Storage-Aware Algorithms for Dilution and Mixture Preparation with Flow-Based Lab-on-Chip

Sukanta Bhattacharjee, *Member, IEEE*, Robert Wille, *Senior Member, IEEE*,
Juinn-Dar Huang, *Member, IEEE*, and Bhargab B. Bhattacharya, *Fellow, IEEE*

***Abstract*—Lab-on-Chip (LoC) technology has emerged as one of the major driving forces behind the recent surge in biochemical protocol automation. Dilution and mixture preparation with fluids in a desired ratio, constitute basic steps in sample preparation for which several LoC based architectures and algorithms are known. The optimization of cost and time for such protocols requires proper sequencing of fluidic mix-and-split steps, and storage-units for holding intermediate-fluids to be reused in later steps. However, practical design constraints often limit the amount of on-chip storage in microfluidic LoC architectures and thus can badly affect the performance of the algorithms. Consequently, results generated by previous work may not be useful (in the case they require more storage-units than available) or more expensive than necessary (in the case when storage-units are available but not used, e.g., to further reduce the number of mix/split operations or reactant-cost). In this paper, we propose new algorithms for dilution and mixing with continuous-flow based LoCs that explicitly take care of storage constraints while optimizing reactant-cost and time of sample preparation. We present a symbolic formulation of the problem that captures the degree of freedom in algorithmic steps satisfying the specified storage constraints. Solvers based on Boolean satisfiability are used to achieve the optimization goals. Experimental results show the efficiency and effectiveness of the solution as well as a variety of applications where the proposed methods would prove beneficial.**

I. Introduction

Labs-on-a-Chip (LoC, [2]) are integrated devices which exploit recent advances in microfluidics to provide an alternative to conventional and bulky lab equipment for biomedical experiments. In fact, they realize standard lab operations such as mixing and splitting of fluids or even heating and observing in an automatic and low-cost fashion which offers much higher throughput. LoCs have already been applied in areas such as point-of-care diagnosis [3], sample preparation [4–8], DNA analysis [9], and drug discovery [10].

Several platforms have been proposed such as digital microfluidic biochips (DMFBs) or continuous-flow microfluidic biochips (CFMBs). Traditionally, the latter class has enjoyed wide acceptance in the engineering community as they mimic chemical protocols more realistically [3, 9, 11]. Modern CFMBs are usually equipped with pressure-driven micro-valves that allow for controlling the fluid flow through a network of micro-channels [12–14]. More specifically, a CFMB normally consists of two layers of permanently etched micro-channels called the flow and control layer as shown in Fig. 1(a). External pressure sources are applied to the control layer to deflect the flexible membrane (placed at the intersection between the two layers) deep into the flow layer Fig. 1(b). This creates a pressure-driven micro-valve that allows for controlling the fluid flow. Based on this, more complex units such as mixers, micro-pumps, multiplexers, and storage-units can be built by suitably organizing micro-valves [11–13].

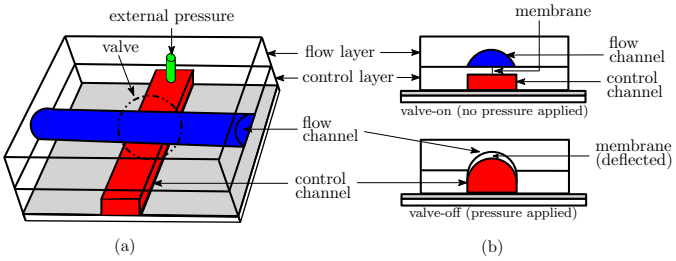


Fig. 1: Schematic of a two-layer microfluidic device: (a) top-view and (b) cross-sectional view of valve-states [13].

Flow-based LoCs such as CFMBs have been considered as one of the key technologies for automatic sample preparation, i.e., the generation of dilutions and mixtures of fluids in certain ratios [6–8, 15, 16]. The corresponding flow-based microfluidic architectures are usually equipped with one or more on-chip mixers and few storage-units. The mixers implement a mixing model, e.g., a rotary mixer, commonly used in CFMB (a detailed description of various kinds of rotary mixers and their supporting mixing models is provided later in Section II).

A "storage-module" is needed when an intermediate fluid-mixture needs to be stored for subsequent use. We consider a biochip architecture as shown in Fig. 2 for sample preparation. It comprises one storage module that has a number of storage-units arranged as parallel channels, each of which can store unit-volume fluids, i.e., the amount contained in one segment of the rotary mixer. The transportation of a fluid to (from) the module is controlled by de-multiplexer (multiplexer), and they can be interchangeably configured. Access to (from) this module requires an additional carrier

The work of B. B. Bhattacharya was supported, in part, from the grant provided by INAE Chair Professorship, and from a special PPEC-funded grant to Nanotechnology Research Triangle provided by Indian Statistical Institute, Kolkata. A preliminary version of this paper has appeared in the proceedings of DATE 2018 [1].

S. Bhattacharjee is with Center for Cyber Security, New York University, Abu Dhabi 129188, UAE, E-mail: sb6538@nyu.edu

R. Wille is with Institute for Integrated Circuits, Johannes Kepler University Linz, 4040 Linz, Austria, E-mail: robert.wille@jku.at

J.-D. Huang is with Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan. E-mail: jdhuang@mail.nctu.edu.tw

B. B. Bhattacharya is with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. Email: bhargab.bhatta@gmail.com

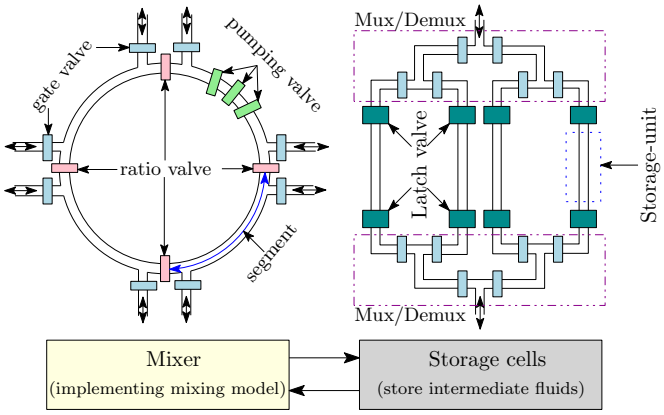


Fig. 2: Microfluidic architecture for sample preparation.

fluid to transport the payload between an on-chip mixer segment and the storage-unit [15]. During this operation, the carrier fluid enters through one port of a mixer-segment and exits through the other port of the mixer – pushing out the payload. Before transporting a payload to a segment of the storage module (or from the module to the mixer), valves in the interconnection network need to be configured to create a close-loop connection between the rotary mixer and the storage module. In this paper, we assume that such transportation mechanism is supported by the underlying biochip architecture.

Instead of using a dedicated storage module, free channels can also store intermediate fluids. The method proposed in [17] realizes a sequencing graph by allocating free channels for intermediate storage during the design process. While this method [17] utilizes free channels for storage, it will need an accurate 'metering' mechanism for selecting unit-volume fluid while loading and storing it from the channels themselves. Furthermore, the use of free channel as storage may lead to inaccuracies which affect the target-concentration and, hence, is crucial particularly for sample preparation [18]. A dedicated storage module equipped with access control can handle this volume-control problem more conveniently at the cost of some additional resources. Because of this and in order to ease the following descriptions, we assume a microfluidic platform with a dedicated storage module as shown in Fig. 2. However, the proposed approach can also be adapted to channel storage as used in [17].

Several approaches have been proposed for sample preparation on LoCs [4–8, 16, 19] considering different optimization objectives such as minimization of the (1) number of mixing operations, (2) consumption of valuable reagents, and (3) amount of waste generation. However, most of the existing LoC sample preparation methods only consider these three optimization objectives, but ignore that, on most chips, the number of available storage-units is additionally limited. This poses a problem since results generated by previous work may not be useful (in the case they require more storage-units than available) or more expensive than necessary (in the case when storage-units are available but not used, e.g., to further reduce the number of mixing operations or reactant-cost). Besides that, many biochemical assays often demand multiple samples to be prepared in parallel – again, requiring a storage-aware sample preparation.

Moreover, the explicit consideration of storage-units may also help to improve on the other objectives. For example, since existing methods for sample preparation do not consider on-chip storage restrictions, a designer has to deploy a maximum number of storage-units (corresponding to the worst case storage requirement of the underlying method) for each sample preparation module – significantly increasing the costs. Vice versa, in cases where sample preparation is possible without any intermediate storage-units, possible improvements by the fact that these units are on the chip anyway are not utilized. Hence, either way, sample preparation methods are required which satisfy restrictions on on-chip storage-units or exploit their availability even if they are not necessarily required in order to reduce the costs.

In this paper, we address these issues by proposing a storage-aware sample preparation method for mixing two or more biochemical reagents on a CFMB. A SAT-based approach is proposed which allows to check several options of generating the desired target ratio and choosing the one which makes the best usage of the available storage-units while, optimizing sample preparation costs and/or time. The proposed method also guarantees that no solution is chosen which requires more storage-units than available for the given platform, and flags when such a solution does not exist. Moreover the proposed algorithm can be generalized for applications to other microfluidic platforms such as micro-electrode-dot-array (MEDA) [20], FPVA [21] and PMD [22], which support even stronger mixing models.

The organization of the rest of this paper is as follows. Section II describes the preliminaries of sample preparation, reviews existing sample preparation algorithms, and motivates this work. Afterwards, an overview of the proposed methods is presented in Section III. Based on this, detailed descriptions for dilution and mixing are given in Section IV and Section V, respectively. Experimental results are reported in Section VI. Finally, the paper is concluded in Section VII.

II. Background and Motivation

This section briefly reviews commonly used fluid mixers on microfluidic platforms. Moreover, it details the task of sample preparation using CFMBs as well as the available methods proposed for this purpose. Afterwards, we are discussing how the restriction on the number of storage-units affects the sample preparation and requires alternative solutions for this crucial task. Overall, this provides the necessary background and motivation for the remainder of this work.

A. Backgrounds

1) Microfluidic mixers and mixing models: Rotary mixers are commonly used in CFMBs for mixing two or more fluids [6–8, 16]. More precisely, CFMBs allow to realize multiple mixing-ratios using a Mixer-N which is divided into N equally large segments as shown in Fig. 3(a). The term "segment" is used to denote a unit-section of Mixer-N, and

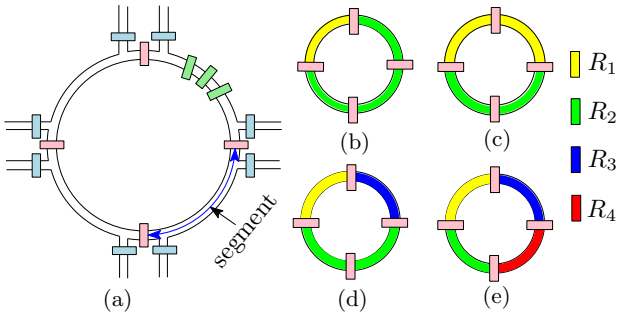


Fig. 3: (a) A 4-segment rotary mixer (Mixer-4), and possible mixing models (b) (1:3) (c) (1:1) (d) (1:2:1) (e) (1:1:1:1).

the corresponding volume of the channel is defined as "unit-volume"; thus a mixer-segment can hold unit-volume fluid. The ratio valves allow to fill each segment with a different fluid as illustrated in Figs. 3(b)-(e). Afterwards, those fluids can be mixed by actuating the pumping valves in a peristaltic sequence at a high frequency. Eventually, this allows to employ various mixing models e.g. (1:3), (1:1), (1:2:1), or (1:1:1:1) if Mixer-4 is deployed.

