PAPER

Reducing the Number of Flushing by Scaling Mixers on PMDs

Masataka HIRAI[†a)], **Debraj KUNDU**[††b)], ***Nonmembers***,
Shigeru YAMASHITA[†c)], ***Senior Member***, **Sudip ROY**[†††d)], ***Nonmember***,
and **Hiroyuki TOMIYAMA**[††††e)], ***Senior Member***

SUMMARY This paper proposes an efficient method for reducing the number of *flushing* operations on *Programmable Microfluidic Device (PMD)*, which consists of an array of valves and cells. Due to its programmability and grid channel-based architecture, an intensive amount of research work has been done on PMD in recent years. During the synthesis of any bioassay (e.g., a mixing tree) on PMD, we may need flushing operations to wash away unnecessary intermediate droplets left in some cells. It is essential to reduce the number of necessary flushing operations as it increases the synthesis time and also the usage of *buffer* solutions. Hence, our proposed method minimizes the flushing by scaling the size of the mixers and transforming the given mixing tree. Our experimental results confirm that after the proposed transformation, we can reduce the number of flushing operations by an average of 81.34%.

key words: *Programmable Microfluidic Device (PMD), Flushing, Scaling*

1. Introduction

In recent years, microfluidic biochips have been gaining attention as new experimental devices replacing conventional laboratory-scale devices in the field of biochemistry [1] [2] [3]. One type of biochip, called Programmable Microfluidic Device (PMD) [4], has garnered particular interest [5] [6] [7]. A PMD consists of a network of flow and control layers, forming an array of valves and cells. By controlling the opening and closing of valves, PMD creates circular flow paths called mixers to mix multiple droplets [8] [9].

A PMD can generate a mixed solution of multiple reagents in the desired ratios by performing a sequence of mixing operations. This process is called reagent synthesis [4] [10]. During reagent synthesis on a PMD, there are cases where intermediate droplets generated by some mixers may be left in cells that should not be used by other subsequent mixers, hindering further mixing. This state is referred to as *mixer overlap*. Mixer overlap should be solved by performing an operation called *flushing*, where the remaining intermediate droplets are washed away with a liquid called buffer solution [11]. The amount of buffer solution used in reagent synthesis increases proportionally with the number of flushing operations, leading to an increase in experimental costs. Moreover, the scheduling time also increases proportionally with flushing operations. Therefore, reducing the number of flushing operations in reagent synthesis is necessary to mitigate the increase in experimental costs.

Various methods of reagent synthesis on PMDs have been studied [12]. One method for generating droplet mixing sequences in reagent synthesis on PMDs is No Transport Mixing (NTM) [13]. Additionally, a method using priority assignment has been proposed to reduce the number of flushing operations in reagent synthesis [14]. To improve such existing works, this paper proposes a new idea, "scaling of mixers," especially for reducing the number of flushing operations in reagent synthesis on PMDs.

Contributions of This Paper.

- Given a mixing tree, we first focus on the possible *mixer overlaps* where flushing operations are necessary. Then, we utilize a unique idea to transform the mixing tree by *scaling* some mixing nodes that help to reduce flushing operations.
- We also propose a heuristic for scheduling and placing mixers of the transformed mixing tree.
- Experimental comparisons confirm that our new idea, *scaling* can indeed reduce the number of necessary flushing operations by 81.34% when we perform droplet mixing procedures for mixing trees of height 5.

This paper is organized as follows. Section 2 explains the basic knowledge of reagent synthesis using PMDs. Section 3 explains our proposed method to reduce the number of flushing operations in reagent synthesis using our new idea: "scaling". After that Section 4 presents the experimental results and discussions.

[†]The author is with the Graduate School of Information Science and Engineering, Ritsumeikan University, Shiga, 525-8577 Japan.
[††]The author is with the Technical University of Munich, Muhich, 80333 Germany.
[†††]The author is with the Indian Institute of Technology Roorkee, Uttarakhand, 247667 India.
[††††]The author is with the Indian Institute of Technology Roorkee, Uttarakhand, 247667 India.
a) E-mail: miku@ngc.is.ritsumei.ac.jp
b) E-mail: debraj.kundu@tum.de
c) E-mail: ger@cs.ritsumei.ac.jp
d) E-mail: sudip.roy@cs.iitr.ac.in
e) E-mail: ht@fc.ritsumei.ac.jp
DOI: 10.1587/transfun.E0.A.1

Finally, Section 5 concludes the paper with future work.

2. Preliminary

In this section, we explain the preliminaries of a Programmable Microfluidic Device (PMD) and the synthesis of any mixing tree on it.

2.1 Architecture of PMDs

A microscopic image of a PMD is shown in Figure 1, where a flow-channel section (blue) called cells is surrounded by four valves (red). The valves are used for manipulating and controlling fluid operations and movements within PMD. As evident from Figure 1, PMD has a structure where cells surrounded by valves are arranged in a grid pattern, which increases the programmability of fluidic operations on PMD. Figure 2 illustrates the process of mixing droplets of two types of reagents using a PMD [13]. Figures 2 (a-b) demonstrate the injection of two types of reagents into the PMD from input ports by applying pressure. Figure 2 (c) depicts a flow path called a mixer, which consists of cells connected in a circular manner. Mixing of droplets is achieved by controlling the opening and closing of valves in the mixer, which generates a peristaltic motion. After the mixing of droplets, the concentration of droplets on the mixer becomes uniform, as shown in Figure 2 (d). The mixer in Figure 2 (c) uses a 2×2 cell configuration of the PMD for droplet mixing. Such a mixer using a 2×2 cell configuration is referred to as a 2×2 mixer, and its size is referred to as 4 in this work. Using the same approach, we can also configure larger mixers on PMD.

