ost}$	
Operation Start Time (st)			$o_{i.st}$	
Operation End Time (et)			$o_{i.et}$	
Operation occupation End Time (oet)			$o_{i.oet}$	
Layer of operation			L_i	
Transportation Time (from parent p to child c)			$T_{p,c}$	
Results Summation	total execution duration		sum_t	
	area cost	ring (r)	$sum_{a,r}$	
		chamber (ch)	$sum_{a,ch}$	
	total area cost		sum_a	
	processing cost	container (con)	$sum_{pr,con}$	
		accessory (acc)	$sum_{m,acc}$	
	total proccesing cost		sum_{pr}	
	total transportaion paths		sum_p	
Very large integer			M	

Unlike the parent/child relationship from the original operation dependency, the set of the modified parent/child relationship ($o_{i.modified\_children}/o_{i.modified\_parents}$) represents the operation dependency after the input processing and the feasibility checking procedure.

$$\forall o_a \in O, o_b \in o_{a.modified\_children} \wedge o_b \in o_{a.upper\_bound} \quad L_a = L_b \tag{2}$$

$$\forall o_a \in O \wedge o_a \in IO, o_b \in o_{a.modified\_children} \wedge o_b \in o_{a.upper\_bound} \quad L_a \leq L_b - 1 \tag{3}$$

$$\forall o_a \in O \wedge o_a \notin IO, o_b \in o_{a.modified\_children} \wedge o_b \in o_{a.upper\_bound} \quad L_a \leq L_b \tag{4}$$

Operation Dependency: a child operation can only start after receiving all its inputs from its parent operations. However, one of the inputs can be imported into the device for input storage and therefore occupies the device. In addition, if the parent and the child operations are mapped to different devices, the transportation time that delays the start time need to be considered. Furthermore, the start time of the parallel operation must be the same.

$$\forall o_i \in O, o_{i.parents} \subseteq \emptyset \quad o_{i.ost} \leq o_{i.st} \tag{5}$$

$$\forall o_a \in O, o_b \in o_{a.parallel\_operation}, \quad o_{a.st} = o_{b.st} \tag{6}$$

$$\forall o_p \in O, o_c \in o_{p.modified\_children}, \quad o_{c.ost} \leq T_{p,c} + o_{p.trans} \cdot OOD_{p,c} \cdot SD_{p,c} \tag{7}$$

$$\forall o_p \in O, o_c \in o_{p.modified\_children}, \quad o_{c.st} \geq T_{p,c} + o_{p.trans} \cdot OOD_{p,c} \cdot SD_{p,c} \tag{8}$$

Upper Bound Limitation: for the child operations with upper bound limitations, which means that the child operation must be executed within the given time after the end of the parent operation, I introduce the following constraints:

$$\forall o_p \in O, o_c \in o_{p.upper\_bound}, \quad o_{p.et} + Upper\_Bound_{p,c} \geq o_{c.st} \tag{9}$$

Similarity, the device occupation ends after the operation execution and the export of the entire operation product for the input collection of all its child operations. The end time of an operation equals its start time plus the execution duration.

$$\forall o_i \in O \wedge o_i \notin IO, \quad o_{i.et} = o_{i.st} + o_{i.duration} \tag{10}$$

$$\forall o_i \in O, o_{i.children} \subseteq \emptyset \quad o_{i.et} \leq o_{i.oet} \tag{11}$$

$$\forall o_p \in O, o_c \in o_{c.modified\_children}, \quad o_{p.et} \leq T_{p,c} \tag{12}$$

$$\forall o_p \in O, o_c \in o_{p.modified\_children}, \quad o_{p.oet} \geq T_{p,c} \tag{13}$$

Indeterminate Execution: since I place the indeterminate operations at the end of the sub-schedules, the end time of an indeterminate operation equals the maximum of the end time of all operations in the same layer of the indeterminate operation.

$$\forall o_i \in O \wedge o_i \in IO, \quad o_{i.et} \leq o_{i.st} + o_{i.duration} \tag{14}$$

$$\begin{aligned} &\forall o_a \in IO, o_b \in O \wedge o_b \notin IO, \\ &L_a \leq L_b - 1 + q_0 \cdot M \qquad (15) \\ &o_{a.et} \geq o_{b.et} - (1 - q_0) \cdot M \qquad (16) \end{aligned}$$

Device Conflict Prevention: if two operations are mapped to the same device, their occupation time has to be without overlapping, since an operation needs to monopolize exactly one device during its execution. In other words, operations with overlapping occupation time must be assigned to different devices.