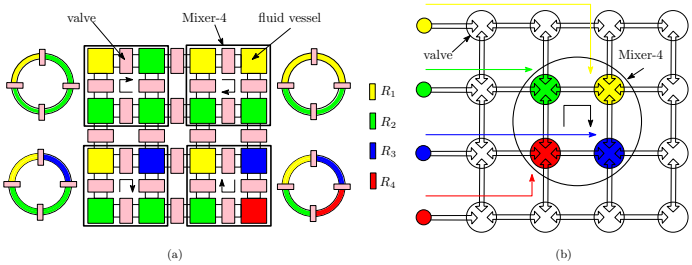


Fig. 4: Schematic of (a) fully programmable valve array [21] and (b) programmable microfluidic device [22].

The same mixing models can be realized by recently proposed flow-based LoC platforms such as fully programmable valve arrays (FPVA [21]) or Programmable Microfluidic Device (PMD [22]). They are flexible and highly re-configurable and, by this, allow to realize the rotary mixers, storage cells, and transportation channels by properly configuring valve states (using methods such as [23–26]). For example, Fig. 4(a) and Fig. 4(b) show a 4×4 FPVA platform and a 4×4 PMD, in which valves and fluid vessels are arranged in a regular fashion. Fluid vessels are connected using horizontal and vertical flow channels. As seen in Fig. 4(a), this allows four mixers (Mixer-4) implementing various mixing models corresponding to the rotary mixers shown in Figs. 3(b)-(e).

2) Sample Preparation: Sample preparation is the process of mixing two or more biochemical fluids in a given volumetric ratio through a sequence of mixing operations[1]. Fig. 5 depicts the main steps involved in sample preparation. Given a target ratio of m input fluids $\mathcal{M} = \{R_1 : R_2 : \cdots : R_m = x_1 : x_2 : \cdots : x_m\}$, where $0 \leq x_i \leq 1$ and $\sum_{i=1}^{m} x_i = 1$, we represent the desired ratio with respect to a mixing model supported by the microfluidic platform (here: CFMB) and a user-defined tolerance $0 \leq \epsilon < 1$. On a CFMB platform that supports multiple mixing model such as a Mixer-N, and given a user-defined tolerance ϵ, the target ratio $\mathcal{M}$ is transformed to a reachable mixing ratio $\{R_1 : R_2 : \cdots : R_m = y_1 : y_2 : \cdots : y_m\}$, where $0 \leq y_i \leq N^d$ and $\sum_{i=1}^{m} y_i = N^d, d \in \mathbb{N}$. Note that d is selected depending on the user-defined error tolerance limit $0 \leq \epsilon < 1$ satisfying $\max_i\{|x_i - \frac{y_i}{N^d}|\} < \epsilon$. The following example describes the ratio-transformation procedure.

[1]Note that the term *dilution* is used when two fluids (usually sample and buffer) are mixed; otherwise the general term *mixing* is common.

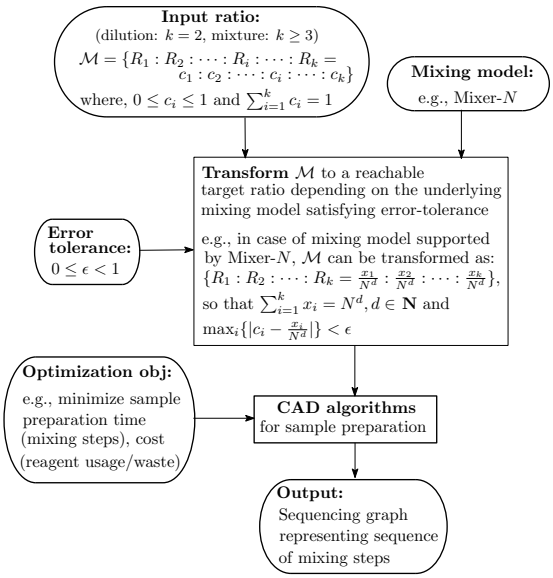


Fig. 5: Overview of sample preparation.

Example 1. *Consider a target ratio of two input fluids* $\{$*sample : buffer = 0.489 : 0.511*$\}$ *(note that,* $0.489 + 0.511 = 1$*). The target ratio can be represented as* $\{$*sample : buffer = 125 : 131*$\}$ *(*$125 + 131 = 4^4$*) for Mixer-4 (i.e.,* $N = 4$*) and for* $\epsilon = 0.001$*. Note that* $\max\{|0.489 - \frac{125}{4^4}|, |0.511 - \frac{131}{4^4}|\} = 0.0007 < \epsilon$*, i.e., d is chosen to be four. Note that,* $d = 4$ *is the smallest number which satisfies the given error-tolerance.*

For the transformed ratio $\{R_1 : R_2 : \cdots : R_m = y_1 : y_2 : \cdots : y_m\}$, the sample-preparation algorithm genMixing [8] represents each y_i as d-digit N-ary number i.e., $(y_i)_{10} = (a^i_{d-1}a^i_{d-2}\cdots a^i_1 a^i_0)_N$. Next, these m d-digits are scanned from right-to-left to construct a mixing tree in a bottom-up fashion. The depth of the mixing tree is determined by d. The mixing tree is used to represent the sequence of mixing operations needed in order to achieve the target ratio. Corresponding to each non-zero digit a^i_j in the N-ary representation of y_i, a^i_j units of input reagent R_i are fed as input to the mixer (shown as leaf nodes of the mixing tree).

The dilution algorithm NWayMix is a special case of genMixing for mixing only two input reagents i.e., $m = 2$, it generates the mixing tree with minimum number of mixing steps. Fig. 6 shows the mixing tree for the target ratio $\{$sample : buffer $= 125 : 131\}$ generated by the NWayMix.

3) State of the Art: In the recent past, Liu *et al.* proposed a tree pruning and grafting method (called *TPG* [7]) that starts from an initial mixing tree (based on a (1:1) mixing model) and transforms it for obtaining a dilution graph for

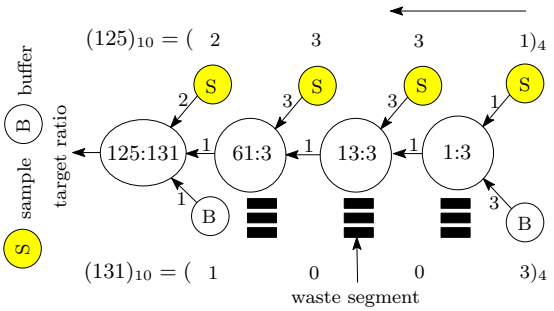


Fig. 6: Dilution graph generated with NWayMix.

unequally segmented rotary mixer (Ring-N). In [6], a volume-oriented sample preparation algorithm (called *VOSPA*) has been introduced which employs a greedy strategy. Lei *et al.* [27] proposed a network-flow based multi-objective dilution method that utilizes the full flexibility of the multiple mixing model offered by Mixer-N. Later, a flow-based sample preparation algorithm (called *FloSPA*) was proposed in [8] that can handle dilution and mixing within one framework and fully utilize the power of the multiple mixing model supported by the Mixer-N. A summary of these existing CFMB-based sample-preparation methods is provided in Table I.

TABLE I: Summary of CFMB sample preparation algorithms.

Method	#-Input reagents	Use all possible mixing ratios of underlying mixing model?	Considers number of storage-units?
NWayMix [8]	2	No	No*
TPG [7]	2	No	No◇
VOSPA [6]	2	No	No◇
Flow-based [27]	2	Yes†	No◇
FloSPA [8]	$\geqslant 2$	Yes	No◇
Proposed	$\geqslant 2$	Yes	Yes

* Does not utilize any storage-unit at all (and, hence, yields rather expensive solutions when storage-units are available).
◇ Provides an invalid solution when the number of storage-units is insufficient compared to what is necessitated by the algorithm.
† Is computationally expensive when the number of segments in Mixer-N increases.

B. Motivation: Storage-aware Sample Preparation

All previously proposed methods for sample preparation using CFMB do not explicitly take the number of available storage-units into account (as reviewed in Table I). This leads to severe problems and drawbacks as illustrated by the following example.

Example 2. *Suppose we need to prepare a mixing ratio* $\{$*sample : buffer = 125 : 131*$\}$ *on a CFMB platform that supports only two on-chip storage-units. The mixing graph determined with existing sample preparation methods, e.g., VOSPA [6] and FloSPA [8], require four and five storage-units as shown in Fig. 7(a) and Fig. 7(b), respectively. Hence, these results obtained by these approaches are useless. Moreover, since a dilution problem is considered here, a mixing graph requiring zero storage-unit as shown in Fig. 7(c) can be determined using the NWayMix [8] approach. But since this does not utilize the available storage-units, a total of 9 units of the sample are required in this case (cf. Fig. 7(c)) – a very expensive solution. In contrast, the desired mixing ratio can be realized more efficiently as shown in Fig. 7(d). Not only the improved solution requires no more than the available number of storage-units (hence, it is a valid solution) but also exploits them to reduce the total number of sample-units from 9 to 4.*

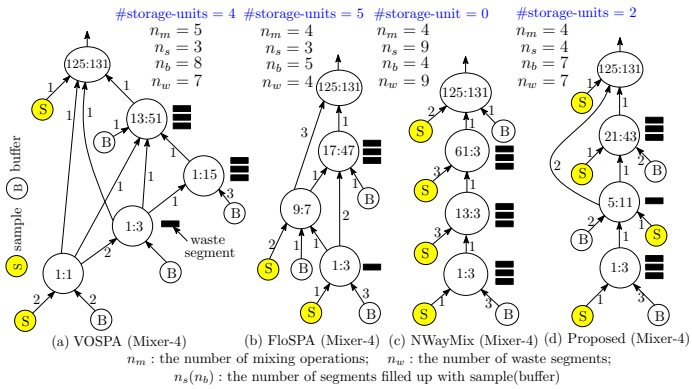


Fig. 7: Dilution graph generated with (a) NWayMix, (b) VOSPA, (c) FloSPA, and (d) the proposed method for $\{$sample : buffer = 125 : 131$\}$.