A mixer is modeled as a collection of continuous cells arranged to form a closed loop such that a Hamiltonian cycle exists within the structure. This cyclic path is crucial, as fluids are circulated and mixed through peristaltic actuation of valves. Therefore, configurations such as 2×2 or 4×2 cells, as well as rectangular shapes with notches (provided they contain an even number of cells), are considered valid mixer designs. In this work, we focus on mixers with volumes of 4 and 6. However, after a scaling operation, the volume may double, resulting in mixers of volume 8 and 12. To initiate any operation on such mixers, we need to load desired reagents into each cell of a mixer. A loading must start at the in-port, end at the out-port, pass through the required cells to be filled, and avoid intersecting any filled region on a PMD.

2.2 Reagent Synthesis

Reagent synthesis is the process of generating a desired ratio of certain reagents, which is achieved through a sequence of reagent mixing of PMD. Figure 3 illustrates the input and output data of reagent synthesis using a

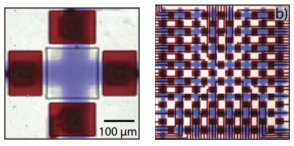


Fig. 1: Microscopic image of a PMD.

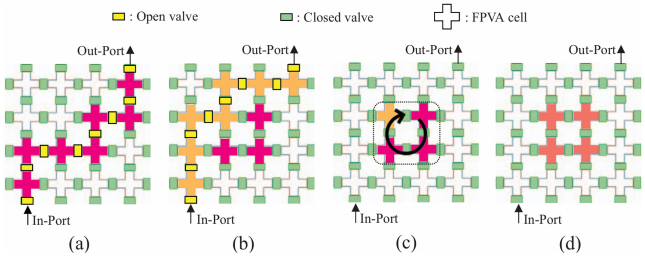


Fig. 2: Process of mixing two types of reagents poured onto a PMD, from reference [13].

.

method based on priority placement [14].

In Figure 3 (a), a tree-structured data called a mixing tree is shown, which represents reagent synthesis that mixes four types of reagents, R1, R2, R3, and R4, in the ratio 24:18:12:10. Nodes located on the leaves of the mixing tree represent reagent droplets. Nodes other than reagent nodes represent mixing nodes, denoting droplet mixing using mixers. Mixing node Mi represents the mixing of droplets in the i^{th} mixer.

Figures 3 (b) to (f) depict the actual droplet mixing steps on a PMD corresponding to the logical mixing nodes presented in Figure 3 (a). In Figure 3 (b), droplets are mixed at M2 and M3. In order to place M4, a flushing operation to wash out a cell of M3 is shown in Figure 3 (c). Then in the next step M4 is placed as shown in Figure 3 (d). Then in Figure 3 (e) the placement of M1 is shown and finally in Figure 3 (f) M0 is placed. The method based on priority placement [14] takes the mixing tree shown in Figure 3 (a) as input and generates droplet mixing steps like those in Figures 3 (b) to (f) by exploring the arrangement of reagents and mixers based on the mixing tree.

2.3 Flushing

During reagent synthesis on a PMD, there are cases where PMD cells used for mixing between multiple mixers may overlap. For instance, in Figure 3 (b-c), we observe that M4 has a single-cell overlap with M3. In such a case, the mixing of M4 cannot be initiated unless the intermediate droplets generated by the previous mixer M3 are removed from the overlapped cell(s). To ensure a proper fluid sharing between child nodes and their parent node we need to overlap their placements.

However, some of those placement forces us to wash out certain cells to ensure a proper sharing. This

state is referred to as "mixer overlap". In order to get rid off any "mixer overlap", we need to wash out the droplets in the concerned cells. This operation is referred to as "flushing". Flushing is similar to a loading operation, except that a buffer reagent is used to clear the desired cells within the overlapped region. As with loading, flushing must ensure that the loading path starts at the in-port and ends at the out-port without intersecting any filled regions of the PMD, except for the specific overlapped region which needs to be flushed. In the mixing tree shown in Figure 3 (a), the child nodes M3 and M4 shares 3 unit and 1 unit volume, respectively to the parent node M1. Once M3 is placed as shown in Figure 3(b), our next task is to place M4, such that when M1 will be placed it should overlap exactly with 3 and 1 cells of M3 and M4, respectively. However, to achieve that we need to flush a cell from M3 (Figure 3(c)) and then we have to place M4 (Figure 3(d)). Finally, as shown in Figure 3(e), we need to place M1 with proper overlapping with its child nodes.

This example implies that if the sharing volumes to a parent node from two child nodes is 3:1 or 5:1; there should always be "mixer overlap" between those child nodes. As shown in Figure 3(c), to resolve this overlap issue flushing is performed by a *buffer* solution, that wash away the intermediate droplet residing in the cell marked as shaded region. As flushing operation increases the completion time for the synthesis of a mixing graph, our proposed method aims to reduce the number of necessary flush operations. Our observation is that it is possible to predict the "mixer overlap" to some extent based on the size of the child mixers and the shared volumes to their parent mixer, which will be discussed in the next section.