$$\forall o_a, o_b \in O, d_j \in D$$
$$o_{a.ost} + (1 - q_1) \cdot M \geq o_{b.oet} \tag{17}$$
$$o_{b.ost} + (1 - q_2) \cdot M \geq o_{a.oet} \tag{18}$$
$$o\_d_{a,j} + o\_d_{b,j} - q_3 \leq 1 \tag{19}$$
$$q_1 + q_2 = q_3 \tag{20}$$

Same Device Detection: it is essential to check whether or not two operations with operation dependency are mapped to the same device since such mapping can avoid operation product transportation time, and save a physical channel between two devices. I first introduce the constraint like inequation (19) to calculate the lower bound of the boolean variable $SD_{a,b}$:

$$\forall o_a, o_b \in O, d_j \in D$$
$$o\_d_{a,j} + o\_db, j - SD_{a,b} \leq 1 \tag{21}$$

Different from the variables in inequality (19) that only need to determine the lower bound, I need to determine the upper and lower bounds of $SD_{a,b}$ at the same time. Therefore, I propose the following constraints:

$$\forall o_a, o_b \in O, d_j \in D$$
$$o\_d_{a,j} + o\_d_{b,j} + (1 - q_d) \cdot M \geq 2 \tag{22}$$
$$SD_{a,b} \leq \sum_{d \in D} q_d \tag{23}$$

Device Configuration: the following constraints are proposed by [19] to define the container and accessories as the components of the devices. As defined in the *Component-oriented Device Concept*, a *component-oriented* device consists of one container, which is either a ring or a chamber, and several accessories including pump, heating pad, optical system, sieve valve, and cell trap :

$$\forall d_j \in D, \quad d_{j,r} + d_{j,ch} = 1 \tag{24}$$

$$\forall d_j \in D, \quad d_{j,c_l} + d_{j,c_m} + d_{j,c_s} + d_{j,c_t} = 1 \tag{25}$$

$$\forall d_j \in D, \quad d_{j,c_l} + d_{j,c_m} + d_{j,c_s} = d_{j,r} \tag{26}$$

$$d_{j,c_m} + d_{j,c_s} + d_{j,c_t} = d_{j,ch} \tag{27}$$

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

$$\sum_{d_j \in D} o\_d_{i,j} = 1 \tag{28}$$

$$d_{j,x} - o\_d_{i,j} + 1 \geq o_{i,x} \tag{29}$$

$$d_{j,y} - o\_d_{i,j} + 1 \geq o_{i,y} \tag{30}$$

$$d_{j,z} - o\_d_{i,j} + 1 \geq o_{i,z} \tag{31}$$

3.5.2 Objective Configuration

The objective of my modeling method is to minimize the total assay execution time, the chip area cost, the chip processing cost, and the number of transportation paths. The proposed objective is similar to [19].

Total Assay Execution Time: it is decided by the last completed operation in the schedule, which is formulated as follows:

$$\forall o_i \in O, sum_t \geq o_{i.et} \tag{32}$$

Chip Area Cost: it is decided by the size of devices integrated on the chip. I introduce the following constraints to describe the area cost:

$$sum_{a,r} = \sum_{d_j \in D, d_{j,r}=1, x \in c_l, c_m, c_s} A_x \cdot d_{j,x} \tag{33}$$

$$sum_{a,ch} = \sum_{d_j^{'} \in D, d_{j,r}^{'}=1, y \in c_m, c_s, c_t} A_y^{'} \cdot d_{j,x}^{'} \tag{34}$$

$$sum_a = sum_{a,r} + sum_{a,ch} \tag{35}$$

Chip Processing Cost: it is decided by the cost to integrate device containers and accessories on the chip. Namely, it is so-called "manufacturing cost" of device containers and accessories.

$$sum_{pr,acc} = \sum_{d_j \in D, z \in p, h, p, s, c} Pr_z \cdot d_{j,z} \tag{36}$$

$$sum_{pr,con} = \sum_{d_j \in D, x \in r, ch} Pr_x \cdot d_{j,x} \tag{37}$$

$$sum_{pr} = sum_{pr,con} + sum_{pr,acc} \tag{38}$$

Transportation Paths: the number of paths is calculated by the following constraints:

$$\forall o_i \in O, o_j \in \{child\ operations\ of\ o_i\}, d, d^{'} \in D$$

$$o\_d_{i,d} + o\_d_{i,d'} - p_{d,d'} \leq 1 \tag{39}$$

$$sum_p = \sum_{d,d^{'} \in D} p_{d,d'} \tag{40}$$

Objective Configuration: the objective of my model is formulated as the sum of the above configurations:

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

The weight coefficient C_t ,C_a ,C_{pr} ,and C_p are defined by users.