The above-mentioned example motivates us to develop a storage-aware sample-preparation method, which does not generate a mixing graph exceeding the number of available storage-units and, at the same time, fully exploits them in order to reduce the costs. In this work, we propose such a method.

III. Overview on the Proposed Storage-Aware Methods

In this section, an overview of the proposed method is described briefly. The main idea is to utilize mixing graphs generated by earlier approaches as basis, which already provide an option how to eventually realize the desired concentration ratio [6, 8]. Next, the mixing tree is augmented with additional nodes (allowing to use further input reagents) and edges (allowing to share intermediate fluids) – eventually providing several further options for realizing the desired input ratio. However, in order to determine the one which gives the minimum reagent usage and, at the same time satisfies the limitations in storage-units is a computationally complex task. In order to cope with this complexity, we use the computational power of Boolean satisfiabiliy solvers [28–30], which already have been found effective for similar tasks in the design of LoCs (see e.g. [31–33]). The main idea is to symbolically represent all possible options (given by the augmented mixing graph) and to extend this representation by constraints enforcing the storage limitation. Finally, the resulting formulation is passed to a solving engine which either determines a satisfying solution (out of which a mixing graph satisfying the storage constraints can be derived) or proves that, considering the available options, no such solution exists.

In the following, the proposed approach is described in two steps. First, in Section IV, we consider dilution problems only, i.e., the case where only two fluids (a sample and a buffer) are mixed. For this case, we start with a mixing tree generated

by NWayMix [8], i.e., the resulting mixing tree will have the minimum number of mixing steps, when all inputs to a Mixer-N are provided as either pure sample (100%) or neutral buffer (0%). Under such conditions, it requires no on-chip storage for saving intermediate fluids for later use, and hence, it constitutes an optimal starting point. Here, in fact, every ratio can be realized without using any additional storage-unit for saving intermediate fluid (although when storage-units are available, they can possibly be utilized to further reduce the cost of sample preparation). Afterwards, the general case of mixing is covered in Section V, i.e., the case where more than two fluids are mixed. We start here with the initial mixing tree generated by genMixing (the only method known for Mixer-N that builds a mixing tree from the scratch), and optimize reagent-usage subject to the constraints on on-chip storage-units. We also assume that only one Mixer-N is available on-chip. Note that while a zero-storage solution always exists for dilution, implementing a given mixing ratio may not always be possible with a given number of storage-units under the two assumptions stated earlier regarding the choice of initial tree and availability of on-chip mixers. The proposed formulation allows for exploiting further possibilities beyond the choice of a particular initial tree, though we have not considered in this work.

IV. Proposed Method for Storage-Aware Dilution

We describe the proposed method for dilution. Given a desired ratio of sample and buffer, the method starts with a mixing graph generated with an existing sample preparation method. Here, the approach called NWayMix and proposed in [8] is suitable due to two main reasons[2]: First, NWayMix generates a target ratio using Mixer-N with a minimum number of mixing steps i.e. a minimum sample preparation time. Second, the mixing tree generated by NWayMix resembles a chain (i.e., a skewed graph) and, hence, can be executed on a single CFMB-mixer without any on-chip storage-unit for intermediate fluids. Fig. 8(a) sketches the resulting graph.

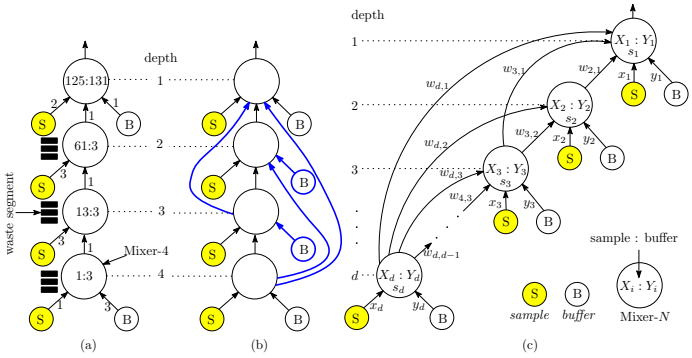


Fig. 8: (a) Dilution tree generated by NWayMix for target ratio {sample : buffer = 125 : 131} (b) dilution graph after adding extra edges and reagent nodes for enabling reagent minimization, (c) general structure of dilution graph.

[2]Nevertheless, the proposed method can similarly be applied using other sample-preparation methods.

The main idea is to augment the mixing graph produced by NWayMix with additional leaf-nodes (input reagents) and edges – allowing for further options to realize the desired ratio. This is sketched by means of blue leaf-nodes and edges in the graph shown in Fig. 8(b). The general structure of such transformation is shown in Fig. 8(c). They eventually represent further options for mixing in which intermediate results (stored in storage-units) are re-used. This yields the question what inputs shall be used in each mixing step (i.e., what edges shall remain in the mixing graph). In order to determine the best possible result, all possibilities should be checked for this purpose. Since doing this enumerately is infeasible, we formulate this problem in terms of a satisfiability instance and the resulting formulation is passed to a solving engine. The satisfiabilty solver assigns variables satisfying storage constraints and desired dilution graph can be produced from the assignment. Fig. 9 summarizes the overall flow of the proposed dilution algorithm.

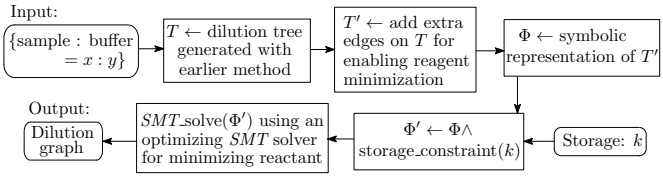


Fig. 9: Flowchart of the proposed algorithm for dilution.

A. Symbolic Formulation

In this subsection, we describe our proposed modeling in detail. We introduce the following free variables for symbolic modeling of general dilution graph.

Node variables: For each mixing node at depth i in the mixing graph, we define two rational variables X_i and Y_i ($1 \leq i \leq d$) that denote the ratio between sample and buffer of the resulting mixing operation at depth i.

Reagent variables: The input reagents (sample and buffer) can be used in any mixing node at depth i, where $1 \leq i \leq d$. For denoting the number of segments filled with sample and buffer in a Mixer-N at depth i, two integer variables x_i and y_i are associated, respectively.

Segment sharing variables: The integer variables $w_{i,j}$ represent the number of segments that are used in Mixer-N at depth j from Mixer-N at depth i, where $1 \leq j < i \leq d$.

Storage variables: An integer variable s_i is associated with each mixing node at depth i ($1 \leq i \leq d$) for denoting the number of on-chip storage-units required for executing the portion of induced subgraph containing mixing nodes at depth j, where $i \leq j \leq d$. Note that s_1 denotes the storage requirement for the entire mixing graph.

The annotations in Fig. 8(c) provide all variables used in this case. However, passing this representation to a solving engine would yield an arbitrary assignment to all variables and, hence, a mixing tree that realizes arbitrary ratios in each depth using an arbitrary number of storage-units – obviously a solution which is neither valid nor desired. Hence, we need to further constrain the introduced variables so that indeed the desired result is determined.

Enforcing the Desired Ratio

First, the correctness of the respective mixing ratios is enforced. Note that in the mixing step represented by a node at depth i, the mixer can fill its segments with sample, buffer, or any unused fluids produced at depth $j > i$. The desired ratio of sample and buffer at depth i i.e., $\{X_i : Y_i\}$ can be determined with the following equations.

$$x_i + w_{d,i}X_d + w_{d-1,i}X_{d-1} + \cdots + w_{i+1,i}X_{i+1} = NX_i \quad (1)$$

$$y_i + w_{d,i}Y_d + w_{d-1,i}Y_{d-1} + \cdots + w_{i+1,i}Y_{i+1} = NY_i \quad (2)$$

The above two equations computes the contribution of sample (buffer), i.e., X_i (Y_i), in the ratio generated at depth i using the contribution of sample (buffer) used at depth i, i.e., $\frac{x_i}{N}$ ($\frac{y_i}{N}$) and the contributions of sample (buffer) in the ratio produced at depth $j = i+1, i_2, \cdots, d$, i.e., $\frac{1}{N}\sum_{j=i+1}^{d} w_{j,i}X_j$ ($\frac{1}{N}\sum_{j=i+1}^{d} w_{j,i}Y_j$). Furthermore, we can remove the non-linearity of above equations by adding few extra constraints as carried out in [8]. This transformation helps to run powerful sound and complete SMT-solvers [28, 30] and speed up the computation significantly. Additionally, we need to ensure that all N input segments for a Mixer-N must be filled with intermediate fluids or reagents (Eqn. 3), whereas, Mixer-N can serve at most N segments to other mixers (Eqn. 4). The required consistency constraints at depth i are enforced by:

$$x_i + y_i + w_{d,i} + w_{d-1,i} + \cdots + w_{i-1,i} = N \quad (3)$$

$$w_{i,i-1} + w_{i,i-2} + \cdots + w_{i,i} \leq N \quad (4)$$

Besides that, all weights must satisfy $0 \leq w_{i,j} \leq N-1$, for $1 \leq j < i \leq d$. Analogously, $0 \leq x_i, y_i \leq N-1$ for $1 \leq i \leq d$. Finally, the constraint $(X_1 = x) \wedge (Y_1 = y)$ guarantees that the desired target ratio of sample and buffer $\{x : y\}$ is produced.

B. Enforcing the Available Number of Storage-Units

Next, we have to enforce that not more than the available number of storage-units is used. To this end, we compute the number of requires storage-units which would be needed according to a particular assignment of the variables by traversing the mixing nodes in a bottom-up fashion. For depth i, the storage variable s_i denotes the number of on-chip storage-units required for executing the portion of induced subgraph containing mixing nodes at depth $i, i+1, \cdots, d$. This amount is determined by the following equation:

$$s_i = \begin{cases} s_{i+1} + w_{d,i} + w_{d-1,i} + \cdots + w_{i+2,i},\ 1 \leq i < d \\ 0, \quad i = d \end{cases} \quad (5)$$

Accordingly, s_1 denotes the storage requirement for the entire mixing graph. Restricting this variable to the number k of available storage-units, i.e., enforcing $s_1 \leq k$, only allows assignments for all other variables which eventually represent solutions that do not use more than k storage-units.

Fig. 10 shows the induced subgraph of the general mixing graph from Fig. 8(b) used in the storage calculation. Note that no storage-unit is required for subgraph containing mixing node at depth d only. Hence $s_d = 0$ (basis of the induction). Assume that, s_{i+1} be the minimum number of on-chip storage-units required to execute the subgraph induced by the nodes at depth $i+1, i+2, \cdots, d$ (induction hypothesis). For computing storage requirement of the induced subgraph given in Fig. 10, i.e., s_i, we need to add s_{i+1} and the total number of on-chip storage-units used for storing unused segments at depth $i+2, i+3, \cdots, d$, that are used in the mixing node at depth i. Hence, $s_i = s_{i+1} + w_{d,i} + w_{d-1,i} + \cdots + w_{i+2,i}$. This proves the correctness of Eqn. (5). Therefore, the bottom-up computation of s_1 gives the minimum number of on-chip storage-units for executing a dilution graph on the CFMB platform (Fig. 1(b)) equipped with a single mixer. Note that it is up to the optimizing SMT-solver [28, 30] that gives a minimal-reagent solution satisfying storage constraint.

