2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)

2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID) | 979-8-3503-8440-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/VLSID60093.2024.00073

Accelerating Fluid Loading in Sample Preparation with Fully Programmable Valve Arrays

Mohit Kumar*, Abhik Kumar Khan*, Sudip Roy†, Krishnendu Chakrabarty‡, Sukanta Bhattacharjee*
*Indian Institute of Technology (IIT) Guwahati, India, †IIT Roorkee, India, ‡Arizona State University, USA

***Abstract*—Microfluidic fully programmable valve array (FPVA) biochips emerge as programmable flow-based labs-on-a-chip consisting of a two-dimensional array of reaction chambers surrounded by four microvalves. Given a mixing tree (represents a sequence of mixing steps) for the target ratio, existing design automation algorithms (DAA) for sample preparation (mixing of reagents in a desired ratio through a sequence of mixing) on FPVA first find the suitable transport-free placement (avoids cumbersome transportation of fluid segment(s) between chambers) and scheduling of mixing operations. After that, reagents are loaded into designated mixer chambers by pushing fluids through input-to-output paths. Since the existing DAA for sample preparation did not consider fluid loading while finding the placement of mixing operations, it often requires a longer and more complex fluid loading path, increasing the sample preparation time and cost. In this paper, we leverage the combinatorial properties of the mixing tree to transform it in favor of finding simpler and shorter paths to load fluids. The simulation results reveal a considerable speedup in the fluid loading time and savings in reagent usage when the proposed algorithm is applied to the mixing tree.**

I. Introduction

Microfluidic biochip or lab-on-a-chip (LoC) is a miniaturized biochemical laboratory that can manipulate the nano/pico-liter volume of fluids in an automated fashion [1]. These lab-on-a-chip microsystems offer several advantages over conventional biochemical analyzers, such as faster biochemical reaction, ultra-sensitive detection, and lower cost per assay. These chips have already made a profound impact on point-of-care (POC) medical diagnostics [2], DNA amplification [3, 4], and biomedical research [5].

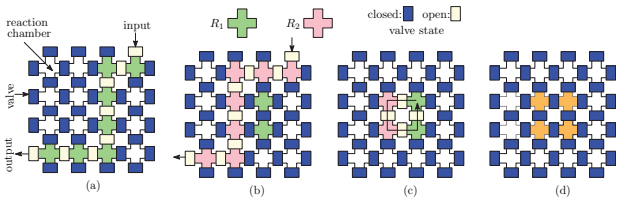


Fig. 1: Fully programmable valve array (FPVA): (a) An input-to-output path is created by configuring valves before loading reagents R_1 and (b) R_2. (c) Mixing operation of R_1 and R_2 in $(1:1)$ ratio using a 2×2 mixer. (d) Storage of fluids after mixing.

Flow-based microfluidic biochips manipulate fluids in a network of microchannels by actuating pressure-driven microvalves. These chips are designed for application-specific purposes, therefore, offer less programmability. Fully programmable valve arrays (FPVAs) have emerged as a general-purpose flow-based microfluidic platform [6]. An FPVA is a two-dimensional array of fluid chambers surrounded by four microvalves, as shown in Fig. 1. Various fluidic operations such as loading a fluid into a specific chamber (Fig. 1(a-b)), mixing (Fig. 1(c)), and storage (Fig. 1(d)) operations can be performed in a re-configurable fashion by setting the valve state suitably. Several biochemical assays, such as cell culture and surface immunoassay, have been demonstrated successfully on FPVAs [6].

Sample preparation is an essential step of almost any biochemical protocol where two or more input reagents are mixed into a desired ratio. LoC-based sample preparation methods deploy a sequence of mixing operations (represented as a mixing tree [7]) to prepare a target ratio of input reagents. This process requires mixing of fluids (input reagents and intermediate fluids generated during the process) in different volumetric proportions (depending on the mixing ratios supported by the underlying LoC platform), storage of intermediate fluids for subsequent mixing, and transportation of fluids between storage and mixer. Although FPVA can store fluids in a chamber and perform active mixing operations in a reconfigurable manner (by grouping cells in a closed loop), transportation of fluids between chambers is cumbersome.

Existing synthesis algorithms for sample preparation require the transportation of fluids between chambers; therefore, they do not apply to the FPVA. To overcome this drawback, an algorithm (a.k.a. no transport mixing (NTM) [8]) was proposed that takes the mixing tree as an input and finds suitable placements of mixing operations obviating any movement of fluids between chambers. Note that NTM did not consider the loading of fluids while finding placements of mixing operations. Therefore, we must load input fluids to the designated chambers before mixing. To load an input fluid to a set of chambers, we need to create an input-to-output path by changing the state of the valves on the path. Due to the limited number of external control lines on FPVA, we must change the valve state sequentially. Therefore, the fluid loading time is proportional to path length. Moreover, the amount of input reagnets i.e., the cost of sample preparation is proportional to the number of flows and the complexity of a flow path (the number of $90°$ bends). Since the flow pressure drops rapidly with an increase in the number of $90°$ bends in a flow path [9], the reagent usage in each fluid loading cycle increases.