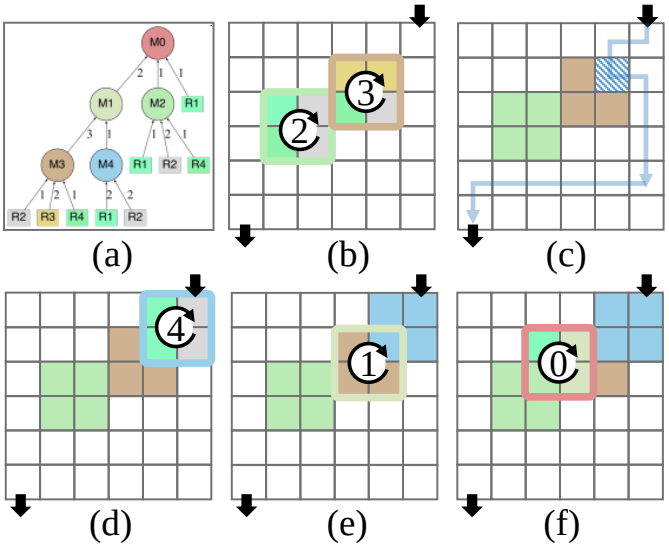


Fig. 3: A mixing graph, and the different stages of the reagent synthesis via placement of its mixing nodes.

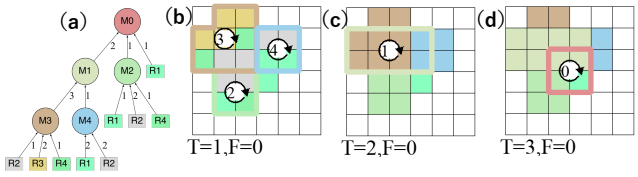


Fig. 4: (a) Input for the proposed method: mixing trees, (b), (c) and (d) Output: mixing procedure of droplets on a PMD.

3. Reducing the Number of Flush Operations by Scaling Mixers

3.1 Outline of the proposed method

Here, we describe the outline of the proposed method. The input to the proposed method is a mixing tree, as shown in Figure 4 (a) and the output is the scheduled mixer placement information, as shown in Figure 4 (b-d). In Figure 4 (b-d), the value T means the time step when the mixing occurs, and the value F means the number of flush operations performed before T. The proposed method aims to perform reagent synthesis with fewer flush operations compared to the existing method using placement priority mentioned in the previous section [14].

Our proposed method can be broadly divided into two processes: (i) transformation of the input mixing tree using *scaling*, and (ii) heuristically determining the scheduling and the placement of mixers (Mi) in the mixing tree.

3.2 Transformation Algorithm for Input Mixing Trees using Scaling

In this section, we describe the transformation process of an input mixing tree by using our new idea: "scaling".

3.2.1 Overview of Scaling

"Scaling" of a mixer is an operation that multiplies all the input volumes of a mixer Mi by the same integer factor, such as double, triple, etc. For instance, after scaling the input volumes of M5 , which is 3:1 in the original mixing tree shown in Fig.5 (a), the input volumes are doubled to 6:2 in the modified mixing tree Fig.5 (c). Additionally, due to the "Scaling", the mixer size of M5 has changed from 4 to 8. By using Figure 5, we explain in a detail how we transform a mixer by using scaling. First, between Figures 5 (a) and (b), the input volumes to M5, which is the target of the scaling, is doubled. As a result, in some cases, contradictions may arise within the mixing tree, as shown in Figure 5 (b). The contradiction in the mixing tree in Figure 5 (b) arises from the mixer node M9 whose size is 4 but it

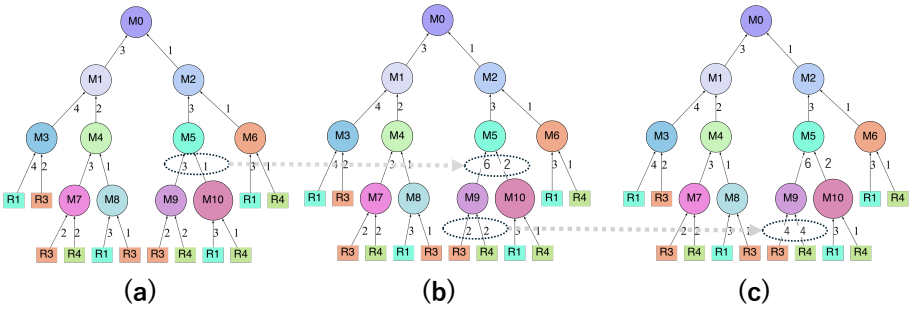


Fig. 5: Details of the transformation of input mixing trees using scaling.

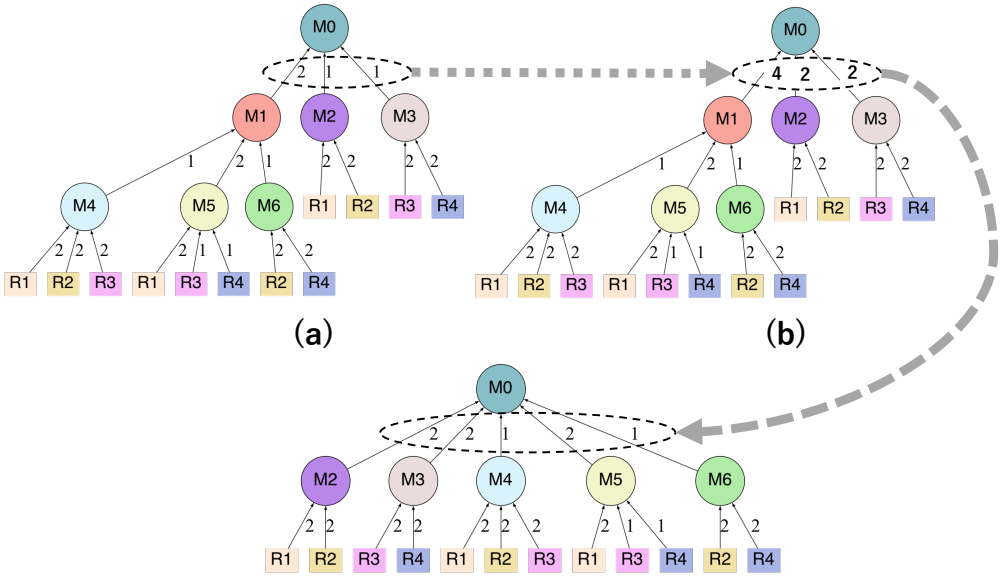


Fig. 6: Merge of mixer nodes during scaling.