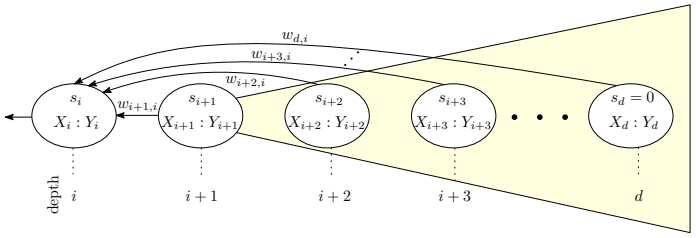


Fig. 10: Structure of the induced subgraph in the general dilution tree, containing mixing nodes at depth $i, i+1, \cdots, d$, used for storage computation.

Example 3. *Fig. 11 shows solutions realizing the target ratio* $\{$*sample : buffer* $= 125 : 131\}$ *with the least reactant-cost based on the graph of Fig. 8(a) and considering the availability of a different number of on-chip storage-units on the considered architecture given in Fig. 1(b). The proposed algorithm modifies the graph based on the number of available storage-units (#storage-units* $= 0, 1, \cdots, 5$*).*

V. Proposed Method for Storage-Aware Mixing

Mixture preparation with microfluidic chips where more than two fluids are involved, poses a significant challenge compared to dilution when storage constraints are enforced. This is due to the inherent tree structure showing up in the baseline mixing algorithm genMixing [8] and the underlying graph structure considered by the reagent saving mixing algorithm FloSPA [8], which minimizes the reagent usage by facilitating the sharing of intermediate fluids between non-adjacent levels. To enforce the storage constraint on the mixture preparation, we need to investigate the general structure of an underlying mixing trees/graphs and devise suitable constraints for it. We also need to ensure correctness of the mixing ratio and storage constraints. Moreover, existing mixing algorithms (genMixing and FloSPA) do not account for storage constraint. Hence, the number of storage-units needed by them may not be available on-chip. The following example demonstrates the need for storage-aware mixing algorithm.

Example 4. *On-chip implementation of the mixing tree generated with genMixing (Fig. 12) uses twenty five units of input reagents and two storage elements for storing intermediate*

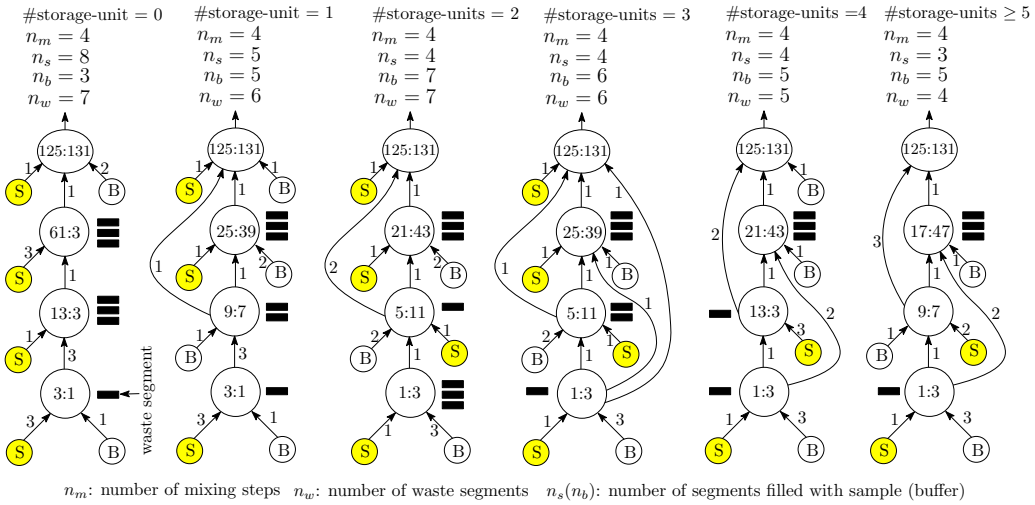


Fig. 11: Reagent minimal solution for the target ratio $\{\text{sample} : \text{buffer} = 125 : 131\}$ considering the availability of different number of available on-chip storage-units on the CFMB architecture having one Mixer-4.

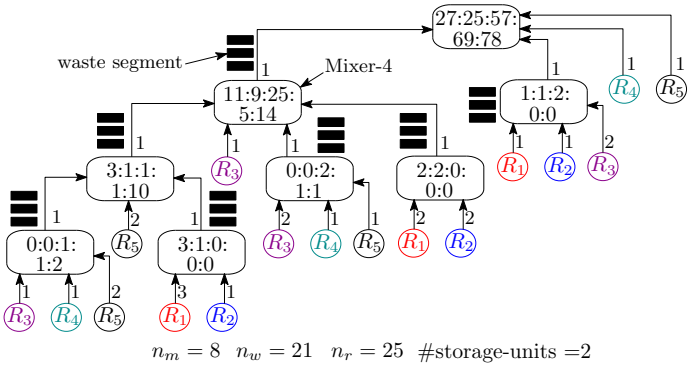


Fig. 12: Mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 : R_5 = 27 : 25 : 57 : 69 : 78\}$ produced by genMixing [8].

fluids. On the other hand, FloSPA reduces the consumption of input reagent significantly (Fig. 13(c)) but it needs four on-chip storage elements. The proposed storage-aware mixing algorithm can produce the desired mixing ratio considering the limited availability of on-chip storage. Fig. 13(a) shows the mixing graph that use only one storage and Fig. 13(b) shows the same when the available number of storage is two or three. Moreover, the proposed method guarantees no solution for zero storage. It can be observed that, we can achieve comparable performance with the best known reagent-saving algorithm FloSPA by deploying only two on-chip storage elements.

In the following, we describe how, starting with this mixing tree generated with an existing sample preparation algorithm genMixing [8], the proposed method determines a storage-aware solution. To this end, we first review the differences between dilution and mixing. Afterwards, we describe how to augment the corresponding mixing tree so that a storage-aware solution can be derived.

A. Overview of Storage-Aware Mixture Preparation

In order to describe the proposed method for storage-aware mixing, we first review the differences between dilution and mixing. To keep the descriptions accessible, we illustrate the respective contents with the following example:

Example 5. *Consider the target mixing ratio of four input reagents* $\{R_1 : R_2 : R_3 : R_4 = 0.34 : 0.22 : 0.22 : 0.22\}$, *where* $0.34+3\times0.22 = 1$. *The target ratio can be transformed as* $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ $(22 + 3 \times 14 = 4^3)$ *for Mixer-4 (i.e.,* $N = 4$*) and the user-defined error-tolerance limit might be set to* $\epsilon = 0.004$. *Note that* $\max\{|0.34 - \frac{22}{4^3}|, |0.22 - \frac{14}{4^3}|\} = 0.003 < \epsilon$, *i.e., d is chosen to be 3.*

For the transformed ratio, genMixing represents each ratio component as a d-bit base-N number and scans the digits from right-to-left in order to create a mixing tree in a bottom-up fashion (see Fig. 14). Note that the numbers 22 and 14 are represented in base-4 $(112)_4$ *and* $(032)_4$, *respectively. Two units of each input reagent are used in the mixing node at depth 3 as the least significant digit of each ratio is 2. Similarly, one unit of* R_1 *and three units of* R_2, R_3, *and* R_4 *are used at depth 2. Moreover, only one unit of intermediate fluid is shared between the mixing nodes that appear in adjacent depth of the mixing tree.*

In contrast to dilution, it may not always be possible to perform mixing of three or more reagents without using any on-chip storage-units. Moreover, each intermediate mixing node has $N - 1$ units of waste. The method proposed in [8] used a SAT-based technique that significantly reduced the intermediate waste, by augmenting the basis tree (generated by genMixing) with additional nodes (representing input reagents), and edges between non-adjacent levels. However, the SAT modeling proposed in [8] was oblivious to storage limitation on a microfluidic platform. Here, we propose an advanced SAT formulation addressing storage constraints during mixture preparation.

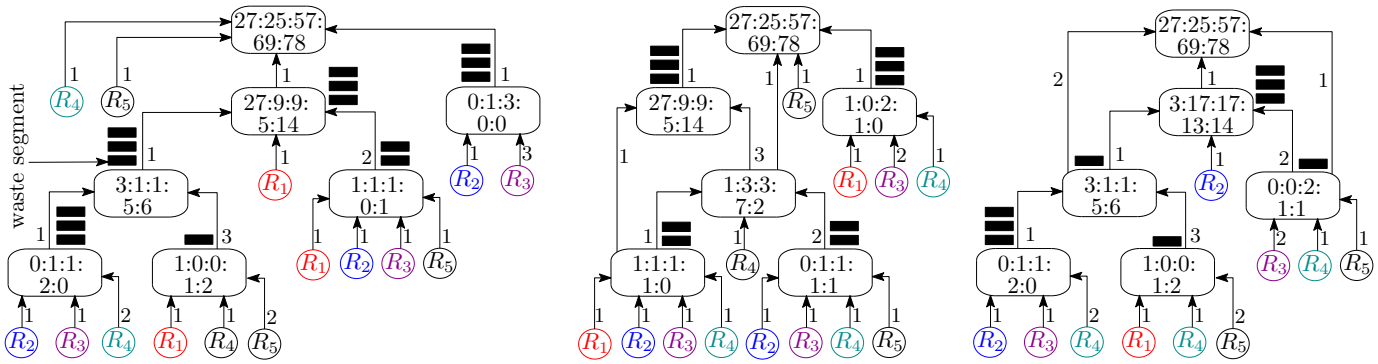


Fig. 13: Mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 : R_5 = 27 : 25 : 57 : 69 : 78\}$ produced by the proposed storage aware mixing algorithm: (a) #storage-units = 1, (b) #storage-units $= 2, 3$, and (c) #storage-units ≥ 4 and FloSPA [8].

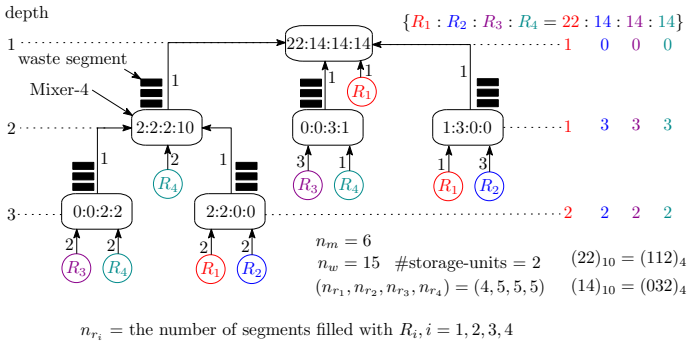


Fig. 14: Mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ produced by genMixing [8].