3.6 Output Refinement

Since the constraints are carefully formulated to avoid being trapped into the local optima, the result should be the theoretically best solution of these binding and scheduling problems under the given coefficients. However, the fact, that some conditions that the schedule depends on could be changed by a physical design, can influence the scheduling and binding performance. During the physical design process, device placement and channel routing could change the length of channels and the distance between devices. Thus, the changes in transportation time of input/output samples can change the optimization result.

Although the physical design is the sequencing step of binding and scheduling for chip design automation, the solution of binding and scheduling problems can still be refined to gain fault-tolerance.

The transportation time according to potential chip layout can be re-estimated. The lengths of the transportation paths depend on the frequency of their usages. If a path p_1 is used more often than another path p_2, the length of the transportation

path p_1 should be designed shorter than the length of transportation path p_2 to reduce the transportation time of p_1 and increase the resource utilization. Using the re-estimation method, each time I get a result of the ILP model, the transportation time can be refined based on the result.

4 Result

The program is implemented in C++ and run on a computer with 2.9GHz CPU[1]. The ILP model is solved by the ILP solver Gurobi[2](version 7.5.2). Gurobi is one of the most popular ILP solver supporting multiple programming languages such as C++, Java, Python, MATLAB, and R. Besides, Python is used as a glue language to process and analyze data more efficiently and conveniently.

The assays ***ChIP***[20] and ***kinase***[29] are used to generate the test cases. Moreover, to test the performance of my method with large cases, the operations with the same protocol of the original assays are replicated. Since the time for sample storage, which will increase the total assay time, is first introduced by my work, the comparison with the solutions generated by the conventional synthesis methods is not made.

4.1 Test Cases

The details about test cases are shown in table 4.1. The original test case for [29] is a fairly easy problem since the corresponding operations are without any execution limitation. However, on the opposite, the original [20] test case contains all execution limitations and thus becomes more complex.

The number of operations includes both the number of normal operations and indeterminate operations. If an operation does not require any specified container,

[1]MacBook Pro 13-inch, 2016, macOS Mojave 10.14, 8 GB 2133 MHz LPDDR3
[2]Gurobi Optimization: http://www.gurobi.com/

Table 4.1: Test case details

	ChIP	replicated ChIP 2X	Kinase	replicated Kinase 2X	replicated Kinase 4X
# of operations	19	36	8	16	32
# of indeterminate operations	2	4	0	0	0
# of pump valves required	8	16	3	6	12
# of sieve vales required	7	14	5	10	20
# of cell traps required	0	0	0	0	0
# of heating pads required	4	8	0	0	0
# of optical systems required	0	0	1	2	4
# of total accessories required	19	38	9	18	36
# of rings required	8	16	3	6	12
# of chambers required	0	0	0	0	0
# of total accessories required	8	16	3	6	12
# of dependencies	18	36	7	14	28
# of upper bound limitations	1	2	0	0	0
# of parallel limitations	4	8	0	0	0

neither the number of rings nor the number of chambers required containers will be increased.

4.2 Performance

The performance of my method in both input processing and ILP modeling phases is recorded to analyze the efficiency of the method.

4.2.1 Input Processing

In order to show the efficiency of my method, the test cases were replicated up to 16 times. Moreover, each program was run five times to calculate their average runtime.

As shown in table 4.2, the average running time of *ChIP* test case with 19 operations is much longer than that of the *kinase2X* text case with 16 operations. The

Table 4.2: Performance of input processing

	# of operations	# of indeterminate operations	runtime (in ms)	min. duration	max. duration	avg. duration
ChIP	19	2	1084.80	60	7200	2197.89
ChIP2X	38	4	2268.40			
ChIP4X	76	8	6615.60			
ChIP8X	152	16	23997.20			
ChIP16X	304	32	90774.40			
ChIP16X duration/10	304	32	75142.00	6	720	219.79
kinase	8	0	343.60	5	30	14.38
kinase2X	16	0	573.60			
kinase4X	32	0	1531.00			
kinase8X	64	0	4354.40			
kinase16X	128	0	15013.60			
kinase16X duration/10	128	0	15111.40	0.5	3	1.44

reason may lie in that they have different numbers of indeterminate operations. The program may need more time to process indeterminate operations. Nonetheless, the relationship between the number of operations and the required runtime is approximately linear.

