Storage-Aware Sample Preparation Using Flow-Based Microfluidic Labs-on-Chip

Sukanta Bhattacharjee*, Robert Wille†, Juinn-Dar Huang‡, and Bhargab B. Bhattacharya§
*Center for Cyber Security, New York University, Abu Dhabi, UAE, E-mail: sb5638@nyu.edu
†Institute for Integrated Circuits, Johannes Kepler University Linz, 4040 Linz, Austria, E-mail: robert.wille@jku.at
‡Department of Electronics Engineering, National Chiao Tung University, Taiwan, E-mail: jdhuang@mail.nctu.edu.tw
§Nanotechnology Research Triangle, Indian Statistical Institute, Kolkata 700108, India, E-mail: bhargab@isical.ac.in

***Abstract*—Recent advances in microfluidics have been the major driving force behind the ubiquity of Labs-on-Chip (LoC) in biochemical protocol automation. The preparation of dilutions and mixtures of fluids is a basic step in sample preparation for which several algorithms and chip-architectures are well known. Dilution and mixing are implemented on biochips through a sequence of basic fluid-mixing and splitting operations performed in certain ratios. These steps are abstracted using a mixing graph. During this process, on-chip storage-units are needed to store intermediate fluids to be used later in the sequence. This allows to optimize the reactant-costs, to reduce the sample-preparation time, and/or to achieve the desired ratio. However, the number of storage-units is usually limited in given LoC architectures. Since this restriction is not considered by existing methods for sample preparation, the results that are obtained are often found to be useless (in the case when more storage-units are required 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). In this paper, we present a storage-aware algorithm for sample preparation with flow-based LoCs which addresses these issues. We present a SAT-based approach to construct a mixing graph that enables the best usage of available storage-units while optimizing sample-preparation cost and/or time. Experimental results on several test cases reveal the scope, effectiveness, and the flexibility of the proposed method.**

I. Introduction

Advances in microfluidic technologies revolutionize laboratory-based diagnostic procedures by offering portable Lab-on-a-Chip (LoC) devices [1]. A microfluidic biochip or LoC is a highly integrated analogue of bulky biochemical analyzers offering a low-cost automated platform for point-of-care diagnosis [2], sample preparation [3–7], DNA analysis [8], and drug discovery [9]. Two major classes of LoCs are commonly used in practice: digital microfluidic biochips (DMFBs) and continuous-flow microfluidic biochips (CFMBs). The latter class enjoys wide acceptance in the chemical engineering community [2, 8, 10] for historical reason. Modern CFMBs are usually equipped with pressure-driven micro-valves that allow for controlling the fluid flow through a network of micro-channels [11–13].

More precisely, a CFMB typically 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. 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 [10–12] e.g. as sketched in Fig. 1(b). Here, a mixing operation is conducted by opening the valves so that fluids are pushed into the ring-channels in the middle of the architecture. Afterwards, the pumping valves are actuated in a peristaltic sequence at a high frequency which, eventually, mixes the inserted fluids.

Microfluidic biochips have been considered as one of the key technologies for automatic sample preparation that includes dilution and mixing of fluids in certain ratios. Several approaches have been proposed for sample preparation on LoCs (see e.g. [3–7, 14]) considering different optimization objectives such as minimization of the (1) number of mixing operations (i.e., time), (2) consumption of valuable reagents, and (3) amount of waste generation.

Most of the existing LoC sample preparation methods indeed consider these three optimization objectives but they are oblivious of a crucial fact: the number of storage-units which are available on a CFMB is usually limited. A storage-unit is needed when an intermediate fluid-mixture needs to be stored for subsequent use. Unfortunately, maintaining an on-chip storage module in a CFMB is significantly expensive [14], since fluidic multiplexers (MUXs) [12] are required for this purpose. They do not only increase the area of the chip and require additional channel connections, but also lead to a more complicated control layer. As a consequence, corresponding sample preparation methods need to consider this restriction. Moreover, in cases where sample preparation is possible without any intermediate storage-unit, the algorithm should exploit the available storage-units at the platform in order to further reduce the consumption of reagents.