In this work, we propose an algorithm to reduce the path length and complexity for loading an input fluid to chambers,

979-8-3503-8440-6/24/$31.00 ©2024 IEEE
DOI 10.1109/VLSID60093.2024.00073

thereby reducing the sample preparation time and cost while increasing reliability of the underlying FPVA platform by actuating fewer valves in each loading operation. The proposed algorithm permutes leaf nodes appearing at the same level of the mixing tree to bring the same reagent together. Note that the combinatorial properties of the mixing tree generated by the genMixing [10] algorithm (ref. Theorem 1) allow us to permute leaf nodes without changing the target ratio. We have also performed an extensive simulation by taking several target ratios varying the number of input reagents. The simulation results show that the proposed algorithm reduces the path length and complexity considerably while loading input reagents.

The rest of the paper is organized as follows. Section II provides the necessary backgrounds of the design automation of the sample preparation on an FPVA. Section III describes the proposed method to minimize the fluid loading path length. Simulation results are presented in Section IV. Finally, Section V concludes the paper.

II. Background

A. Fully Programmable Valve Array (FPVA) Biochip

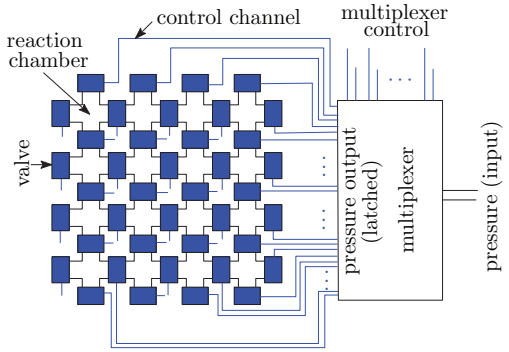


Fig. 2: Schematic diagram of FPVA biochip.

An FPVA consists of a two-dimensional array of fluid chambers surrounded by four microvalves. A microfluidic multiplexer is used to control valves, which reduces the number of external control lines. Since valves surrounding fluid cells are actuated through the multiplexer output, those valves are actuated sequentially through multiplexer control lines. Fig. 2 shows a schematic diagram of an FPVA biochip with 16 fluid chambers. Approximately 100 ms is required to actuate a valve through multiplexer control [6]. Therefore, the time to create a fluidic path depends on the number of valves that must be actuated to create the path.

B. Basics of Sample Preparation

Sample preparation is a process of mixing two or more input reagents in a given volumetric ratio. Given a target ratio of k input fluids $\mathcal{M} = \{R_1 : R_2 : \cdots : R_k = x_1 : x_2 : \cdots : x_k\}$, where $0 \leq x_i \leq 1$ and $\sum_{i=1}^{k} x_i = 1$, we represent the desired ratio for the type of mixer supported by the microfluidic platform (here: FPVA) and a user-defined tolerance $0 \leq \epsilon < 1$. Since on an FPVA we use 2×2 mixers[1] for mixing fluids, the target ratio $\mathcal{M}$ is transformed to another mixing ratio $\{R_1 : R_2 : \cdots : R_k = y_1 : y_2 : \cdots : y_k\}$, where $0 \leq y_i \leq N^d$ and $\sum_{i=1}^{k} y_i = N^d, d \in \mathbb{N}$. Note that, d is chosen depending on the user-defined error tolerance limit $0 \leq \epsilon < 1$ satisfying $\max_i\{|x_i - \frac{y_i}{N^d}|\} < \epsilon$.

For the transformed ratio $\{y_1 : y_2 : \cdots : y_k\}$, the sample-preparation algorithm genMixing [10] represents each y_i as d-digit base-4 number i.e., $(y_i)_{10} = (a^i_{d-1}a^i_{d-2}\cdots a^i_1 a^i_0)_4$. Next, these k d-digits numbers are scanned from left-to-right to construct a mixing tree in bottom-up fashion. The depth of the mixing tree is determined by d. Corresponding to each non-zero digit a^i_j in the base-4 representation of y_i, a^i_j units of input reagent R_i are fed as input to the mixer. An internal (leaf) node in the mixing tree represents a mixing operation (input reagent), and an edge represents a unit volume of fluid shared between the nodes.