Moreover, the results show that the running time may depend on the execution duration of the operations. The test case *ChIP16X duration/10* contains the same operations as in the test case *ChIP16X*, but the execution duration of these operations is divided by ten. As a result, the runtime of processing *ChIP16x duration/10* is significantly reduced. However, at the same time, the runtime of *kinase16X duration/10* and *kinase16X* is of not much difference. The reason may be that the program needs more time to process large numbers.

4.2.2 Variables and Constraints generated by Model

As shown in table 4.3, an increase in the number of both operations and devices causes a boost in the number of both variables and constraints. However, since the number of operations is fixed by the assay protocols, and the number of devices is given by the user, accurate estimation of the number of devices can significantly

Table 4.3: Variables and Constraints generated by Model

	# of operations	# of indeterminate operations	# of variables	# of boolean variables	# of constraints
	# of devices = 7				
ChIP	19	2	1229	1087	5328
ChIP2X	38	4	3501	3241	14379
kinase	8	0	469	396	1799
kinase2X	16	0	938	816	4047
kinase4X	32	0	2452	2232	10463
	# of devices = 15				
ChIP	19	2	1825	1667	13080
ChIP2X	38	4	4425	4149	32563
kinase	8	0	865	776	4599
kinase2X	16	0	1462	1324	9951
kinase4X	32	0	3232	2996	24111

reduce the number of constraints and therefore improve the efficiency of the method.

4.2.3 Modeling Result

Some representative test cases are executed twice with different device numbers (such as 7 and 15, respectively) to show the different runtime and solutions. However, for example, if a test case needs only one device to get its optimum, the replicated test case only needs two devices theoretically. After the solver gets its optimum, the path transportation time will be re-estimated.

kinase: As shown in the table 4.4, this test case is executed four times. The total execution time of the assay remains at 115, while the runtime of the program fluctuates between 0.08s and 0.15s. All operations are assigned into two layers by the model. However, there is actually only one valid layer, since this test case contains a series of operations, which connect each other in a chain, and a layer can be inserted between any dependency edge according to the definition of the layer. The fluctuation in runtime may be caused by the changes in the operating efficiency of the CPU. Because the case where the container of the device is a ring or a chamber is mutually

Table 4.4: kinase case result

		round 1	round 2		round 1	round 2
runtime in second		0.15	0.10		0.08	0.14
execution time		115	115		115	115
# of operations	D7	8	8	D15	8	8
# of indeterminate operations		0	0		0	0
# of devices		7	1		15	1
# of layers		2*	2*		2*	2*
	ring/chamber	pump valve	sieve valve	cell trap	heating pad	optical system
device details round 1						
device 1	0	1	1	0	0	1
device details round 2						
device 1	0	1	1	0	0	1

exclusive, the result indicating that the container type of the device is integrated into a cell, where 0 means that the type of the container is a ring, and 1 means chamber.

kinase2X: As expected, the number of devices shown in table 4.5 is two in this case. The total execution time of the assay is slightly reduced in the second round by adding an extra device. However, whether or not to use additional devices depends on the coefficients of my target function. The current coefficients are chosen at random.

kinase4X: As shown in table 4.6, the runtime explodes while the increase of the operation numbers stays linear. This result shows that it is still a challenge to solve binding and scheduling problems with a large number of operations using my method. However, the total execution time remains 180s, which means all sub-assays are performed in parallel. Furthermore, this indirectly proves that my solution to the original test case is optimal.

ChIP: This test case contains 19 operations, two of which are indeterminate operations. Compared with the *kinase2X* test case containing 16 operations, the total runtime required by *ChIP* is much longer than *kinase2X*.

ChIP2X: As shown in table 4.8, the runtime is 2162.19 seconds. Compared to

Table 4.5: kinase2X case result

	round 1			round 2		
runtime in second	8.34			2.16		
execution time	180			174		
# of operations	16			16		
# of indeterminate operations	0			0		
# of devices	2			3		
# of layers	2*			2*		
device details round 1						
	ring/chamber	pump valve	sieve valve	cell trap	heating pad	optical system
device 1	0	1	1	0	0	1
device 2	1	1	1	0	0	1
device details round 2						
device 1	0	1	1	0	0	1
device 2	1	1	1	0	0	1
device 3	1	1	1	0	0	1