In this paper, we propose a storage-aware sample preparation method for mixing two or more biochemical reagents on a CFMB which, for the first time, addresses these issues. To this end, a SAT-based approach is proposed which allows to efficiently check several options of generating the desired target ratio and, eventually, choosing the one which makes the best usuage of the available storage-units while, at the same time, optimizing sample preparation costs and/or time. Implicitly, the proposed method also guarantees that no solution is chosen which requires more storage-units than available for the given platform, and flags when a solution does not exist.

The organization of the remainder of this paper is as follows. Section II briefly descibes the preliminaries of sample preparation, reviews existing sample preparation algorithms on CFMBs, and motivates this work. Afterwards, a detailed description of the proposed method for dilution and mixing is presented in Section III. Experimental results are reported in Section IV. Finally, the paper is concluded in Section V.

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.

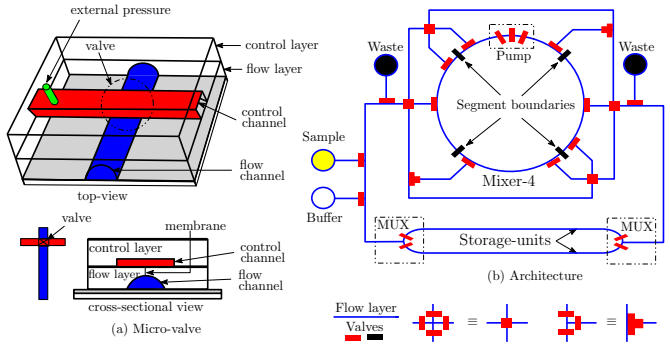


Fig. 1: Schematic of a two-layer microfluidic device: (a) top and cross-sectional view (b) CFMB platform

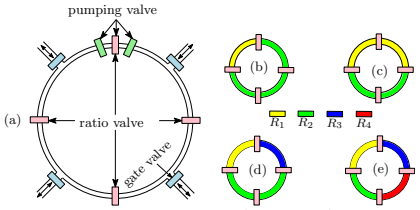


Fig. 2: (a) A 4-segment rotary mixer (Mixer-4) and possible mixing models (b) (1:3) (c) (1:1) (d) (1:2:1) (e) (1:1:1:1)

II. Background and Motivation

This section briefly reviews the task of sample preparation using CFMB 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.

A. Background: Sample Preparation and State of the Art

Sample preparation is the process of mixing two or more biochemical fluidic reagents in a given volumetric ratio through a sequence of mixing operations[1]. Given a target ratio, sample preparation methods represent the desired ratio with respect to a mixing model supported by the microfluidic platform (here: CFMB) and a user-defined tolerance $0 \leq \epsilon < 1$ for the target ratio.

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. 2(a). The ratio valves allow to fill each segment with a different fluid as exemplarily illustrated in Figs. 2(b)-(e). Afterwards, those fluidics 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. Applying such mixing steps in a sequence (usually represented by a so-called mixing graph) allows to mix m input reagents so that eventually the desired ratio results. Beforehand, a ratio of m input reagents $\{R_1 : R_2 : \cdots : R_m = x_1 : x_2 : \cdots : x_m\}$, where $\sum_{i=1}^{m} x_i = 1$, must be represented as reachable mixing ratio $\{R_1 : R_2 : \cdots : R_m = y_1 : y_2 : \cdots : y_m\}$, where $\sum_{i=1}^{m} y_i = N^d, d \in \mathbb{N}$, on the CFMB platform supporting multiple mixing model which are possible with a Mixer-N. Fig. 3 summarizes the main steps of this process.

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 depth of the mixing tree is determined by d. A detailed description of ratio transformation in sample preparation can be found in [7]. For example, a target ratio $\{\text{sample} : \text{buffer} = 0.489 : 0.511\}$ (note that, $0.489 + 0.511 = 1$) can be represented as $\{\text{sample} : \text{buffer} = 125 : 131\}$ ($125 + 131 = 4^4$) for Mixer-4 (i.e., $N = 4$) and user-defined $\epsilon = 0.001$. 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. In fact, $d = 4$ gives the smallest number of mixing operations which satisfies the given error-tolerance.