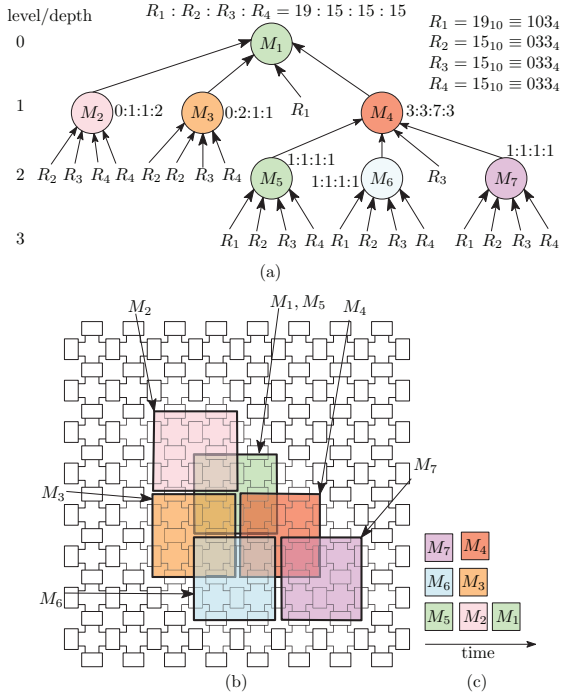


Fig. 3: (a) Mixing tree for the target ratio $R_1 : R_2 : R_3 : R4 = 19 : 15 : 15 : 15$ generated by the genMixing [10] algorithm. (b) Transport free placements of the mixing operations obtained by the NTM algorithm [8]. (c) Gnatt chart for scheduling of mixing operations.

Example 1. *Consider the sample preparation for the target mixing ratio* $\{R_1 : R_2 : R_3 : R_4 = 0.30 : 0.23 : 0.23 : 0.24\}$ *on an FPVA biochip (use* 2×2 *mixer) for the given error tolerance* $\epsilon = 0.007$. *The desired ratio can be represented as* $\{R_1 : R_2 : R_3 : R_4 = 19 : 15 : 15 : 15\}$ *since* $\max\{|0.4 - \frac{19}{4^3}|, |0.23 - \frac{15}{4^3}|, |0.24 - \frac{15}{4^3}|\} = 0.005 < \epsilon$. *The sample preparation algorithms scans the base-4 representation*

[1] a 2×2 mixer can mix fluids in (1 : 1), (2 : 1 : 1), (3 : 1), and (1 : 1 : 1 : 1) ratios

of the digits of $19 : 15 : 15 : 15$ *(*$19_{10} \equiv 103_4$ *and* $15_{10} \equiv 033_4$*) right-to-left to determine the number of reagents used. Fig. 3(a) shows the mixing tree for the target ratio* $19 : 15 : 15 : 15$*. Note that the rightmost digits (3333) of the base-4 representation of the target ratio denote three units of all reagents used in depth-3. Similarly, three units of* R_2, R_3*, and* R_4 *are used in depth-2 (0333), and one unit of* R_1 *is used in depth 1 (1000).*

C. Design Automation of Sample Preparation on FPVA

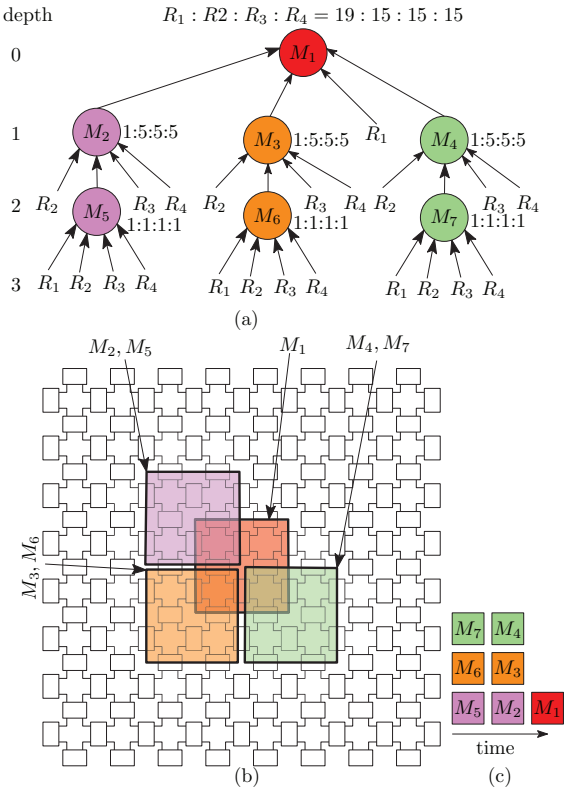


Fig. 4: (a) Mixing tree for the target ratio $R_1 : R_2 : R_3 : R_4 = 19 : 15 : 15 : 15$ generated by the genMixing is transformed by the HDA algorithm [8]. (b) Transport-free placements of the mixing operations obtained by the NTM algorithm [8]. (c) Gnatt chart for scheduling of mixing operations.