Table 4.6: kinase4X case result

runtime in second	960.04					
execution time	180					
# of operations	32					
# of indeterminate operations	0					
# of devices	4					
# of layers	2*					
device details						
	ring/chamber	pump valve	sieve valve	cell trap	heating pad	optical system
device 1	0	1	1	0	0	1
device 2	1	1	1	0	0	1
device 3	0	1	1	0	0	1
device 4	1	1	1	0	0	1

Table 4.7: ChIP case result

runtime in second	160.22					
execution time	12460					
# of operations	19					
# of indeterminate operations	2					
# of devices	4					
# of layers	4*					
device details						
	ring/chamber	pump valve	sieve valve	cell trap	heating pad	optical system
device 1	0	1	1	0	1	1
device 2	0	1	1	0	1	1
device 3	0	1	1	0	1	1
device 4	0	1	1	0	1	1

the test case *kinase4X* which containing 32 operations, this test case contains the similar number of operations which is 38 but requires 125% more time than that of the test case *kinase4X*. The possible reasons might be the additional operations and the increased complexity to process indeterminate operations related constraints. However, these results prove that the input processing is meaningful for saving the required time to detect the feasibility of inputs using the ILP model directly.

ChIP4X: As an extra test case, the *ChIP4X* test case is also performed. However, the program terminates when the given time limit, which is defined as 60000 seconds, is exceeded.

4.2.4 Heuristic

Since the transportation time can be modified after the optima are solved by re-estimation, if two operations are mapped to a device, the corresponding transportation time can be set to 0 and the constraints to check whether the operations are mapped to the same device can be neglected.

As shown in table 4.9, the runtime is reduced significantly while the total

Table 4.8: ChIP2X case result

runtime in second	2162.19					
execution time	12460					
# of operations	38					
# of indeterminate operations	4					
# of devices	8					
# of layers	3					
device details						
	ring/chamber	pump valve	sieve valve	cell trap	heating pad	optical system
device 1	0	1	1	0	1	1
device 2	0	1	1	0	1	1
device 3	0	1	1	0	1	1
device 4	0	1	1	0	1	1
device 5	0	1	1	0	1	1
device 6	0	1	1	0	1	1
device 7	0	1	1	0	1	1
device 8	0	1	1	0	1	1

execution time is increased slightly.

Table 4.9: Heuristic result

	original method		heuristic	
	runtime	execution time	runtime	execution time
kinase	0.15	115	0.04	150
kinase2X	8.34	180	0.55	195
ChIP	160	12460	1.26	12760

5 Conclusion

In this work, I have proposed a method to solve binding and scheduling problems with execution limitations and upper bound limitation using a component-oriented device and operation concept. My approach tries to formulate an integer-linear-programming model to solve these problems comprehensively without being trapped into local optima. Besides, an algorithm to check the feasibility of the inputs is also proposed to improve the robustness of my method.

The experimental results show that my approach is feasible with all execution limitations considering storage usage and on-chip-resource-related costs. However, the runtime of my program exploded when the number of operations is very large. I am confronting with the difficulty that it is still a challenge to make the method scalable. The bottleneck of my method is to detect whether the operation is assigned to the same device. The number and complexity of the constraints have increased significantly in this step. In the future, heuristics might be applied to reduce the complexity of these problems.

Bibliography

[1] P. Wang, L. Robert, J. Pelletier, W. L. Dang, F. Taddei, A. Wright, and S. Jun. "Robust growth of Escherichia coli". In: *Current biology* 20.12 (2010), pp. 1099–1103.

[2] V. Chokkalingam, J. Tel, F. Wimmers, X. Liu, S. Semenov, J. Thiele, C. G. Figdor, and W. T. Huck. "Probing cellular heterogeneity in cytokine-secreting immune cells using droplet-based microfluidics". In: *Lab on a chip* 13.24 (2013), pp. 4740–4744.

[3] J. Liu, C. Hansen, and S. R. Quake. "Solving the "world-to-chip" interface problem with a microfluidic matrix". In: *Analytical chemistry* 75.18 (2003), pp. 4718–4723.

[4] C. L. Hansen, S. Classen, J. M. Berger, and S. R. Quake. "A microfluidic device for kinetic optimization of protein crystallization and in situ structure determination". In: *Journal of the American Chemical Society* 128.10 (2006), pp. 3142–3143.

[5] J. Liu, B. A. Williams, R. M. Gwirtz, B. J. Wold, and S. Quake. "Enhanced signals and fast nucleic acid hybridization by microfluidic chaotic mixing". In: *Angewandte Chemie International Edition* 45.22 (2006), pp. 3618–3623.