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

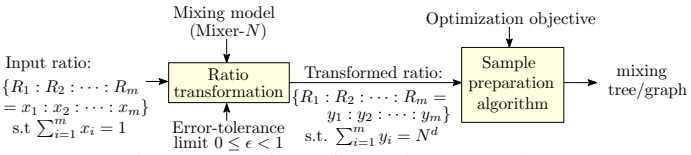


Fig. 3: Overview of sample preparation

In the recent past, Liu *et al.* proposed a tree pruning and grafting method (called *TPG* [6]) that starts from an initial mixing tree (based on a (1:1) mixing model) and transforms it for obtaining a dilution graph for unequally segmented rotary mixer (Ring-N). In [5], a volume-oriented sample preparation algorithm (called *VOSPA*) has been introduced which employs a greedy strategy. Lei *et al.* [15] 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 [7] 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 [7]	2	No	No*
TPG [6]	2	No	No$^\diamond$
VOSPA [5]	2	No	No$^\diamond$
Flow-based [15]	2	Yes†	No$^\diamond$
FloSPA [7]	≥ 2	Yes	No$^\diamond$
Proposed	≥ 2	Yes	Yes

* Does not utilize any storage-unit at all (and, hence, yields rather expensive solutions when storage-units are available).
$^\diamond$ 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 1. *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 [5] and FloSPA [7], require four and five storage-units as shown in Fig. 4(a) and Fig. 4(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. 4(c) can be determined using the NWayMix [7] 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. 4(c)) – a very expensive solution. In contrast, the desired mixing ratio can*

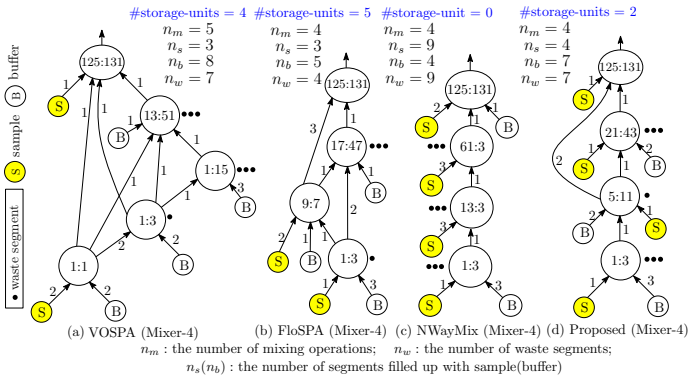


Fig. 4: Dilution graph generated with (a) NWayMix, (b) VOSPA, (c) FloSPA, and (d) the proposed method for $\{\text{sample} : \text{buffer} = 125 : 131\}$

be realized more efficiently as shown in Fig. 4(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.

The above-mentioned example motivates 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. Storage-Aware Sample Preparation

In this section, the proposed method is described in detail. 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 [5, 7]. 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 [16, 17], which already have been found effective for similar tasks in the design of LoCs (see e.g. [18, 19]). 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, we consider dilution problems only, i.e., the case where only two fluids (a sample and a buffer) are mixed. Here, in fact, every ratio can be realized with zero storage-unit (although the storage-units which are available anyway can be utilized to reduce the costs of the sample preparation). Afterwards, the general case of mixing is covered, i.e. the case where more than two fluids are mixed. This requires stricter constraints to be satisfied and, hence, is described in a separate subsection. The main steps of both flows for dilution and mixing are summarized in Fig. 5 and Fig. 6, respectively.