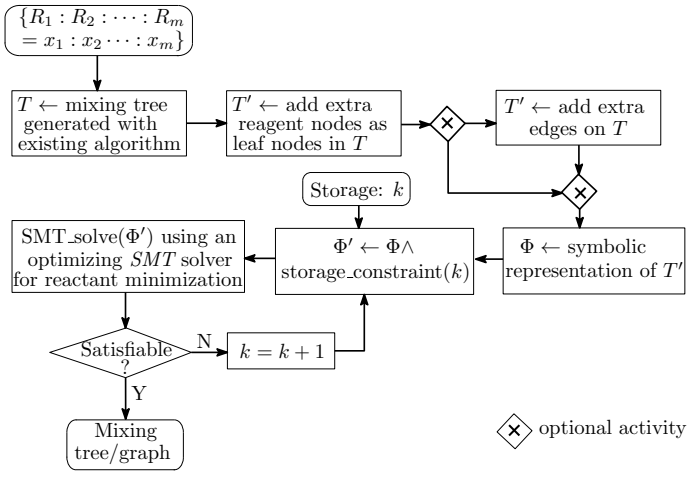


Fig. 15: Flowchart of the proposed algorithm for mixing.

As before, we start with a basis mixing tree (T) produced by genMixing [8] and it is augmented with additional leaf-nodes (input reagents) and edges (optional) – allowing for further options to realize the desired ratio. We denote the augmented mixing tree (after adding additional reagent nodes) or graph (after adding additional reagent nodes and edges between non-adjacent mixing nodes) graph as T'.

Then, we introduce variables and constraints on T' to provide a symbolic representation (Φ) of all possible options of T' out of which the best one satisfying the storage limitation is determined by the solving engine. To this end, we particularly adjust the storage constraints over Φ in order to generate a storage-constrained symbolic representation (Φ').

After that, Φ' is passed to the SAT-solver for checking whether the desired input ratio can be generated with a mixing tree/graph that uses at most k on-chip storages. If Φ' is satisfiable, then the desired mixing tree/graph can be obtained from the satisfying assignment to all variables. Otherwise (Φ' is unsatisfiable), no such mixing tree/graph exists with only k on-chip storage elements. In this case, k is incremented by one and the process is repeated until a satisfying solution and, hence, a corresponding mixing tree/graph is obtained. Following this flow, a minimal number of on-chip storages is guaranteed[3]. Fig. 15 summarizes this overall flow.

In the remainder of this section, we now describe the detailed modeling of the proposed storage-aware mixture preparation. We divide the descriptions into two parts: First, we cover the detailed SAT modeling (including the addition of storage constraints) for a simpler case where the basis graph is augmented with extra reagent nodes, i.e., restricting the sharing of intermediate fluids between adjacent levels. Afterwards, we consider the more general case where the basis graph is augmented with extra reagent nodes and edges, i.e., intermediate fluids can be shared between nonadjacent mixing nodes. For the sake of clarity, we intentionally skip of detailed coverage of the original SAT modeling Φ and, instead, focus on the enforcement of the desired storage constraints.

B. Augmenting the Tree for Adjacent-Level Sharing

We start with the simplest case, in which we augment the basis mixing graph with extra input reagents only, i.e. how to

[3]Note that designers might not necessarily be interested in the minimum number of on-chip storages, but just want to make sure that their aspect ration can be realized with an upper bound of them. Then, he/she can simply check whether this is possible by setting k to this upper bound.

get from the basis tree (generated with genMixing [8]) to the augmented tree. For example, the mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ generated by genMixing [8] (and as discussed before in Fig. 14) shall be augmented with all missing reagents as shown in Fig. 16(a).

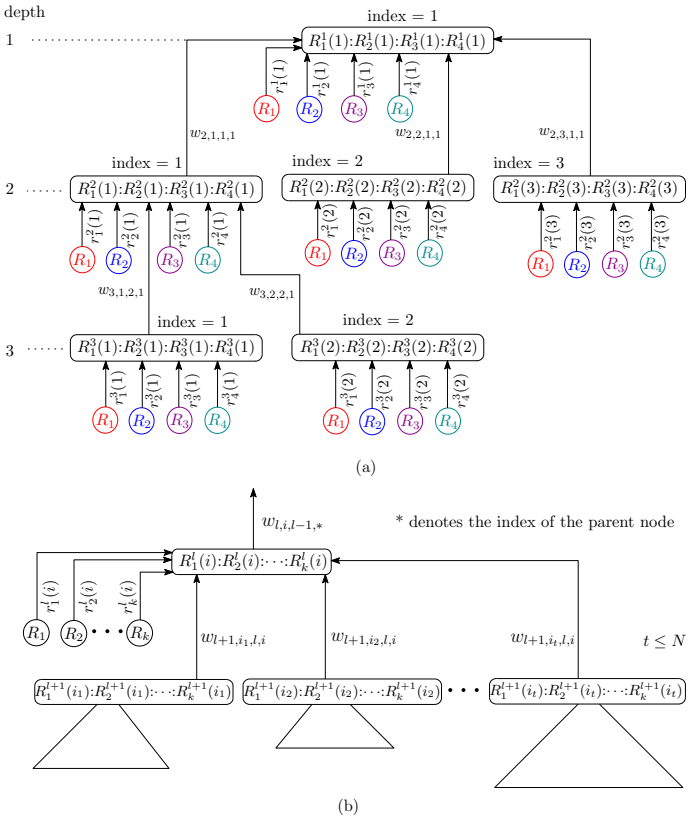


Fig. 16: (a) Mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ after adding extra reagents nodes for enabling reagent minimization (b) general structure of a mixing node.

Symbolic Formulation

The symbolic formulation is defined over the following variables:

Node variables: For every mixing node of the mixing tree, k rational node variables $R_1^l(i), R_2^l(i), \cdots, R_k^l(i)$ are assigned for denoting the portion of input reagents $R_1, R_2, \cdots, R_k$, respectively, where l is the depth of that node and i is the index of the node at depth l. Note that, at each depth, the index of a mixing node is assigned in a left-to-right fashion starting with 1. For example, the node variables corresponding to the leftmost node of depth 2 in Fig. 16(a) are $R_1^2(1), R_2^2(1), R_3^2(1), R_4^2(1)$.

Reagent variables: We define rational reagent variables $r_j^l(i)$, where $j = 1, 2, \ldots, k$, to denote the number of segments filled with reagent R_j, where l is the depth of the mixing node where reagent R_j is used, and i is the index of the mixing node. For example, the reagent variables corresponding to the leftmost node on depth 2 in Fig. 16(a) are $r_1^2(1), r_2^2(1), r_3^2(1), r_4^2(1)$.

Segment-sharing variables: An edge (u, v) between two mixing nodes denotes the sharing of intermediate fluids from node u to node v in the mixing tree. A rational edge variable w_{l_1,i_1,l_2,i_2} denotes the number of segments shared between a node with index i_1 at depth l_1, and a node with index i_2 at depth l_2. For example, edge weight $w_{3,2,2,1}$ in Fig. 16(a) denotes the number of segments shared from the node indexed as 2 at depth 3 (the rightmost node at depth 3) to the node with index 1 at depth 2 (the leftmost node at depth 2).
Passing this formulation to a SAT solver will result an arbitrary assignments to the variables (since they are not restricted). Hence, as a next step, we add constraints which enforce that desired target ratio is achieved, and afterwards, the number of storage-units is considered.

Enforcing the Desired Ratio

In order to enforce the desired target ratio at the root node of the augmented mixing tree, we need to ensure that the mixing ratio generated at each internal node is correct. Consider the general structure of a node in the augmented mixing tree as shown in Fig. 14. Note that a segment of intermediate fluid can be shared between adjacent levels. Moreover, on the mixing tree, a mixing node may receive a segment of fluid from one of its subtrees (second part of the left hand side of Eqns. 6) or it may take input reagents to fill one of its N segments (first part of the left hand side of Eqns. 6). Hence, there can be at most N subtrees possible for a mixing node.

The ratio consistency conditions of the mixing node at depth l and index i are given as follows:

$$\begin{aligned}
r_1^l(i) + \sum_{j\in\{i_1,i_2,\cdots,i_t\}} R_1^{l+1}(j)\cdot w_{l+1,j,l,i} &= N\cdot R_1^l(i)\\
r_2^l(i) + \sum_{j\in\{i_1,i_2,\cdots,i_t\}} R_2^{l+1}(j)\cdot w_{l+1,j,l,i} &= N\cdot R_2^l(i)\\
\ddots \qquad\qquad\qquad\qquad\qquad\qquad &\vdots\\
r_k^l(i) + \sum_{j\in\{i_1,i_2,\cdots,i_t\}} R_k^{l+1}(j)\cdot w_{l+1,j,l,i} &= N\cdot R_k^l(i)
\end{aligned} \tag{6}$$

Moreover, we need to ensure that all N input segments for a Mixer-N are filled with intermediate fluids/input reagents (otherwise, the concerned mixing operations are not necessary for generating the desired target ratio). This yields the following consistency conditions:

$$\begin{aligned}
&\left(\sum_{j=1}^{k} r_j^l(i) + \sum_{j\in\{i_1,i_2,\cdots,i_t\}} w_{l+1,j,l,i} = N\right) \bigvee\\
&\left(\sum_{j=1}^{k} r_j^l(i) + \sum_{j\in\{i_1,i_2,\ldots,i_t\}} w_{l+1,j,l,i} = 0\right),\\
&0 \leq w_{l,i,l-1,*} \leq N-1
\end{aligned} \tag{7}$$

All weights must satisfy $0 \leq w_{l+1,j,l,i} \leq N-1$, for $j \in \{i_1, i_2, \ldots, i_t\}$. Analogously, $0 \leq r_j^l(i) \leq N-1$ for $j = 1, 2, \ldots, k$.

Passing the resulting formulation to a SAT solver yields a satisfying assignment out of which a mixing tree realizing the desired aspect ratio can be obtained or proves that no such solution exist.

Enforcing the Number of Storage-Units

We consider the generic node structure, and for sake of simplicity, use a simplified notation for the reagent and segment-sharing variables. Fig. 17 sketches a generic node in a mixing graph as well as a simplified notation of the variables that are used in storage calculation. In Fig. 17, w_i (segment sharing variable) and r_i (reagent variable) denote the number of segments of a Mixer-N is filled with fluids taken from the root node of subtree-i and the input reagent R_i, respectively. Besides that, s_i denotes the storage requirement of subtree-i. Then, the number of storage-units (s) can be determined with the following equation:

$$s = \bigvee_{j=1}^{t} \left(\sum_{i=1}^{j-1} w_i + s_j + \sum_{i=j+1}^{t} w_i \right) \tag{8}$$

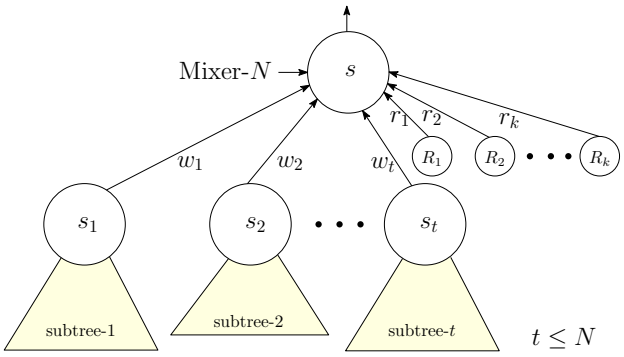


Fig. 17: Structure of a node in the mixing tree used in storage computation.