should provide 6 intermediate droplets to M5. A mixer cannot provide more intermediate droplets than its own size. In order for the mixer of M9 to provide 6 intermediate droplets to the parent node, M5, the mixer size of M9 needs to be increased to at least 6. Therefore, between Figures 5 (b) and (c), the input volume to M9 is also doubled by scaling, increasing the mixer size of M9 to 8. The scaling of the input volumes to M9 resolves the contradictions within the mixing tree. After Scaling, it may be possible to merge some mixer nodes. An example of a mixing tree where mixer node merging is performed after Scaling is shown in Figure 6. In Figure 6, Scaling is performed on the input volumes to M0. Due to the scaling, the input volumes to M0, which is 2:1:1 in the mixing tree as shown in Figure 6 (a), becomes 4:2:2 in the mixing tree as shown in Figure 6 (b). In the mixing tree as shown in Figure 6 (b), M1 is a size-4 mixer, and it provides all 4 droplets mixed in the mixer to the parent node M0's mixer. Therefore, we can just transfer all the 4 input droplets of M1 directly to M0 without mixing them in M1. Since mixing of droplets in M1 is unnecessary, M1 can be merged into M0. This simplification is called merging of mixer nodes, which transforms the mixing tree from Figure 6 (b) to Figure 6 (c).

The architectural constraints of PMD with a single inlet and outlet port, incurs reagent wastage during its loading operation. To load m cells, $n - m$ cells of reagent wastage will be incurred in each loading cycle, where n is the length of the loading path. Since the flow cannot be stopped precisely at the point when the loading path is filled, it is necessary to flow much more reagent than required. In conclusion, the reagent wastage highly depends on the number of loading cycles, and not on the number of cells loaded. Hence, to reduce reagent wastage, we need to reduce the number of loading cycles required for the entire operation.

The theory behind Scaling is to reduce the number of flushing and eventually helps in reducing overall loading cycles. The additional reagent usage introduced by Scaling, relative to multiple loading cycles, is negligible. Hence, the Scaling does not significantly affect the reagent requirement for the entire process. On contrary, merging of mixing nodes due to Scaling reduces

the number of loading counts for reagents, which in turn helps in reducing reagent wastage.

3.2.2 Overview of Mixer Overlap

It is possible to predict the occurrence of mixer overlap to a certain extent based on information such as the input volumes of a mixer. Our idea is to consider the relationship between input volumes and mixer overlap as follows. We consider the following three groups of mixers based on input volumes to predict the likelihood of mixer overlap during reagent synthesis.

Group A Mixers that provide 3 or more odd-numbered droplets to their parent mixer node.
Group B Mixers that provide 4 or more even-numbered droplets to their parent mixer node.
Group C Mixers that provide less than 3 droplets to their parent mixer node.

A mixer in Group A has higher probability to cause a "mixer overlap", as shown in Figure 3 (a), where M3 provides 3 unit droplets to M1 causing a "mixer overlap". To ensure a no-transport placement, child mixer nodes are positioned to overlap with the placement cells of their parent mixer node such that the number of overlapped cells matches the number of shared droplets. Therefore, especially when the size of the parent mixer is small, such as 4 or 6, the placement cells of child mixers are more likely to be placed close to each other, which increases the chance of "mixer overlap".

As explained in Section 3.2.1, Scaling enlarges the mixer size, which convert the mixers from Group A to Group B, and reduces the chance of "mixer overlap". However, due to the size increment of the parent mixer by Scaling, the demand from it's child nodes also increases. As a result we also need to increase the size of the respective child nodes in equal proportion. Consequently, as the size of the parent mixer increases, the options for arranging child mixers also increase, making it easier to select an arrangement that avoids "mixer overlap". After the Scaling we will be left with mixers belonging to either Group B or Group C. The difference between these two groups lies in the number of droplets provided, expecting that Group C has a very low probability of causing overlaps.

In Figure 7 (a), applying Scaling transforms the mixing tree into the one as shown in Figure 7 (b). When we generate the mixing procedure on a 10x10 cell PMD by the algorithm mentioned in Section 3.3, the mixing tree in Figure 7 (a) requires four flush operations for the reagent synthesis, while with the proposed approach we need only one flush operation for the mixing tree, as shown in Figure 7. Flushing is dependent on (i) placement of the mixers, (ii) size of the mixers, and (iii) size of PMD. The following example shows that even though we have enough space with a 10x10 PMD, the mixing graph in Fig. 8(a), will always require 4 flushing. Hence, we aim to determine an efficient placement that reduces the number of flushing requirement for "mixer overlap".