Since the movement of fluids between two chambers is infeasible in FPVA, it is challenging to place mixers to avoid the transportation of fluids between two chambers. Choudhary et al. proposed an algorithm (a.k.a. No Transport Mixing (NTM)) to find placement and scheduling of mixing operations in a mixing tree using 2×2 mixers without requiring any transportation of fluid(s) between chambers [8]. Fig. 3(b) and Fig. 3(c) show the transport-free placement and scheduling of mixing operations in the mixing tree (Fig. 3(a)) generated by the genMixing algorithm for the target ratio $R_1 : R_2 : R_3 : R_4 = 19 : 15 : 15 : 15$ using a 2×2 mixing module with NTM, respectively.

Mixing tree obtained by the genMixing algorithm is transformed by the Heuristic for Distribution (HDA) algorithm to reduce the total number of chambers (footprint) used for mixer placement. Fig. 4(a) shows the transformed mixing tree generated by the HDA starting from the mixing tree generated by genMixing (Fig. 3(a)). Fig. 4(b) and Fig. 4(c) show the transport-free placement and scheduling of mixing operations in the mixing tree shown in Fig. 4(a) using a 2×2 mixing module with NTM, respectively. Note that, the footprint of mixing operations is reduced with HDA+NTM compared to NTM. Recently Hirai et al. reported a method to schedule and place mixing operations in an input mixing tree using only 2×2 and 2×3 mixing modules that do not need transportation of fluids between chambers [11]. Note that, before starting a mixing operation, we need to load the required reagents into the mixing chambers. Kundu et al. proposed an algorithm that initially finds a loading-aware fluid-to-cell assignment (a.k.a LAFCA) to map input reagents to mixer chambers. After that, it finds input-to-output paths for loading input reagents by determining flow from the last (a.k.a DFL) algorithm. Note that a sequence of algorithms (a.k.a synthesis recipe or recipe in short) (genMixing+NTM+(LAFCA+DFL) or genMixing+HDA+NTM+(LAFCA+DFL)) is used to determine desired operations to realize the target mixing ratio on the target FPVA biochip. Fig. 5 shows the overview of the design automation steps for sample preparation on an FPVA.

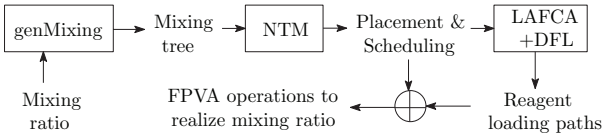


Fig. 5: Design automation of sample preparation on an FPVA: a sequence of algorithms (genMixing+NTM+(LAFCA+DFL)) transforms a mixing ratio to the FPVA actuation

III. Proposed Method

A. Motivation

Given a mixing tree, existing design automation algorithms for sample preparations [8, 11] find transport-free placement and scheduling of mixing operations on an FPVA. Since the methods proposed in [8, 11] did not consider the loading operations during sample preparation, we must solve the loading problem after placement. However, fluid loading after the mixer placement might require longer and complex flow paths which elongates the sample preparation time and cost. Let us consider a motivating example as follows.

Example 2. *Let us consider the loading of reagents for the mixing operations* M_5, M_6*, and* M_7 *that can be scheduled in parallel (ref. Figs. 4(a-c)). Placement of mixers and fluids to chamber assignment are shown in Fig. 4(b) and Fig. 6(d), respectively. Fluid loading paths are shown in Figs. 6(a-d) and 80 valve actuations are required to setup the paths.*

Note that we can permute the leaf nodes in each level of the mixing tree to maximize the number of chambers loaded with the same reagent. This transformation reduces the total path

length for fluid loading. Fig. 7(a) shows the mixing tree after permuting the leaf nodes at each level. Moreover, Figs. 7(b)-(e) shows the fluid loading paths. The total path length (or the number of valve actuation) and path complexity (the number of 90° bends) are reduced from 80 to 58 (27.5%) and 26 to 20 (23%) when the leaves in the mixing tree are permuted.

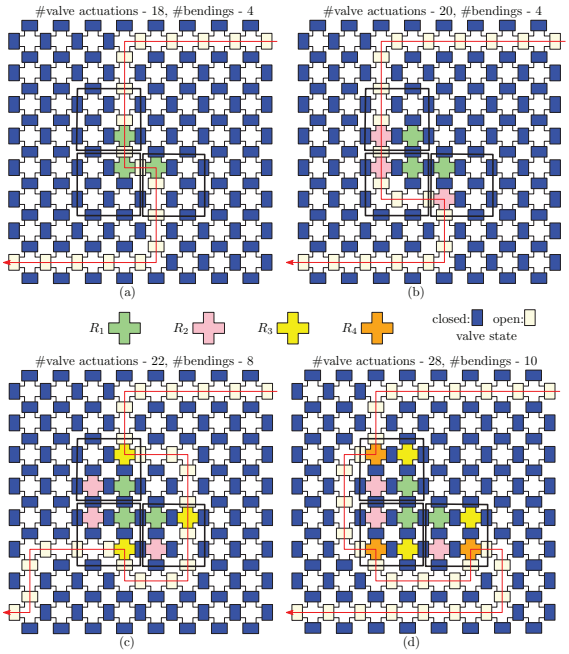


Fig. 6: Loading of reagent (a) R_1, (b) R_2, (c) R_3, and (d) R_4 and fluid(s) to chamber(s) assignment.