Note that Equation 8 symbolically represents various scheduling issues for each node. As in dilution, enforcing the corresponding s-variables of all mixing nodes in the mixing tree to be smaller or equal than k denoting the available storage-units, will eventually allow only those solutions, which can be realized under this restriction.

Example 6. *Figs. 18(a)-(b) show the mixing graphs for the target ratio* $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ *obtained without enforcing storage limitation, i.e., by using FloSPA [8] and by the proposed method, respectively. Moreover, Fig. 14 shows the same ratio generated with genMixing [8], which requires two storage-units. Fig. 18(a) shows a cheaper solution by the proposed method. Note that FloSPA also returns this solution, which is oblivious to storage restriction. On the other hand, the proposed method ensures that no solution exists with zero storage-unit when Fig. 14 is used as the base graph; it also provides a solution with one storage-unit (Fig. 18(b)).*

C. Augmenting the Tree for Non-Adjacent Level Sharing

So far we have considered sharing of intermediate fluids between adjacent levels only. However, more reagents can

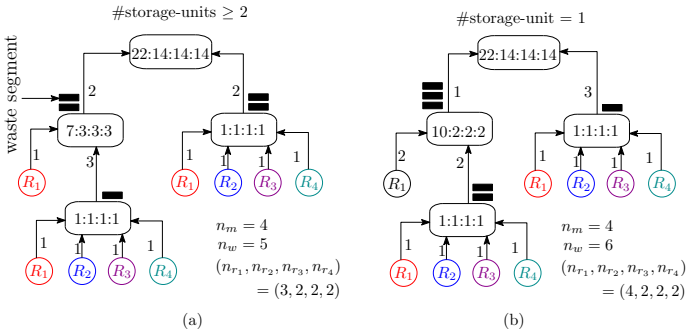


Fig. 18: Mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ generated by (a) FloSPA [8] and by the proposed method for #storage-units ≥ 2 and (b) by the proposed method when #storage-unit = 1.

be saved if we allow sharing of intermediate fluids between ancestors. In order to enable such reagent sharing, additional edges have to be introduced into the basis tree. Fig. 19 shows the basis graph after augmenting reagent nodes and edges for enabling intermediate fluid sharing between a node and its ancestors in the basis tree. Extra edges and their corresponding variables are shown by blue color in Fig. 19.

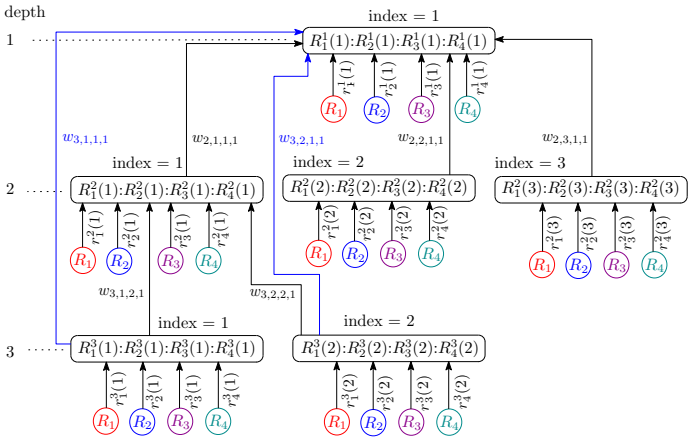


Fig. 19: Augmenting basis tree generated by genMixing [8] with extra reagent nodes and edges for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$.

Before adding storage constraints on the augmented graph, we need to enforce the correctness of the mixing ratio at each mixing node. The modeling will be similar to that used in the case of mixing tree (Section V-B), with certain modifications to ratio condition (Equation 6), and mixer consistency (Equation 7).

Enforcing the Number of Storage-Units

Deriving the constraints corresponding to the storage requirement for this case is more complex, and to incorporate them, we start with the general structure of a mixing node. Fig. 20 shows the generic structure of a mixing node in the augmented graph.

Note that each subtree in Fig. 20 has a similar general structure as the dilution graph (cf. Fig. 8). In case of Fig. 20,

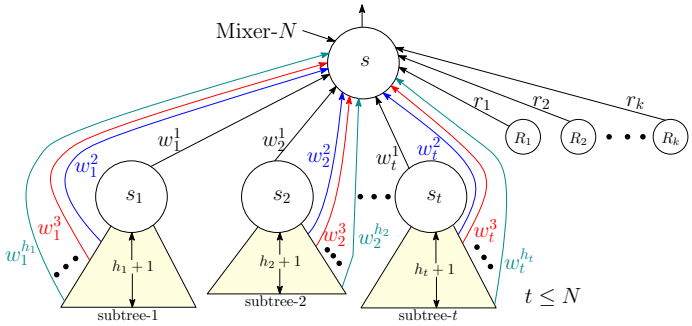


Fig. 20: Structure of a node in the mixing graph used in storage computation.

we need to consider multiple subtrees – each of them sharing the same structure as in dilution. The following constraint calculates the storage requirement:

$$s = \bigvee_{j=1}^{t} \left(\sum_{i=1}^{j-1} w_i^1 + s_j + \sum_{i=j+1}^{t} w_i^1 \right) + \sum_{i=2}^{h_1} w_1^i + \sum_{i=2}^{h_2} w_2^i + \cdots + \sum_{i=2}^{h_t} w_t^i \quad (9)$$

In order to define storage requirement (s) of the root node, we consider all possible ordering of its subtree execution. The first part of Equation 9 models this. Moreover, for each edge arriving from its grandchildren (highlighted in blue), we need to store intermediate fluids into on-chip storage cells. The second part of Equation 9 represents this.

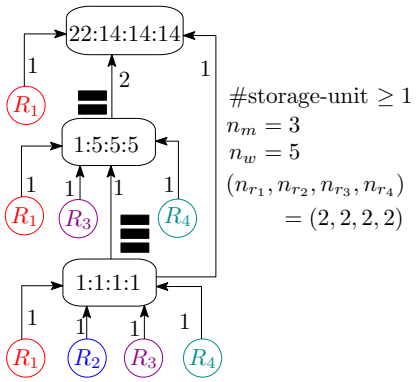


Fig. 21: Mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ generated by the proposed method.

Example 7. *Fig. 21 shows the mixing tree for the target ratio* $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$*, when the basis graph is augmented with additional reagent nodes and edges. Note that the resultant mixing graph has a cheaper solution compared to the mixing trees shown in Figs. 18(a)-(b).*

VI. Experimental Results

The methods described above have been implemented and compared with several state-of-the-art sample-preparation methods, namely NWayMix [8] and VOSPA [6] for dilution as well as with FloSPA [8] for dilution and general mixing. All experiments have been performed on an Intel machine with 3.5 GHz and 16 GB of main memory running an Ubuntu 16.04 LTS operating system. In order to solve the proposed storage-aware dilution/mixing instances, we invoke *Z3OPT* [28, 30], an optimized SMT solver. In the following, the results for both cases are summarized and discussed.

A. Performance for Dilution

VOSPA [6] and FloSPA [8] provide efficient solutions for dilution problems. They focus on reagent minimization and do not take the number of available storage-units into account. This can have crucial consequences as illustrated by a first series of experiments summarized in Fig. 22. Here, we summarized the results obtained by these methods for all possible target ratios with $d = 4, 5$ and $N = 4$ and report the data for number of storage-units required by different ratios. As can be seen, for the vast majority of ratios, both approaches require a substantial amount of storage-units. If this amount is not available on the available microfluidic platform, these results may not be useful. In contrast, the approach proposed in this work is capable of determining a mixing graph for all ratios and for all given numbers of available storage-units (even zero). This is a clear improvement compared to VOSPA and FloSPA since the desired dilution can always be realized using the method proposed in this work.

Moreover, even with respect to cost, significant improvements can be observed as shown in Table II. Here, we have generated dilution graphs for all possible target ratios considering $\{\text{sample} : \text{buffer} = x : 256 - x\}$, where $1 \leq x \leq 255$, i.e., $N = 4, d = 4$; we list the average number of mixing steps ($\bar{n}_m$), waste segments ($\bar{n}_w$), number of segments filled with sample ($\bar{n}_s$) and buffer ($\bar{n}_b$) as well as the average running time[4] (t_{avg}) of each target ratio for different values of on-chip storage-units (k). For previously proposed approaches, we list the best results, i.e., those obtained with zero storage-unit for NWayMix, with seven storage-units in case of VOSPA, and with six storage-units for FloSPA.

Several issues can be observed here: First, NWayMix always uses zero storage-units. By this, other potential solutions are missed out since a few storage-units are usually available on any platform which, as shown by the numbers in Table II, can be exploited to reduce reactant usage. Either way, even with zero storage-unit, the proposed approach still determines better results than NWayMix. It can also be observed from Table II that the proposed approach only needs two (four) storage-units in order to get a comparable performance with respect to VOSPA (FloSPA) requiring a total of seven (six) storage-units.

B. Performance for Mixing

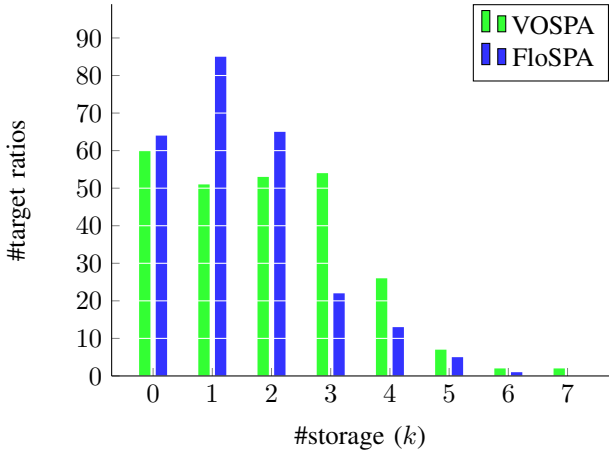
Finally, we have experimented with the proposed storage-aware mixing algorithms by considering several synthetic as well as real-life mixing ratios for various values of available

[4]dilution graph can be generated within a minute on the average.