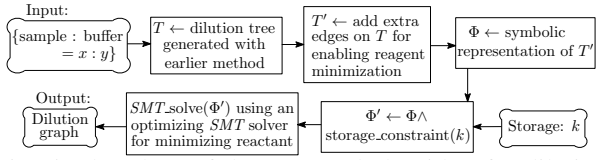


Fig. 5: Flowchart of the proposed algorithm for dilution

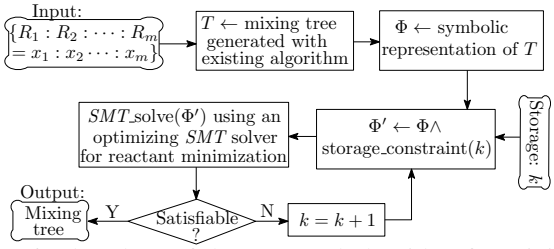


Fig. 6: Flowchart of the proposed algorithm for mixing

A. Proposed Method for 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 [7] 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. 7(a) sketches the resulting graph.

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. 7(b). The general structure of such transformation is shown in Fig. 7(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. To this end, we introduce the following free variables:

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.

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

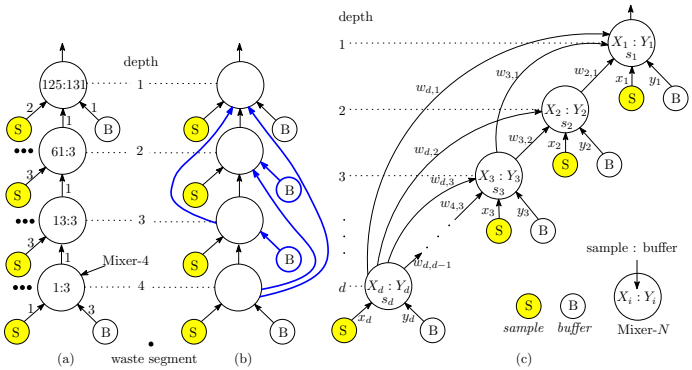


Fig. 7: (a) Dilution tree generated by NWayMix for target ratio {sample : buffer = 125 : 131} (b) dilution graph after adding extra edges and reagent nodes for enabling reagent minimization, (c) general structure of dilution graph

The annotations in Fig. 7(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. To this end, the following constraints are introduced for each mixing node at depth i of the mixing graph:

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

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

Note that each mixing node at depth i can fill its segments with sample, buffer, or any unused fluid segments produced at depth $j > i$. Hence, the desired ratio of sample and buffer at depth i i.e., $\{X_i : Y_i\}$ is determined with these equations. Furthermore, the non-linearity of above equations can be removed easily by adding few extra constraints as carried out in [7]. This transformation helps to run powerful sound and complete SMT-solvers [16, 17] 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, whereas, Mixer-N can serve at most N segments to other mixers. 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.

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. Recall that, for each depth, a storage variable s_i is available which denotes the number of on-chip storage-units required for executing the portion of induced subgraph containing mixing nodes at depth j, where $i \leq j \leq d$. This amount is determined by the following equation:

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

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. 8 shows the induced subgraph of the general mixing graph from Fig. 7(b) used in the storage calculation. Note that no storage-unit is required for subgraph containing mixing node at depth d only. Hence $s_d = 0$. However for computing storage requirement of the induced subgraph given in Fig. 8, 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, $s_i = s_{i+1} + w_{d,i} + w_{d-1,i} + \cdots + w_{i+2,i}$. It can be easily verified that 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 [16, 17] that finds the reagent minimal solution on the general dilution tree (fig. 7(b)) that gives minimal reagent solution satisfying storage constraint.