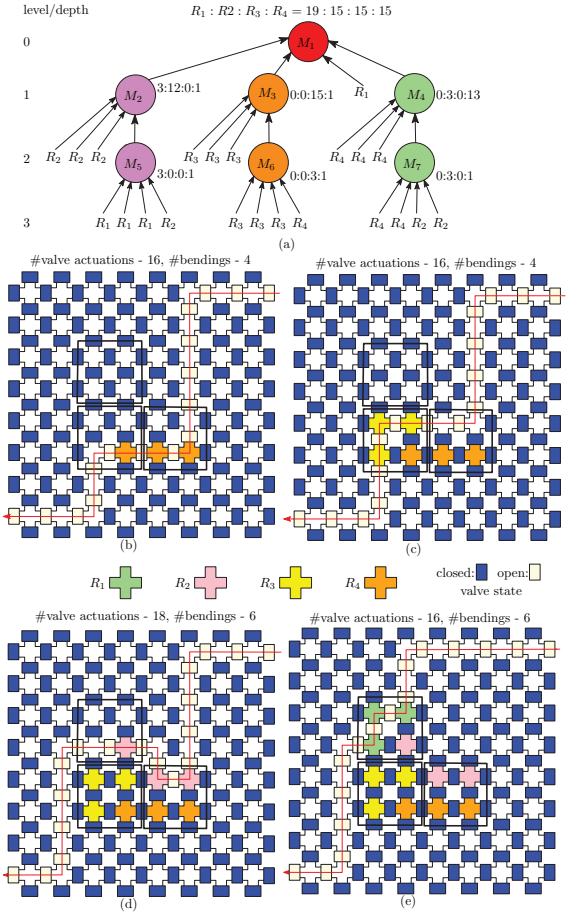


Fig. 7: (a) Mixing tree after permuting the leaf nodes. Loading of reagents (a) R_4, (b) R_3, (c) R_2, and (d) R_4.

B. Loading-aware Transformation of Mixing Tree

The mixing process for generating a target ratio can be represented as a mixing tree. As shown in Fig. 3(a), each leaf node of a mixing tree represents an input reagent, while each internal node represents the mixture resulting from the combination of its children. The mixing ratio of an internal node can be calculated by taking the arithmetic mean of the concentration factor (CF) of each reagent. The CF of a reagent R_i in the mixing ratio $R_1 : R_2 : \cdots : R_k = x_1 : x_2 : \cdots : x_k$ is defined as $\frac{x_i}{\sum_{j=1}^{k} x_j}$. Note that, an input reagent R_i can also be represented as $R_1 : R_2 : \cdots : R_k = x_1 : x_2 : \cdots : x_k$, where $x_i = 1$ and $x_j = 0$ for $i \neq j$.

Example 3. *In the mixing tree shown in Fig. 3(a)* R_1, R_2, R_3, *and* R_4 *can be represented as* $1:0:0:0, 0:1:0:1, 0:0:1:0$, *and* $0:0:0:1$ *respectively. The mixing ratio of* M_4 *can be calculated as* $R_1 : R_2 : R_3 : R_4 = \frac{1}{4}\big(3 \cdot \frac{1}{4} : 3 \cdot \frac{1}{4} : 1 + 3 \cdot \frac{1}{4} : 3 \cdot \frac{1}{4}\big) = 3 : 3 : 7 : 3$. *Similarly, the mixing ratio of* M_1 *can be calculated as* $R_1 : R_2 : R_3 : R_4 = \frac{1}{4}\big(1 + \frac{3}{16} : \frac{1}{4} + \frac{2}{4} + \frac{3}{16} : \frac{1}{4} + \frac{1}{4} + \frac{7}{16} : \frac{2}{4} + \frac{1}{4} + \frac{3}{16}\big) = 19 : 15 : 15 : 15$.

The following theorem describes the CF of an input reagent in the target mixture based on the depths of leaf nodes (i.e., input reagents) appeared in the mixing tree. The depth of a node n in a mixing tree is the length of the path from the root node to n.

Theorem 1. *Consider a mixing tree and an input reagent* R. *Let* n_i *denote the number of leaf nodes of the reagent* R *appearing at level* i *of the tree. Then the concentration factor (CF) of* R *contained in the root mixture is* $\sum_i \frac{n_i}{4^i}$.