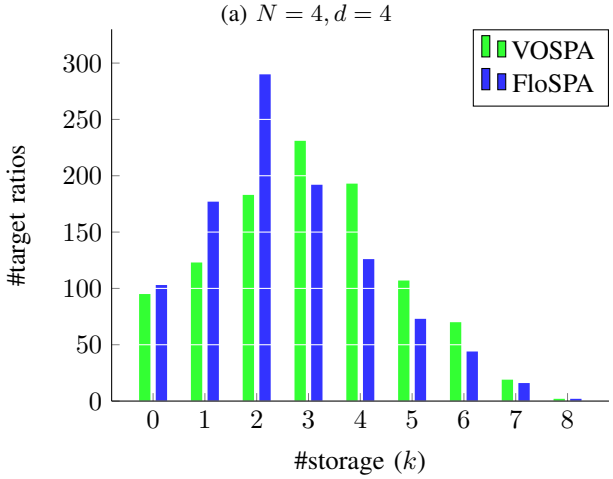
TABLE II: Performance of the proposed dilution algorithm.

Proposed approach						NWayMix [8]	VOSPA [6]	FloSPA [8]
k	$\bar{n}_m$	$\bar{n}_w$	$\bar{n}_s$	$\bar{n}_b$	t_{avg}	#storage-unit = 0	#storage-units = 7	#storage-units = 6
0	3.68	5.39	4.22	5.18	0.07 sec.	$\bar{n}_m = 3.68$	$\bar{n}_m = 4.13$	$\bar{n}_m = 3.68$
1	3.68	4.19	3.51	4.67	0.26 sec.	$\bar{n}_w = 8.04$	$\bar{n}_w = 6.70$	$\bar{n}_w = 3.66$
2	3.68	3.80	3.40	4.40	0.52 sec.	$\bar{n}_s = 6.02$	$\bar{n}_s = 3.34$	$\bar{n}_s = 3.36$
3	3.68	3.71	3.38	4.32	0.67 sec.	$\bar{n}_b = 6.02$	$\bar{n}_b = 7.36$	$\bar{n}_b = 4.30$
≥ 4	3.68	3.66	3.36	4.30	0.74 sec.			

k: number of storage.
$\bar{n}_m/\bar{n}_w/\bar{n}_s/\bar{n}_b/t_{avg}$: average mix/waste/sample/buffer/CPU-time.



(a) $N = 4, d = 4$



(b) $N = 4, d = 5$

Fig. 22: Histogram of the number of storage-units required by VOSPA and FloSPA for all target ratios corresponding to (a) $\{\text{sample : buffer} = x : 256 - x\}$, where $1 \leq x \leq 255$, i.e., $N = 4, d = 4$ and (b) $\{\text{sample : buffer} = x : 1024 - x\}$, where $1 \leq x \leq 1023$, i.e., $N = 4, d = 5$.

on-chip storage-units (k) and summarized the results in Table III. Here, we list the number of mixing steps (n_m), waste segments (n_w), the total number of segments filled with input reagents (n_r), and the CPU-time required to generate a mixing tree/graph, for a number of mixing ratios.

It can be observed that the proposed method based on fluid sharing exclusively between adjacent levels can perform mixture preparation using only a few on-chip storage-units. In fact, all ratios can be realized with at most two storage-units – in many cases even only one is sufficient. Moreover, it not only takes the same or fewer number of on-chip storage-units compared to genMixing but also produces the target ratio with a smaller number of mixing steps, waste, and input segments.

Adoption of non-adjacent level sharing of intermediate fluid units can save reagents further. The enhancement in quality of the solution comes with the increase of running time. One may easily observe from Table III that the SMT-solver finds the best solutions quickly for most of the instances. Note that we generate the mixing tree offline, so we can afford a little longer time to get a solution that runs faster and cheaper in the microfluidic platform in real-time.

Note that FloSPA when used for mixture preparation, reduces the cost of input reagents disregarding the availability of on-chip storage-units. In the case of Mixture 5 and Mixture 11, both FloSPA and the proposed method for non-adjacent level sharing (highlighted with yellow color in Table III) require four on-chip storage-units when the optimization objective is to minimize reagent-usage. However, the latter can produce the same mixture at the cost of more input reagents even when a smaller number of storage-units ($k = 2$) is made available.

VII. Conclusions

In this work, we have proposed a storage-aware sample-preparation method for continuous-flow microfluidic biochips. To this end, we augmented mixing graphs determined by previously proposed algorithms providing several further options for realizing the desired input ratio. Afterwards, Boolean satisfiabiliy solvers are utilized to determine the option that gives the minimum reagent-usage and, at the same time, satisfies the limitations in storage elements. This provides significant benefits as it can be ensured that a generated mixing graph indeed can be executed on the biochip device (compared to previously proposed solutions which may generate mixing graphs that require more storage than available). Moreover, the proposed approach explicitly allows to fully exploit the available number of storages, e.g., in order to reduce the use of reagents. Although we have used an equally-segmented mixer in the proposed method, the SAT-based method can be extended, in principle, to handle mixers with unequal segments. Such generalization can be investigated as a future research problem.

TABLE III: Performance of the proposed mixing algorithms.

Synthetic mixing ratio										
#	Mixing ratio	genMixing [8]	Prop. method (adj. level sharing)			Proposed method (non-adjacent level sharing)				
		(n_m,n_w,n_r) k	$k=0$	$k=1$	$k\geq 2$	$k=0$	$k=1$	$k=2$	$k=3$	$k\geq 4$
1	90:90:76	(5, 12, 16) 1 0.00 sec.	(4, 8, 12) 0.07 sec.	(5, 6, 10) 0.30 sec.	(5, 6, 10) 0.45 sec.	(4, 8, 12) 0.14 sec	(4, 3, 7) 1.19 sec.	(4, 2, 6) 2.73 sec.	(4, 2, 6) 4.33 sec.	(4, 2, 6) 5.77 sec.
2	20:14:16:14	(4, 9, 13) 1 0.00 sec.	NA	(4, 6, 10) 0.18 sec	(4, 5, 9) 0.15 sec.	NA	(3, 3, 7) 0.15 sec.	(3, 3, 7) 0.21 sec.	(3, 3, 7) 0.21 sec.	(3, 3, 7) 0.22 sec.
3	27:7:13:17	(5, 12, 16) 1 0.00 sec.	NA	(5, 8, 12) 0.28 sec.	(5, 8, 12) 1.48 sec.	NA	(5, 8, 12) 0.44 sec.	(4, 6, 10) 0.85 sec.	(4, 6, 10) 1.55 sec.	(4, 6, 10) 1.31 sec.
4	68:86:56:46	(6, 15, 19) 1 0.00 sec.	NA	(5, 9, 13) 1.35 sec.	(5, 9, 13) 3.82 sec.	NA	(4, 5, 9) 2.33 sec.	(4, 5, 9) 11.16 sec.	(4, 5, 9) 20.90 sec.	(4, 5, 9) 20.14 sec.
5	27:25:57:69:78	(8, 21, 25) 2 0.00 sec.	NA	(7, 15, 19) 0.90 sec	(7, 13, 17) 7.90 sec.	NA	(7, 15, 19) 3.08 sec.	(6, 10, 14) 1553.52 sec.	(6, 10, 14) 3719.85 sec.	(6, 9, 13) 4094.46 sec.
6	30:24:55:68:79	(8, 21, 25) 2 0.00 sec.	NA	(6, 13, 17) 1.34 sec.	(6, 12, 16) 10.31 sec	NA	(6, 13, 17) 5.60 sec.	(5, 9, 13) 1601.56 sec.	(5, 7, 11) 2831.40 sec.	(5, 7, 11) 6391.80 sec.
7	30:24:155:38:9	(8, 21, 25) 2 0.00 sec.	NA	(7, 15, 19) 1.68 sec.	(7, 13, 17) 8.59 sec.	NA	(7, 15, 19) 5.31 sec.	(6, 10, 14) 269.07 sec	(6, 10, 14) 1434.89 sec.	(6, 10, 14) 2018.36 sec.
Mixing ratios from real-life bio-protocols										
#	Mixing ratio	genMixing [8]	Prop. method (adj. level sharing)			Proposed method (non-adjacent level sharing)				
		(n_m,n_w,n_r) k	$k=0$	$k=1$	$k\geq 2$	$k=0$	$k=1$	$k=2$	$k=3$	$k\geq 4$
8	Plasmid DNA by alkaline lysis with SDS miniprep. [34] 300:499:225	(7, 18, 22) 1 0.00 sec.	(5, 4, 8) 0.10 sec.	(5, 4, 8) 1.96 sec.	(5, 4, 8) 2.13 sec.	(5, 4, 8) 0.29 sec.	(5, 4, 8) 12.02 sec.	(5, 4, 8) 455.92 sec.	(5, 4, 8) 777.81 sec.	(5, 4, 8) 1414.57 sec.
9	Plasmid DNA [35] 57:28:6:6:6:3:150	(9, 24, 28) 2 0.00 sec	NA	(6, 10, 14) 1.76 sec.	(6, 10, 14) 11.14 sec.	NA	(6, 10, 14) 5.41 sec.	(6, 10, 14) 208.10 sec.	(6, 10, 14) 904.18 sec.	(6, 10, 14) 1023 sec.
10	Splinkerette PCR [35] 102:26:3:3:122	(8, 21, 25) 2 0.00 sec.	NA	(7, 12, 16) 2.39 sec.	(7, 12, 16) 5.08 sec.	NA	(7, 12, 16) 8.07 sec.	(5, 7, 11) 34.71 sec.	(5, 7, 11) 78.77 sec.	(5, 7, 11) 117.29 sec.
11	PCR master mix [36] 4:6:10:14:22:26:174	(10, 27, 31) 3 0.00 sec.	NA	NA	(7, 14, 18) 84.44 sec.	NA	NA	(7, 14, 18) 985.81 sec.	(7, 14, 18) 1085.87 sec.	(6, 12, 16) 7132.16 sec.
12	Touchdown PCR [35] 26:15:51:26:5:5:1:127	(12, 33, 37) 3 0.00 sec.	NA	NA	(10, 24, 28) 6.93 sec.	NA	NA	(8, 16, 20) 1572.46 sec.	time out*	time out*
13	Silver-Restriction Digest [35] 180:26:26:5:5:14	(8, 21, 25) 3 0.00 sec.	NA	NA	(8, 18, 22) 16.25 sec.	NA	NA	(6, 11, 15) 21.55 sec.	(6, 11, 15) 102.54 sec.	(6, 11, 15) 136.42 sec.

Each entry shows the parameter values (n_m,n_w,n_r) and the CPU time for finding the mixing tree/graph.
The best parameter values (total number of reagents i.e., n_r) are highlighted with yellow color.
* Time out was set to two hours.

REFERENCES

[1] S. Bhattacharjee, R. Wille, J.-D. Huang, and B. B. Bhattacharya, "Storage-aware sample preparation using flow-based microfluidic labs-on-chip," in *Proc. of DATE*, 2018, pp. 1399–1404.