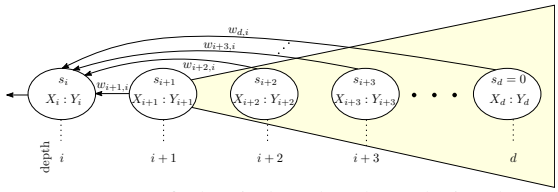


Fig. 8: Structure of the induced subgraph in the general dilution tree, containing mixing nodes at depth $i, i+1, \cdots, d$, used for storage computation

Example 2. *Fig. 9 shows solutions realizing the target ratio {sample : buffer = 125 : 131} with the least reactant-cost based on the graph of Fig. 7(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, ⋯ , 5).*

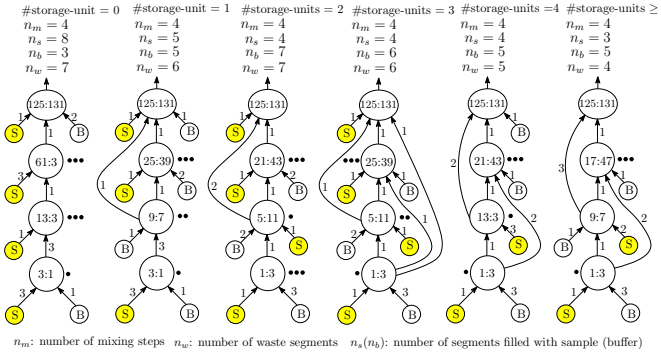


Fig. 9: Reagent minimal solution for the target ratio {sample : buffer = 125 : 131} considering the availability of different number of available on-chip storage-units on the CFMB architecture given in Fig. 1(b)

B. Proposed Method for Mixing

The proposed method for mixing basically follows the same idea as the method for dilution. However, in contrast to dilution, it is not always possible to perform mixing of three or more reagents with zero on-chip storage-unit. This is due to the inherent tree structure of the mixing graph which commonly appears in mixture preparation. As before, we start with a basis mixing graph produced by previously proposed methods (here, e.g., genMixing [7]) and introduce additional variables and constraints to provide a symbolic representation of all possible options out of which the best one satisfying the storage limitation is determined by the solving engine. To this end, we particularly have to adjust the storage constraints[3].

Enforcing the Available Number of Storage-Units

Fig. 10 sketches a generic node in a mixing graph as well as a simplified notation of the variables that are used in storage calculation. On a mixing tree, a mixing node may take a segment of fluid from one of its subtrees or it may take input reagents to fill one of its N segments. Hence, there can be at most N subtrees possible for a mixing node. In Fig. 10, 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 again the storage requirement of subtree-i. Then, the number of storage-units s can be determined with the following equation:

$$s = \bigvee_{j=1}^{N} \left(\sum_{i=1}^{j-1} w_i + s_j + \sum_{i=j+1}^{N} w_i \right) \tag{6}$$

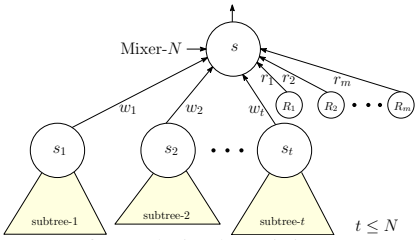


Fig. 10: Structure of a node in the mixing tree used in storage computation

Note that Equation 6 symbolically represents the 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 the number k of available storage-units will eventually only allow solutions which can be realized under this restriction.

Example 3. *Figs. 11(a)-(c) show the mixing graphs for the target ratio* $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ *obtained by genMixing [7], FloSPA [7], and by the proposed method, respectively. Note that Fig. 11(a) requires two storage-units. Fig. 11(b) shows a cheaper solution obtained by FloSPA using two storage-units. On the other hand, the proposed method ensures that no solution exists with zero storage-unit starting from Fig. 11(a); it also provides a solution with one storage-unit (Fig. 11(b)), and with two storage-units (Fig. 11(c)).*

[3]Note that, due to page limitations, we are not repeating the definition of the variables and constraints enforcing the target ratios. They are basically identical to the ones used for dilution.