Proof. The CF of an input reagent R used at level i of the mixing tree is divided (by 4) i times in the mixing process. Hence, the input reagent in depth i contributes 4^{-i} to the root mixture. Note that the CF of the input reagent is 1. Since each mixing operation sums the CFs from child nodes, the overall contribution is the sum across the leaf nodes at all depths, which is $\sum_i \frac{n_i}{4^i}$ □

The above theorem guarantees the following target ratio preserving transformation of a mixing tree. If we permute leaf nodes appearing in the same depth of the mixing tree, the target mixing ratio remains unchanged. Given a mixing tree, we permute leaf nodes appearing in the same depth to maximize the usage of same reagent in the mixing operations. Since the proposed transformation maximizes the number of

chambers loaded with the same fluid within a mixer, the length and complexity (the number of 90° turns) of fluid loading path decreases. We define the problem of permuting leaf nodes appearing in the same depth of the mixing tree as follows.

Problem statement: Consider a mixing tree having k input reagents $R_1, R_2, \cdots, R_k$. Suppose we represent the input reagents in depth d of the mixing tree as a multiset $R^d = \{R_i : r_i | r_i > 0 \text{ for } i = 1, 2, \cdots, k\}$, where r_i denotes the number reagent R_i used in depth d of the mixing tree. Let us denote mixing nodes appearing in the mixing tree at depth $d-1$ as $M_1^{d-1}, M_2^{d-1}, \cdots, M_l^{d-1}$ and the number of leaf nodes (reagent) used in each mixing node M_i^{d-1} be n_i for $i = 1, 2, \cdots, l$. Additionally, $\sum_{(R_i:r_i)\in R^d} r_i = \sum_{j=1}^{l} n_j$. The objective is to partition R^d into l multisets of size $n_1, n_2, \cdots, n_l$ respectively, so that the number of different input reagents appearing in each partition is minimized.

Algorithm 1: permuteLeaf(T, d)

```
   Input: T: mixing tree having k input reagents, d: depth
   Output: Mixing tree with permuted leaf nodes at depth d
1  Create list L^d = [(R_i, r_i) for each R_i : r_i ∈ R^d];
2  Create list M^{d-1} = [(M_i^{d-1}, n_i) for each mixing node M_i^{d-1}
       present in depth d-1 of T] /* n_i: number of
       reagents used in the mixing node M_i^{d-1}      */
3  Sort L^d in the descending order of r_i's;
4  for each (M, n) ∈ M^{d-1} do
5      while n ≥ 1 do
6          (R, r) = L^d[0];
7          if n ≥ r then
8              Use r leaf nodes of R in mixer M;
9              Remove L^d[0] from L^d;
10             n = n − r;
11         else
12             Use n leaf nodes of R in mixer M;
13             L^d[0] = (R, r − n);
14             Sort L^d in the descending order of r_i's;
15             break;
16 return T;
```

We have proposed a greedy algorithm to permute leaf nodes appearing in a mixing tree. Algorithm 1 permutes leaf nodes appearing in the depth d of a mixing tree (T) to minimize the number different input reagents used in the mixing nodes at depth $d-1$. Note that, we can maintain L^d as a max-heap for efficient implementation of Algorithm 1. Since we need to permute all leaf nodes appearing in the same depth of the mixing tree, Algorithm 1 is called for depth $d = 2, 3, \cdots, h$, where h is the maximum depth of all leaf nodes in T. The following example illustrates Algorithm 1 on the mixing tree shown in Fig. 4(a).

Example 4. *Consider the mixing tree shown in Fig. 4(a) and let us permute leaf nodes of the mixing tree at depth 3. The sorted list of input reagents is* $L^3 = [R_1 : 3, R_2 : 3, R_3 : 3, R_4 : 3]$ *and the number of reagents used in mixing nodes* M_5, M_6, *and* M_7 *at depth 2 is represented as a list* $M^2 = [(M_5, 4), (M_6, 4), (M_7, 4)]$. *Algorithm 1 iterates over the mixing nodes in* M^2 *and assigns the required number of input reagents from* L^3 *to each mixing node (lines 4-15). In the first iteration, three reagents of* R_1 *are added to the mixing node* M_5 *and update list* $L^3 = [R_2 : 3, R_3 : 3, R_4 : 3]$ *(lines 6-10). In the second iteration, one reagent of* R_2 *is added to* M_5 *and update list* $L^3 = [R_3 : 3, R_4 : 3, R_2 : 2]$ *(lines 12-15). Similarly, three reagents of* R_3 *and one reagent of* R_4 *is added to* M_6. *Finally, remaining two reagents of* R_2 *and* R_4 *are added to* M_7. *Algorithm 1 is applied to leaf nodes at depth 2 of the mixing tree. After permuting leaf nodes in each depth, the transformed mixing tree is shown in Fig. 7(a).*