3.3 Algorithm for Generating Droplet Mixing Procedure Using Large Mixers on a PMD

Our proposed method to generate droplet mixing procedures on a PMD can be broadly divided into two steps. The first step is to reorder the sequence of mixer placement for a reagent synthesis based on the estimated number of cells used by each sub-tree in a given mixing tree. This process involves rearranging the decision sequence of mixer placement based on a metric called *estimated cell use number (ECN)*, which will be defined later. The second step is determining how to place mixers based on the ECN.

3.3.1 Algorithm for Reordering Decision Sequence of Mixer Placement Using Estimated Cell Use Numbers (ECN)

In the algorithm for reordering the decision sequence of mixer placement using the ECNs, we calculate an estimated cell use number (ECN) for each mixing node in the input mixing tree. ECN for a mixing node Mn (where n is the mixing node number) is calculated using Equation 1. The ECN is computed as the sum of the reagent volumes (Reagent Vol.) for Mn and the ECNs of its child nodes.

$$\mathrm{ECN}(Mn) = \mathrm{ReagentVol}(Mn) + \sum \mathrm{ECN}(Mn\text{'s child nodes}). \quad (1)$$

Equation (1) for the Estimated Cell Use Number calculates the maximum sum of cell usage for mixing droplets across all mixing nodes in the subtree rooted at the target mixing node.

After calculating ECNs, the child nodes of each mixing node are reordered according to their ECNs. For example, in Table 1, the ECN(M2) > ECN(M1). Therefore, the placement decision for M0's child nodes is made, which states M2 must be placed before M1.

Our proposed approach utilizes ECNs as an approximate measure of the number of cells required by the mixing nodes in the sub-tree until all droplet mixing is completed. Mixing nodes having higher ECNs are more likely to overlap with other mixers, as they require more cells until all the mixing operations in the

Table 1: Estimated cell use numbers assigned to each mixing node in the mixing tree of Figure 7 (b).

	M1	M2	M3	M4	M5	M6	M9	M10	M11
ECNs	20	22	12	8	12	10	6	6	8

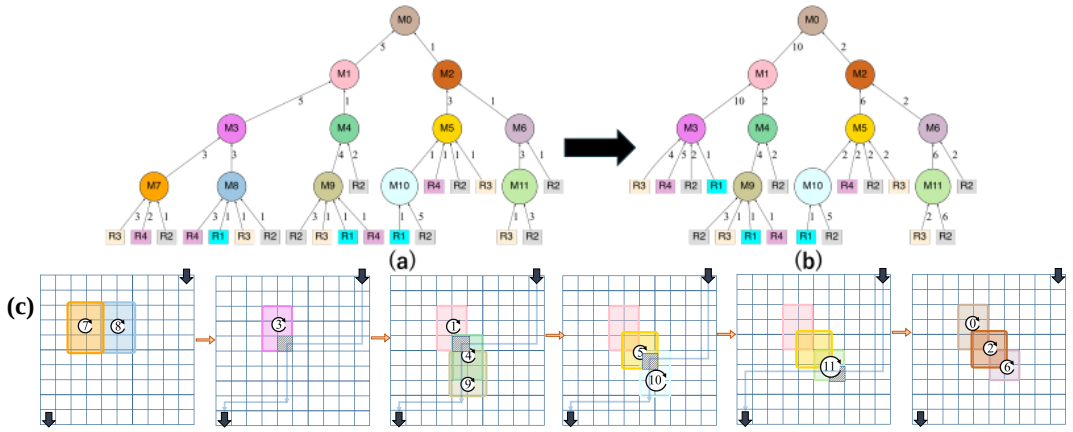


Fig. 7: Illustration of a mixing graph (a) before and (b) after Scaling. (c) Sequence of placement for the mixing nodes in the graph shown in Fig. 7(a) on a 10 × 10 PMD, which requires 4 flushing due to "mixer overlap".

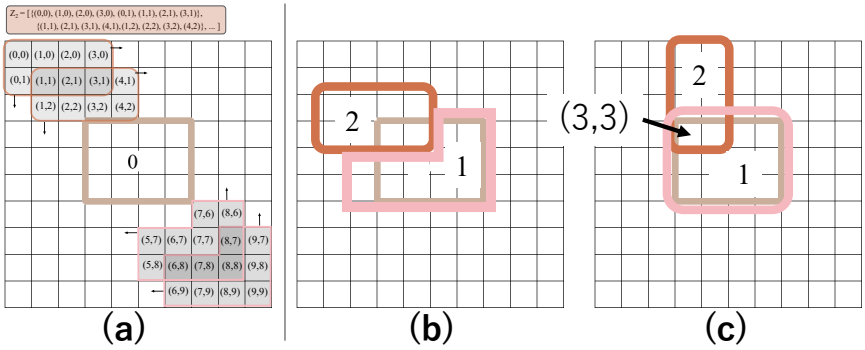


Fig. 8: (a) Process of determining the mix positions using mixing window. Placement of the child nodes of M0, (b) without flushing, and (c) with flushing.

sub-tree are completed.

Hence, our approach prioritizes the placement of nodes with higher ECNs and place them earlier when more empty cells are available on the given PMD. This ensures that we have more freedom to select the placement configurations that prevent mixer overlap as much as possible.

3.3.2 Mixing Node Placement Algorithm

In this section, we explain the second step of the proposed method. As mentioned earlier, we determine the placement of mixers according to the reordered placement schedule, based on ECNs. For a given mixing tree and an NxM PMD, let L be the ordered list of the mixers to be placed. Considering these inputs, we then propose an iterative approach for placing the mixers to minimize the number of flushing operations. At each iteration, we place all the child mixers of a parent mixer. Using the size of each mixer, we determine all possible mix positions using mixing-window for that mixer, and then enumerate all valid placements of the child mixers that maintain proper overlap with their parent. The placement order of the child mixers is determined by their ECN values, with mixers having larger ECN values placed first. The proposed placement approach is presented in **Algorithm** 1.