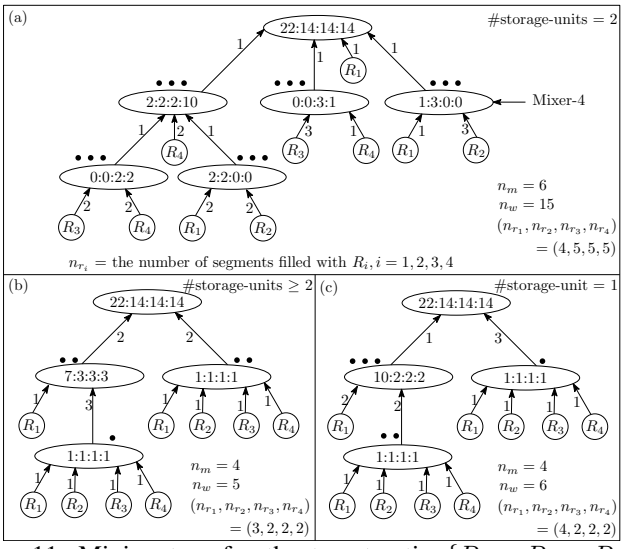


Fig. 11: Mixing tree for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$ generated by (a) genMixing [7], (b) FloSPA [7] and by the proposed method for #storage-units $\geq$ 2 , and (c) by the proposed method when #storage-unit = 1

IV. Experimental results

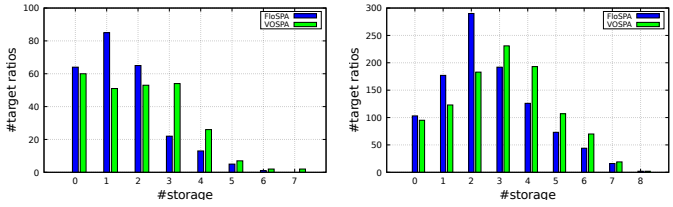
The methods described above are compared with several state-of-the-art sample-preparation methods, namely NWayMix [7] and VOSPA [5] for dilution as well as with FloSPA [7] for dilution and general mixing. In the following, the obtained results for both cases are summarized[4].

A. Performance for Dilution

VOSPA [5] and FloSPA [7] 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. 12. Here, we have generated dilution graphs for all possible target ratios of sample and buffer by varying $d = 4, 5$ for $N = 4$ and listed for how many ratios a particular number of storage-units is required. 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 considered platform, the obtained result is useless. 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 costs, significant improvements can be observed as shown in Table II. Here, we have generated dilution graphs for all possible target ratios in $\{\text{sample} : \text{buffer} = x : 256 - x\}$, where $1 \leq x \leq 255$, i.e., $N = 4, d = 4$ and listed the average number of mixing steps ($\bar{n}_m$), waste segments ($\bar{n}_w$), and number of segments filled with sample ($\bar{n}_s$) and buffer ($\bar{n}_b$) for a different number of available on-chip storage-units (k) in case of the proposed approach. For the previously proposed approaches, we list the best results, i.e. obtained with zero storage-unit in case of NWayMix, obtained with seven storage-units in case of VOSPA, and obtained with six storage-units in case of FloSPA.

[4]Note that, due to page limitation, we do not report details on the run-time performance. However, the run-time of the proposed approach is similar to the run-time of FloSPA [7].



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

Fig. 12: Histogram of the number of storage-units requirement of VOSPA and FloSPA for all target ratio belong 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., $N = 4, d = 5$

TABLE II: Performance of proposed dilution algorithm

Proposed approach					NWayMix [7]	VOSPA [5]	FloSPA [7]
k	$\bar{n}_m$	$\bar{n}_w$	$\bar{n}_s$	$\bar{n}_b$	#storage-unit = 0	#storage-units = 7	#storage-units = 6
0	3.68	5.39	4.22	5.18	$\bar{n}_m = 3.68$	$\bar{n}_m = 4.13$	$\bar{n}_m = 3.68$
1	3.68	4.19	3.51	4.67	$\bar{n}_w = 8.04$	$\bar{n}_w = 6.70$	$\bar{n}_w = 3.66$
2	3.68	3.80	3.40	4.40	$\bar{n}_s = 6.02$	$\bar{n}_s = 3.34$	$\bar{n}_s = 3.36$
3	3.68	3.71	3.38	4.32	$\bar{n}_b = 6.02$	$\bar{n}_b = 7.36$	$\bar{n}_b = 4.30$
≥ 4	3.68	3.66	3.36	4.30			