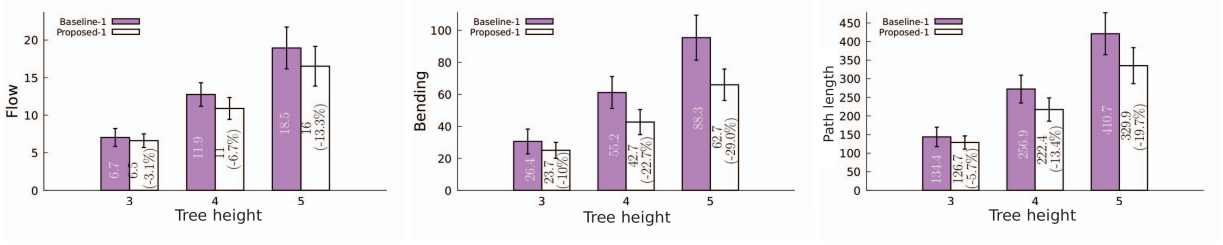
IV. Simulation results

We perform an extensive simulation to analyze the performance of the proposed transformation on mixing tree in the design automation of sample preparation with FPVA biochip. In our experiments, we have considered 750 mixing ratios by varying the number of input reagents to three, four, and five. These mixing ratios are further partitioned into three equal-sized groups (250 mixing ratios in each group) based on the height of the mixing trees.

We perform two sets of experiments and calculate the average number of reagent loading cycles (K), the 90° turns (B) (measures the path complexity), and the flow path length (L) for each group of mixing ratios. In the first set of experiments, we generate the mixing tree using genMixing [10]. Note that, a sequence of algorithms (a.k.a recipe) (genMixing+NTM+(LAFCA+DFL)) is used to determine desired operations to realize the target mixing ratio on the FPVA biochip (Fig. 5). Let us denote the above recipe as baseline-1, i.e., baseline-1: genMixing+NTM+(LAFCA+DFL). Note that we apply the proposed transformation (permuteLeaf) to permute the leaf nodes of the genMixing tree. Let us denote the proposed recipe as proposed-1 (genMixing+permuteLeaf+NTM+(LAFCA+DFL)). Fig. 8 shows the comparative results of the average value of (K, B, L) parameters associated with fluid loading operations in sample preparation. In the second set of experiments, we use the recipe genMixing+HDA+NTM+(LAFCA+DFL) (denoted as baseline-2) in the design automation of sample preparation on FPVA. We apply the proposed transformation on the resultant mixing tree after applying HDA on the genMixing tree. Let us denote the proposed recipe as proposed-2 (genMixing+HDA+permuteLeaf+NTM+(LAFCA+DFL)). Fig. 9 shows the comparative results of the average value of (K, B, L) parameters associated with fluid loading operations during sample preparation.

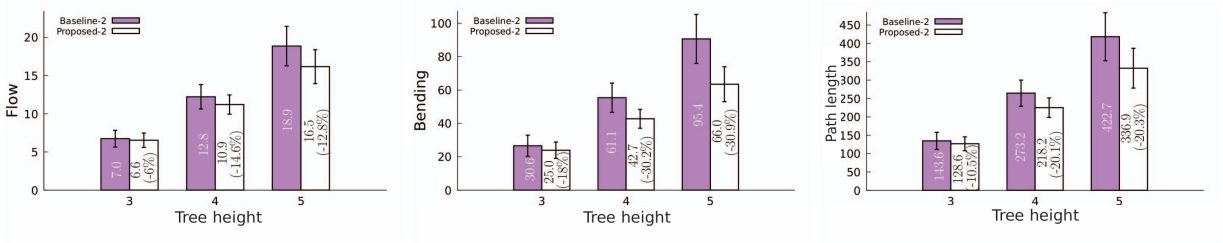
From the plots (Figs. 8(a)-(c) and Figs. 9(a)-(c)), one can observe that the average values (given in the bar chart) of (K, B, L) parameters decrease when the proposed transformation is applied to the mixing tree, i.e., reducing the sample preparation time and cost. Moreover, the %-savings in the average values of the K, B, and L increase with the increase in the height of the mixing tree.

We have also considered a few real-life and synthetic mixing ratios to show the exact value of (K, B, L) parameters in Table I. For all mixing ratios, (B, L) parameters have been decreased i.e., the length and complexity of the loading path decrease considerably when the proposed transformation is applied on the mixing tree.



(a) Avg. number of flows (K) (b) Avg. number of bending in a flow (B) (c) Avg. length of a flow (L)

Fig. 8: Histograms show the average value of (K, B, L) parameters (shown inside) of mixing ratios partitioned into three groups based on the height of the mixing tree. Error bars denote the standard deviation. The recipe of baseline-1 and proposed-1 are genMixing+NTM+(LAFCA+DFL) and genMixing+permuteLeaf+NTM+(LAFCA+DFL), respectively. For the proposed recipe, %-saving is shown inside the histogram.