At each iteration, we choose a parent mixer and then according to its known placement we aim to find a good placement of its child mixers. In the first iteration, at **line 4** of **Algorithm** 1, we choose to place the root node at the center of the chip. For next iterations, we always get the placement of the parent mixer from a previous step. The satisfying conditions for the placement ensuring proper mixing are presented in **line 14**. Out of all possible placements stored in P, we choose the placement with minimum flushing requirement at **line 15** Finally, at **line 17** we get the final placement configurations of each mixer stored in $\bar{P}$.

Figure 8(a) shows the technique to determine all possible mix positions for a given size mixer. We first choose a widow of the same size as the mixer and then scan that window across a PMD and choose the corresponding cells. Figures 8 (b) and (c) show two different placements of mixers of M1 and M2, which are child nodes of M0. We need to place the child nodes M1 and M2 in a way that shares 10 and 2 droplets, respectively, with the parent node M0. The placement method in

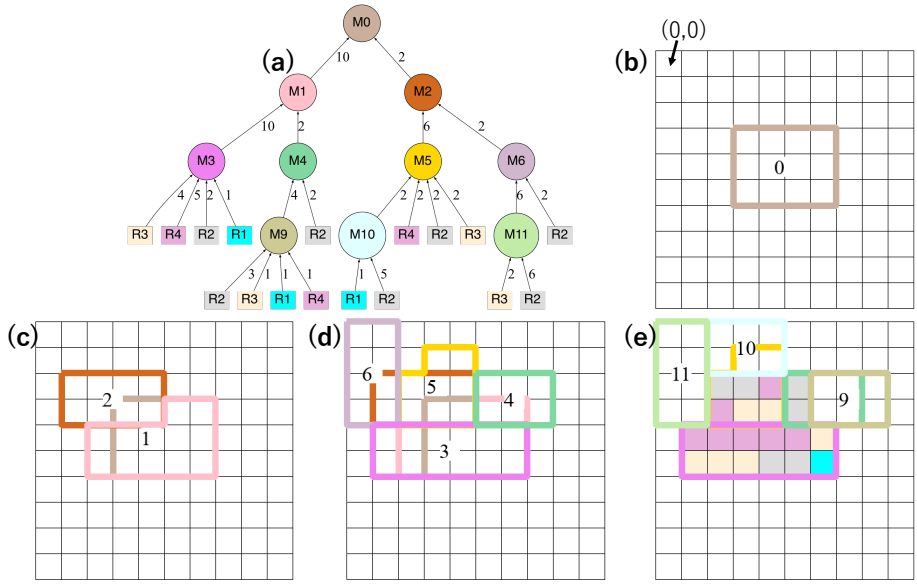


Fig. 9: The process of placing mixers on a 10x10 cell PMD.

Figure 8 (b) allows the mixer cells of M1 and M2 not to overlap with each other, enabling the placement of M1 and M2 mixers in the same time step. However, according to the placement shown in Figure 8 (c), the mixer cells of M1 and M2 overlap with each other. As mentioned in Section 2.3, such "mixer overlap" needs flush operations. However, the provided droplets to the parent node M0 should not be washed away because they will be required for mixing at M0. Thus, in the placement method of Figure 8 (c), if the mixers are placed in the order of M2 followed by M1, the provided droplets to M0 left by M2 at cells (3,3) and (4,3) need to be flushed away to allow mixing by M1. Consequently, it is impossible to perform mixing by placing the mixers in the order of M2 followed by M1 in the placement method of Figure 8 (c).

Algorithm 1: Mixer Placement

Input: (i) Mixing tree (T) consisting of Mi and Rj, (ii) placement sequence (L) of Mi based on ECN, and a PMD of size $N \times M$
Output: Placements of the mixers Mi on PMD
begin
  while *L is NOT Empty* **do**
    $X \leftarrow L[0]$;/* X is the current parent node */
    $\bar{X} \leftarrow$ add the placement cells of $L[0]$;/* for root node M0, place at the center of $N \times M$ */
    $Y = \{\}$;
    $Y \leftarrow$ child mixining nodes of X;
    if $Y \neq \{\}$ **then**
      for $i \leftarrow 1$ *to* $|Y|$ **do**
        $Z_i \leftarrow$ add all possible mix positions w.r.t the size of Y_i;
      $C \leftarrow Z_1 \times Z_2 \cdots \times Z_{|Y|}$;/* cross product of $\forall_i Z_i$ */
      $P = \{\{\}_1, \{\}_2, \cdots, \{\}_{|C|}\}$;
      for $i \leftarrow 1$ *to* $|C|$ **do**
        for $j \leftarrow 1$ *to* $|C_i|$ **do**
          /* $\overrightarrow{Y_j, X}$ denotes the edge weight between the child Y_j and parent X */
          $P_i \leftarrow P_i \cup C_{i,j}$, such that $|X \cap C_{i,j}| \geq \overrightarrow{Y_j, X} \bigwedge X \cup C_{i,j} \leq N \times M$;
      $\bar{P} \leftarrow$ add the placement from P requring smallest flush operation;
    Remove $L[0]$ from L;
  return Placement information stored in $\bar{P}$;

For determining the mixer placement, the proposed method first generates all the possible placements, as shown in Figure 8(a). Then, it evaluates the number of flush operations needed for each placement. When we evaluates the number of flush operations, as explained in Section 3.3.1, the proposed method rearranges the order of mixer placement based on the ECN values. However, as mentioned in the example shown in Figure 8 (c), there may be an inappropriate order of mixers; we simply exclude such orders. We select the best placement and the order of child mixers which minimizes the number of flush operations.