[2] Introduction to lab-on-a-chip 2015 : review, history and future. [Online]. Available: http://www.elveflow.com/microfluidic-tutorials/microfluidic-reviews-and-tutorials/introduction-to-lab-on-a-chip-2015-review-history-and-future/

[3] C. D. Chin, V. Linder, and S. K. Sia, "Commercialization of microfluidic point-of-care diagnostic devices," *Lab Chip*, vol. 12, pp. 2118–2134, 2012.

[4] S. Roy, B. B. Bhattacharya, and K. Chakrabarty, "Optimization of dilution and mixing of biochemical samples using digital microfluidic biochips," *IEEE Trans. on CAD*, vol. 29, no. 11, pp. 1696–1708, 2010.

[5] J.-D. Huang, C.-H. Liu, and T.-W. Chiang, "Reactant minimization during sample preparation on digital microfluidic biochips using skewed mixing trees," in *Proc. of ICCAD*, 2012, pp. 377–383.

[6] C.-M. Huang, C.-H. Liu, and J.-D. Huang, "Volume-oriented sample preparation for reactant minimization on flow-based microfluidic biochips with multi-segment mixers," in *Proc. of DATE*, 2015, pp. 1114–1119.

[7] C.-H. Liu, K.-C. Shen, and J.-D. Huang, "Reactant minimization for sample preparation on microfluidic biochips with various mixing models," *IEEE Trans. on CAD*, vol. 34, no. 12, pp. 1918–1927, 2015.

[8] S. Bhattacharjee, S. Poddar, S. Roy, J.-D. Huang, and B. B. Bhattacharya, "Dilution and mixing algorithms for flow-based microfluidic biochips," *IEEE Trans. on CAD*, vol. 36, no. 4, pp. 614–627, 2017.

[9] S. W. Dutse and N. A. Yusof, "Microfluidics-based lab-on-chip systems in DNA-based biosensing: An overview," *Lab Chip*, vol. 11, pp. 5754–5768, 2011.

[10] P. Neuži, S. Giselbrecht, K. Länge, T. J. Huang, and A. Manz, "Revisiting lab-on-a-chip technology for drug discovery," *Nat. Rev. Drug Discovery*, vol. 11, pp. 620–632, 2012.

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

[12] P. Pop, I. E. Araci, and K. Chakrabarty, "Continuous-flow biochips: Technology, physical-design methods, and testing," *IEEE Design & Test*, vol. 32, no. 6, pp. 8–19, 2015.

[13] J. Melin and S. Quake, "Microfluidic large-scale integration: The evolution of design rules for biological automation," *Annual Reviews in Biophysics and Biomolecular Structure*, vol. 36, pp. 213–231, 2007.

[14] I. E. Araci and P. Brisk, "Recent developments in microfluidic large scale integration," *Current Opinion in Biotechnology*, vol. 25, pp. 60 – 68, 2014.

[15] J. P. Urbanski, W. Thies, C. Rhodes, S. P. Amarasinghe, and T. Thorsen, "Digital microfluidics using soft lithography," *Lab Chip*, vol. 6, pp. 96–104, 2006.

[16] W. Thies, J. P. Urbanski, T. Thorsen, and S. P. Amarasinghe, "Abstraction layers for scalable microfluidic biocomputing," *Natural Computing*, vol. 7, no. 2, pp. 255–275, 2008.

[17] C. Liu, B. Li, H. Yao, P. Pop, T. Ho, and U. Schlichtmann, "Transport or store? synthesizing flow-based microfluidic biochips using distributed channel storage," in *Proc. of DAC*, 2017, pp. 1–6.

[18] S. Poddar, S. Ghoshal, K. Chakrabarty, and B. B. Bhattacharya, "Error-correcting sample preparation with cyberphysical digital microfluidic lab-on-chip," *ACM Trans. Design Autom. Electr. Syst.*, vol. 22, no. 1, pp. 2:1–2:29, 2016.

[19] S. Poddar, R. Wille, H. Rahaman, and B. B. Bhattacharya., "Error-oblivious sample preparation with digital microfluidic lab-on-chip," *IEEE Trans. on CAD*, 2018.

[20] S. Saha, D. Kundu, S. Roy, S. Bhattacharjee, K. Chakrabarty, P. P. Chakrabarti, and B. B. Bhattacharya, "Factorization based dilution of biochemical fluids with micro-electrode-dot-array biochips," in *Proc. of ASP-DAC*, 2019, pp. 462–467.

[21] L. M. Fidalgo and S. J. Maerkl, "A software-programmable microfluidic device for automated biology," *Lab Chip*, vol. 11, pp. 1612–1619, 2011.

[22] E. C. Jensen, B. P. Bhat, and R. A. Mathies, "A digital microfluidic platform for the automation of quantitative biomolecular assays," *Lab Chip*, vol. 10, pp. 685–691, 2010.

[23] A. Grimmer, B. Klepic, T.-Y. Ho, and R. Wille, "Sound valve-control for programmable microfluidic devices," in *Proc. of ASP-DAC*, 2018, pp. 40–45.

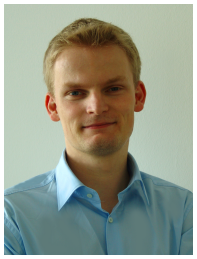
[24] T.-M. Tseng, B. Li, T.-Y. Ho, and U. Schlichtmann, "Reliability-aware synthesis for flow-based microfluidic biochips by dynamic-device mapping," in *Proc. of DAC*, 2015, pp. 141:1–141:6.

[25] Y. Zhu, B. Li, T.-Y. Ho, Q. Wang, H. Yao, R. Wille, and U. Schlichtmann, "Multi-channel and fault-tolerant control multiplexing for flow-based microfluidic biochips," in *Proc. of ICCAD*, 2018, pp. 123–128.

[26] Q. Wang, H. Zou, H. Yao, T.-Y. Ho, R. Wille, and Y. Cai, "Physical co-

design of flow and control layers for flow-based microfluidic biochips," *IEEE Trans. on CAD*, vol. 37, no. 6, pp. 1157–1170, 2018.
[27] Y.-C. Lei, T.-H. Lin, and J.-D. Huang, "Multi-objective sample preparation algorithm for microfluidic biochips supporting various mixing models," in *Proc. of SOCC*, 2016, pp. 96–101.
[28] L. M. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. of TACAS*, 2008, pp. 337–340, [Z3 is available at https://github.com/Z3Prover/z3].
[29] R. Wille, G. Fey, D. Große, S. Eggersglüß, and R. Drechsler, "SWORD: A SAT like prover using word level information," in *Proc. of VLSI-SoC*, 2007, pp. 88–93.
[30] N. Bjørner, A.-D. Phan, and L. Fleckenstein, "*vZ* - an optimizing SMT solver," in *Proc. of TACAS*, 2015, pp. 194–199.
[31] O. Keszocze, R. Wille, T.-Y. Ho, and R. Drechsler, "Exact one-pass synthesis of digital microfluidic biochips," in *Proc. of DAC*, 2014.
[32] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," in *Proc. of ASP-DAC*, 2017, pp. 530–535.
[33] A. Grimmer, W. Haselmayr, A. Springer, and R. Wille, "Design of application-specific architectures for networked labs-on-chips," *IEEE Trans. on CAD*, vol. 37, no. 1, pp. 193–202, 2018.
[34] Y.-L. Hsieh, T.-Y. Ho, and K. Chakrabarty, "A reagent-saving mixing algorithm for preparing multiple-target biochemical samples using digital microfluidics," *IEEE Trans. on CAD*, vol. 31, no. 11, pp. 1656–1669, 2012.
[35] S. Bhattacharjee, Y.-L. Chen, J.-D. Huang, and B. B. Bhattacharya, "Concentration-resilient mixture preparation with digital microfluidic lab-on-chip," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 2, pp. 49:1–49:12, 2018.
[36] S. Roy, P. P. Chakrabarti, S. Kumar, K. Chakrabarty, and B. B. Bhattacharya, "Layout-aware mixture preparation of biochemical fluids on application-specific digital microfluidic biochips," *ACM Trans. Design Autom. Electr. Syst.*, vol. 20, no. 3, pp. 45:1–45:34, 2015.

Sukanta Bhattacharjee (M'18) received the B.Tech. degree in computer science and engineering from the University of Calcutta, India and the M.Tech. and Ph. D. degrees in computer science from the Indian Statistical Institute, Kolkata, India. He is currently working as a post-doctoral associate in Center for Cyber Security, New York University, Abu Dhabi. His research interests include design automation algorithms for microfluidic biochip, formal methods, and security.



Robert Wille (M'06-SM'09) is Full Professor at the Johannes Kepler University Linz, Austria. He received the Diploma and Dr.-Ing. degrees in computer science from the University of Bremen, Germany, in 2006 and 2009, respectively. Since then, he worked at the University of Bremen, the German Research Center for Artificial Intelligence (DFKI), the University of Applied Science of Bremen, the University of Potsdam, and the Technical University Dresden. Since 2015, he is working in Linz. His research interests are in the design of circuits and systems for both conventional and emerging technologies. In these areas, he published more than 250 papers in journals and conferences and served in editorial boards and program committees of numerous journals/conferences such as TCAD, ASP-DAC, DAC, DATE, and ICCAD. For his research, he was awarded, e.g., with a Best Paper Award at ICCAD, a Google Research Award, and more.

Junin-Dar Huang (M'96) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1992 and 1998, respectively. He is currently a Professor and the Curriculum Committee Chair in the Department of Electronics Engineering, National Chiao Tung University, Taiwan. He received an Outstanding Teaching Award of the University in 2014.

His research interests include behavioral and logic synthesis, design automation for biochips, and synthesis for single-electron transistor arrays. He received the Best Paper Award in IEEE Computer Society Annual Symposium on VLSI 2011. He was the Technical Program Committee Chair of SASIMI 2015. Dr. Huang is a member of the IEEE, ACM, IEICE, and Phi Tau Phi.

Bhargab B. Bhattacharya (F'07) is currently Distinguished Visiting Professor of Computer Science and Engineering at the Indian Institute of Technology Kharagpur, India. Prior to that he had been on the faculty of Indian Statistical Institute, Kolkata, India, for more than 35 years. He received the B.Sc. degree in physics from the Presidency College, Kolkata in 1971, the B.Tech. and M.Tech. degrees in radiophysics and electronics in 1974 and 1976, respectively, and the Ph.D. degree in computer science in 1986, all from the University of Calcutta. He held visiting professorship at the University of Nebraska-Lincoln, and at Duke University, USA, at the University of Potsdam, Germany, at the Kyushu Institute of Technology, Iizuka, Japan, at Tsinghua University, Beijing, China, and at IIT Guwahati, India. His current research interest includes design and test of microfluidic chips and integrated circuits. He has published more than 400 technical articles, and he holds 10 United States Patents. He is a Fellow of the Indian National Academy of Engineering and a Fellow of the National Academy of Sciences (India). He is on the editorial board of the *Journal of Electronic Testing: Theory and Applications* (JETTA).