(a) Avg. number of flows (K) (b) Avg. number of bending in a flow (B) (c) Avg. length of a flow (L)

Fig. 9: Histograms show the average value of (K, B, L) parameters (shown inside)of mixing ratios partitioned into three groups based on the height of the mixing tree. Error bars denote the standard deviation. The recipe of baseline-2 and proposed-2 are genMixing+HDA+NTM+(LAFCA+DFL) and genMixing+HDA+permuteLeaf+NTM+(LAFCA+DFL), respectively. For the proposed recipe, %-saving is shown inside the histogram.

TABLE I: Performance of the proposed algorithms.

Real-life mixing ratio				
Mixing ratio	baseline-1	proposed-1	baseline-2	proposed-2
	(K, B, L)	(K, B, L)	(K, B, L)	(K, B, L)
ChIP Lysis Buffer [12] 67:38:3:67:14:67	$(18, 75, 364)$ $(5, 3)$	$(18, 57, 286)$ $(5, 3)$	$(17, 91, 365)$ $(5, 3)$	$(13, 62, 273)$ $(5, 5)$
Silver Restc. Digest [12] 180:26:26:5:5:14	$(16, 71, 346)$	$(17, 57, 337)$	$(16, 77, 384)$	$(16, 58, 320)$
Plasmid DNA [10] 56:113:87	$(9, 39, 183)$	$(9, 39, 181)$	$(9, 47, 193)$	$(9, 33, 173)$
Splinkerette PCR [10] 102:26:2:2:124	$(12, 57, 358)$	$(13, 50, 267)$	$(12, 64, 260)$	$(12, 42, 232)$
Synthetic mixing ratio				
19:15:15:15	$(9, 57, 221)$	$(9, 38, 185)$	$(11, 58, 237)$	$(8, 41, 172)$
318:123:234:237:112	$(26, 112, 570)$	$(18, 80, 392)$	$(24, 134, 554)$	$(21, 80, 409)$
179:254:289:99:203	$(21, 101, 483)$	$(17, 73, 373)$	$(24, 111, 510)$	$(21, 77, 417)$
167:43:19:27	$(15, 72, 321)$	$(12, 55, 256)$	$(13, 77, 299)$	$(13, 50, 255)$

V. Conclusions

In this paper we propose an algorithm to speed-up (reduce) the sample preparation time (cost). The proposed algorithm leverages combinatorial properties of mixing trees to permute the leaf nodes in the mixing tree to minimize the reagent loading time and amount of reagent usage during sample preparation. The resultant fluid loading paths are simpler (take lesser 90° turns) and shorter, i.e., less number of valve actuation is required to create the fluid loading path hence, increase the reliability of the fully programmable valve array.

References

[1] S. F. Berlanda, M. Breitfeld, C. L. Dietsche, and P. S. Dittrich, "Recent advances in microfluidic technology for bioanalysis and diagnostics," *Analytical Chemistry*, vol. 93, no. 1, pp. 311–331, 2021.

[2] S.-M. Yang, S. Lv, W. Zhang, and Y. Cui, "Microfluidic point-of-care (POC) devices in early diagnosis: A review of opportunities and challenges," *Sensors*, vol. 22, pp. 1620:1–33, 2022.

[3] 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.

[4] Microfluidics for DNA analysis. [Online]. Available: https://www.elveflow.com/microfluidic-reviews/general-microfluidics/microfluidics-for-dna-analysis-pcr/

[5] N. Convery and N. Gadegaard, "30 years of microfluidics," *Micro and Nano Engineering*, vol. 2, pp. 76–91, 2019.

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

[7] S. Bhattacharjee, B. B. Bhattacharya, and K. Chakrabarty, *Algorithms for Sample Preparation with Microfluidic Lab-on-Chip*. River Publisher, 2019.

[8] G. Choudhary, S. Pal, D. Kundu, S. Bhattacharjee, S. Yamashita, B. Li, U. Schlichtmann, and S. Roy, "Transport-free module binding for sample preparation using microfluidic fully programmable valve arrays," in *Proc. DATE*, 2020, pp. 1335–1338.

[9] P. Spedding, E. Benard, and G. Mcnally, "Fluid flow through 90 degree bends," *Asia-Pacific Journal of Chemical Engineering*, 2008.

[10] 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.

[11] M. Hirai, D. Kundu, S. Yamashita, S. Roy, and H. Tomiyama, "Transport-free placement of mixers for realizing bioprotocol on programmable microfluidic devices," in *Proc. VLSID*, 2023, pp. 193–198.

[12] D. Kundu, J. Giri, S. Maruyama, S. Roy, and S. Yamashita, "Fluid-to-cell assignment and fluid loading on programmable microfluidic devices for bioprotocol execution," *Integration*, vol. 78, pp. 95–109, 2021.