Figure 9 illustrates the placement sequence on 10×10 PMD. The top-left cell of the PMD in Figure 9 is denoted as $(x, y) = (0, 0)$. Figure 9 (a) represents the same mixing tree as Figure 7 (b). Initially, the root mixing node M0 is placed at the centre of the PMD, as shown in Figure 9 (b). Next, as illustrated in Figure 9 (c), mixers of child nodes M1 and M2 of M0 are placed on the PMD, overlapping with the placement cells of the M0 mixer. This process continues, as shown in Figure 9 (d) and (e), where mixers of child nodes are placed overlapping their parent mixer nodes.

During the mixer placement with enlarged mixers, we may run out of empty cells on PMD, leading to an impossible placement. In such a case, the operations proceed in the following order: loading of reagent

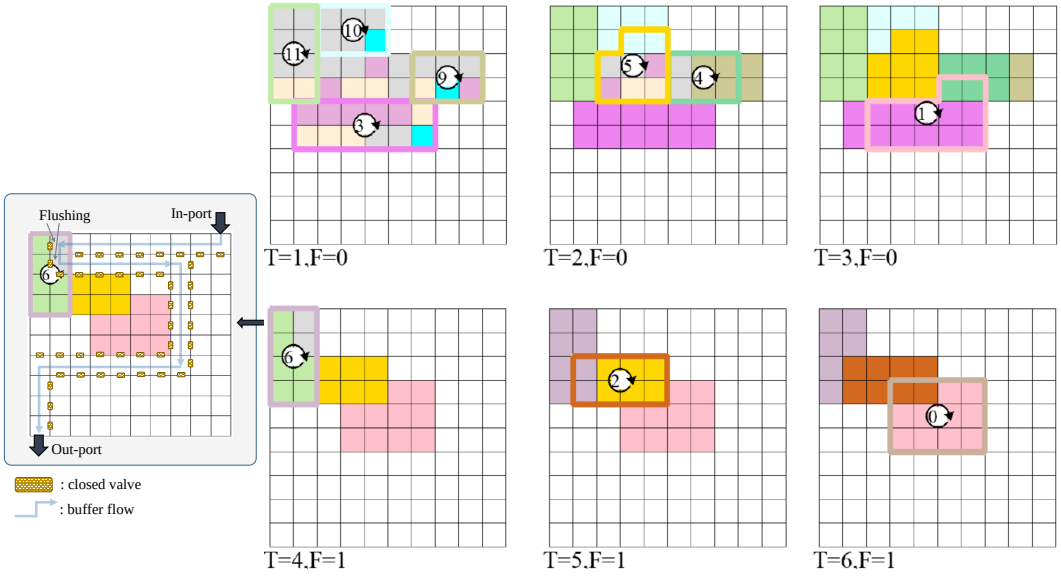


Fig. 10: Mixing procedure for reagent synthesis corresponding to the mixing tree in Figure 9(a).

droplets, mixing of droplets in the mixers, and flushing. Flushing creates space on a PMD, allowing for the continuation of mixer placement. The sequence of operations repeats, with mixer placement, reagent loading, mixing of droplets in mixers, flushing, and again mixer placement. Eventually, when the mixing of M0 is completed, the droplet mixing procedure as shown in Figure. 10 is generated. In Figure. 10, "T" represents the time step at which mixer mixing occurs, and "F" represents the number of flush operations performed until that time step. For the droplet mixing procedure corresponding to the mixing tree as shown in Figure. 9(a), it is evident that one flushing is required.

After placing the mixers starting from the root node, the reagent loading and mixing operations are performed in reverse order. Accordingly, we first load reagents into the mixing nodes that do not have any child nodes As depicted in Figure. 9, droplet mixing takes place starting from the mixers of the leaf mixing nodes, as shown in Figure. 10. In the first step reagents are loaded and mixed for M3, M9, M10 and M11 as shown in Figure. 10 at T=1. Then we repeat the same procedure for the successive parent mixing nodes. At $T = 4$, before mixing M6, we need to flush 2 cells from the previously placed M11, and load there with R2. So at T=4, we need a flushing operation for which F increases to 1.

4. Experimental Results

To evaluate the effect of scaling in the proposed method, we first generate mixing trees of heights 3, 4, and 5 randomly, with 300 mixing trees generated for each height as our benchmark input mixing trees, and then we generate the mixing procedures of each of the input mixing trees by our method with/without scaling by assuming a 10x10 PMD. It is difficult for us to compare our method exactly with the existing method [14], but we could consider that the existing method would be almost similar to our method without scaling.