[6] S. Haeberle and R. Zengerle. "Microfluidic platforms for lab-on-a-chip applications". In: *Lab on a Chip* 7.9 (2007), pp. 1094–1110.

[7] J. Melin and S. R. Quake. "Microfluidic large-scale integration: the evolution of design rules for biological automation". In: *Annu. Rev. Biophys. Biomol. Struct.* 36 (2007), pp. 213–231.

[8] K. S. Elvira, X. C. i Solvas, R. C. Wootton, et al. "The past, present and potential for microfluidic reactor technology in chemical synthesis". In: *Nature chemistry* 5.11 (2013), p. 905.

[9] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann. "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.8 (2018), pp. 1588–1601.

[10] D. Roberge. "Lonza–hazardous flow chemistry for streamlined large scale synthesis". In: *Green Processing and Synthesis* 1.1 (2012), pp. 129–130.

[11] D. Mark, S. Haeberle, G. Roth, F. Von Stetten, and R. Zengerle. "Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications". In: *Microfluidics Based Microsystems*. Springer, 2010, pp. 305–376.

[12] G. T. Kovacs et al. "Micromachined transducers sourcebook". In: (1998).

[13] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake. "Monolithic microfabricated valves and pumps by multilayer soft lithography". In: *Science* 288.5463 (2000), pp. 113–116.

[14] Y. Xia and G. M. Whitesides. "Replica molding with a polysiloxane mold provides this patterned microstructure". In: *Angew. Chem. Int. Ed* 37 (1998), pp. 550–575.

[15] I. E. Araci and S. R. Quake. "Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves". In: *Lab on a Chip* 12.16 (2012), pp. 2803–2806.

[16] J. S. Marcus, W. F. Anderson, and S. R. Quake. "Microfluidic single-cell mRNA isolation and analysis". In: *Analytical chemistry* 78.9 (2006), pp. 3084–3089.

[17] 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: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*. IEEE. 2016, pp. 624–629.

[18] H.-P. Chou, M. A. Unger, and S. R. Quake. "A microfabricated rotary pump". In: *Biomedical Microdevices* 3.4 (2001), pp. 323–330.

[19] M. Li, T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann. "Component-oriented high-level synthesis for continuous-flow microfluidics considering hybrid-scheduling". In: *Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEE*. IEEE. 2017, pp. 1–6.

[20] 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". In: *Lab on a Chip* 8.1 (2008), pp. 68–74.

[21] J. Liu, M. Enzelberger, and S. Quake. "A nanoliter rotary device for polymerase chain reaction". In: *Electrophoresis* 23.10 (2002), pp. 1531–1536.

[22] D. Di Carlo, N. Aghdam, and L. P. Lee. "Single-cell enzyme concentrations, kinetics, and inhibition analysis using high-density hydrodynamic cell isolation arrays". In: *Analytical chemistry* 78.14 (2006), pp. 4925–4930.

[23] Y. Marcy, T. Ishoey, R. S. Lasken, T. B. Stockwell, B. P. Walenz, A. L. Halpern, K. Y. Beeson, S. M. Goldberg, and S. R. Quake. "Nanoliter reactors improve multiple displacement amplification of genomes from single cells". In: *PLoS genetics* 3.9 (2007), e155.

[24] 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". In: *Lab on a Chip* 9.10 (2009), pp. 1365–1370.

[25] O. Flores, Ö. Deniz, M. Soler-Lopez, and M. Orozco. "Fuzziness and noise in nucleosomal architecture". In: *Nucleic acids research* 42.8 (2014), pp. 4934–4946.

[26] A. R. Wu, T. L. Kawahara, N. A. Rapicavoli, J. Van Riggelen, E. H. Shroff, L. Xu, D. W. Felsher, H. Y. Chang, and S. R. Quake. "High throughput automated chromatin immunoprecipitation as a platform for drug screening and antibody validation". In: *Lab on a Chip* 12.12 (2012), pp. 2190–2198.

[27] D. C. Ku and G. De Mitcheli. "Relative scheduling under timing constraints: Algorithms for high-level synthesis of digital circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11.6 (1992), pp. 696–718.

[28] J. E. Kelley Jr. "Critical-path planning and scheduling: Mathematical basis". In: *Operations research* 9.3 (1961), pp. 296–320.

[29] 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, et al. "Integrated microfluidic and imaging platform for a kinase activity radioassay to analyze minute patient cancer samples". In: *Cancer research* (2010), pp. 0008–5472.