Several issues can be observed here: First, NWayMix always uses zero storage-unit. 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 improve e.g. the number of input reagent-units. 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 several actual mixing ratios and the number of on-chip storage-units (k) as summarized in Table III. Here, we list the considered mixing ratios as well as the number of mixing steps (n_m), waste segments (n_w), and the total number of segments filled with input reagents (n_r). It can be observed that the proposed method 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 same or fewer number of on-chip storage-units compared to genMixing but also produces the target ratio with a samller number of mixing steps, waste, and input segments.

TABLE III: Performance of the proposed mixing algorithm

#	mixing ratio	genMixing [7]				Proposed method		
		n_m	n_w	n_r	k	$k = 0$	$k = 1$	$k \geq 2$
						(n_m, n_w, n_r)	(n_m, n_w, n_r)	(n_m, n_w, n_r)
1	90:90:76	5	12	16	1	(4, 8, 12)	(5, 6, 10)	(5, 6, 10)
2	20:14:16:14	4	9	13	1	no sol.	(4, 6, 10)	(4, 5, 9)
3	27:7:13:17	5	12	16	1	no sol.	(4, 8, 12)	(4, 8, 12)
4	68:86:56:46	6	15	19	1	no sol.	(5, 9, 13)	(6, 9, 13)
5	27:25:57:69:78	8	21	25	2	no sol.	(7, 15, 19)	(7, 13, 17)
6	30:24:55:68:79	8	21	25	2	no sol.	(6, 13, 17)	(6, 12, 16)
7	30:24:155:38:9	8	21	25	2	no sol.	(7, 15, 19)	(7, 13, 17)
8	300:499:225	7	18	22	1	(5, 4, 9)	(5, 4, 9)	(5, 4, 9)
9	57:28:6:6:6:3:150	9	24	28	2	no sol.	(6, 10, 14)	(6, 10, 14)
10	102:26:3:3:122	8	21	25	2	no sol.	no sol.	(7, 12, 16)
11	4:6:10:14:22:26:174	10	27	31	3	no sol.	no sol.	(7, 14, 18)
12	26:15:51:26:5:5:1:127	12	33	37	3	no sol.	no sol.	(10, 24, 28)

parameter values corresponding to FloSPA [7] are highlighted with yellow color

Note that mixture preparation using FloSPA reduces the input reagent consumption without considering the limited availability of on-chip storage-units. For example, in case of Mixture 3 and Mixture 4 shown in Table III, the mixing graph generated with FloSPA may require one/two on-chip storage-units as both solutions (for $k = 1$ and $k \geq 2$) fill the same number of segments with input reagents, i.e., both solutions meet the optimization objective of minimum reagent-usage. Moreover, for Mixtures 1-7 in Table III, the proposed method can produce the same mixture with a smaller number of storage-units by consuming little more input reagents compared to FloSPA.

V. Conclusions

In this work, we proposed a storage-aware sample-preparation method for continuous-flow microfluidic biochips. To this end, we augmented mixing graphs determined by previously proposed sample-preparation methods eventually 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 storages than available and, hence, are useless). Moreover, the proposed approach explicitly allows to fully exploit the available number of storages, e.g., in order to reduce the use of reagents.

References

[1] 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/

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] C.-H. Liu, T.-W. Chiang, and J.-D. Huang, "Reactant minimization in sample preparation on digital microfluidic biochips," *IEEE Trans. on CAD*, vol. 34, no. 9, pp. 1429–1440, 2015.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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].

[17] N. Bjørner, A.-D. Phan, and L. Fleckenstein, "*vZ* - an optimizing SMT solver," in *Proc. of TACAS*, 2015, pp. 194–199.

[18] O. Keszocze, R. Wille, T.-Y. Ho, and R. Drechsler, "Exact one-pass synthesis of digital microfluidic biochips," in *Proc. of DAC*, 2014.

[19] 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.