Table 2 shows the comparison between the two methods with and without scaling. In Table 2, the second column from the left shows the average number of nodes in the input mixing trees. (Note that the height includes the leaf reagents, so a mixing trees of heights 3 has two levels of mixing nodes.) The third and fourth columns show the average number of overlaps of mixers by the method without and with scaling, and the fifth column shows the average reduction rate by scaling. The sixth and seventh columns show the average number of necessary flush operations by the method without and with scaling, and the eighth column shows the

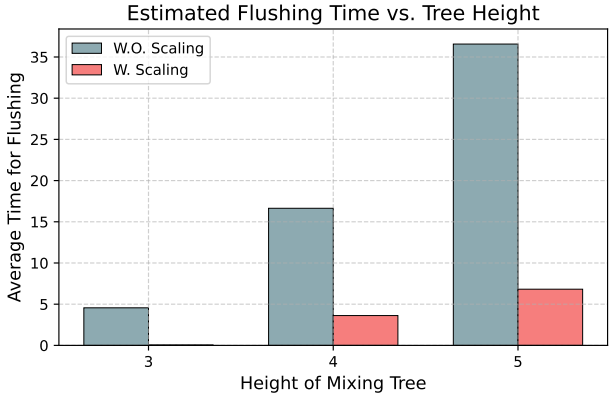


Fig. 11: Comparison of the estimation of average flushing time before and after Scaling operation.

Table 2: Experimental results.

Height of Mixing Tree	Avg. Number of Mixers	Avg. Number of Overlaps		Avg. Reduction of Overlaps (%)	Avg. Number of Flushing		Avg. Reduction of Flushing (%)
		W.O. Scaling	W. Scaling		W.O. Scaling	W. Scaling	
3	3.097	0.277	0.003	98.92	0.3	0.003	99.00
4	6.663	1.063	0.27	74.60	1.095	0.238	78.26
5	11.449	2.449	0.493	79.87	2.406	0.449	81.34

average reduction rate by scaling. The total processing time is influenced by three main factors: (i) mixing, (ii) flushing, and (iii) reagent loading. However, mixing time is highly dependent on fluid properties, and loading time varies with the chosen loading algorithm. Therefore, we exclude these two components from our calculation of overall processing time. Although the scaling operation may increase the size of some mixers, potentially affecting mixing time, it also reduces the total number of mixing operations. Hence, the resulting variation in mixing time is considered negligible. In this work, we focus on evaluating the effectiveness of our approach in minimizing flushing time. In practice, during a flushing we need to configure a loading path for the buffer, that passes through the desired cells and cleaned them for the future reagents to be placed.

According to Fidalgo et al. [13], to load a fluid in a cell, a PMD takes 7.6 sec. to configure all the valves and then to flush out the contents of that cell. As the number of cells needed to be washed can be one or more, in most cases we need to configure at least two separate buffer paths to flush out the contents. Hence, to get an estimation of the average time required for flushing, we multiply the average number of flushing with (2×7.6). We compute and report an estimation of average flushing time without and with Scaling in Figure 11, which demonstrates that the proposed Scaling operation significantly reduces the total time required for flushing. As presented in Figure 11, for height 5, the time required for flushing reduces from 36.57 sec. to 6.82 sec. after the Scaling operation. From this, we conclude that the proposed Scaling approach reduces the overall operation time required to synthesize a mixing graph.

Note that it is possible to resolve two or more overlaps by a single flush operation. Also, we may need a flush operation when there is no space left on a PMD. Thus, the number of overlaps does not necessarily relate to the number of flush operations. However, from the experimental results, we can observe that a smaller number of "mixer overlap" would need fewer flushing. Thus, our hypothesis underlying the proposed method, which states that reducing the number of overlaps is crucial in reducing the number of flush operations has been demonstrated to be somewhat correct. Mixing trees with larger heights are more likely to contain many mixing nodes that we can scale. Therefore, it is expected that taller mixing trees will be more affected by scaling transformations, leading to reduction in flushing operation as shown in experimental results.

5. Conclusion

A programmable microfluidic device is an advanced microfluidic biochip, which efficiently executes various fluidic operations and bioassays. In spite of the advantages of PMD, flushing is one of the most costly operations where we need to wash-out certain PMD cells. In this work, we propose a method to reduce the number of flushing cycles required to automate bioassays on PMDs. We propose a heuristic choice "Scaling", that transforms the given mixing tree and a heuristic approach to determine the placement of mixers. Experimental results confirm that it reduces the number of flushing operations by 81.34% while executing mixing trees of height 5. However, in future, we may need to develop an exact algorithm that optimizes mixer placement and minimizes the number of flushing operations.

References

[1] H. Stone, A. Stroock, and A. Ajdari, "Engineering flows in small devices: Microfluidics toward a lab-on-a-chip," Annual Review of Fluid Mechanics, vol.36, no.1, pp.381–411, 2004.

[2] J.P. Urbanski, W. Thies, C. Rhodes, S. Amarasinghe, and T. Thorsen, "Digital microfluidics using soft lithography," Lab on a Chip, vol.6, no.1, pp.96–104, 2006.

[3] V. Gubala, L.F. Harris, A.J. Ricco, M.X. Tan, and D.E. Williams, "Point of care diagnostics: status and future," Analytical chemistry, vol.84, no.2, pp.487–515, 2012.

[4] L.M. Fidalgo and S.J. Maerkl, "A software-programmable microfluidic device for automated biology," Lab Chip, vol.11, no.9, pp.1612–1619, 2011.

[5] A. Grimmer, B. Klepic, T.Y. Ho, and R. Wille, "Sound valve-control for programmable microfluidic devices," 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp.40–45, 2018.

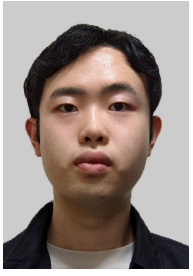
[6] S. Maruyama, D. Kundu, S. Yamashita, and S. Roy, "Optimization of fluid loading on programmable microfluidic devices for bio-protocol execution," 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), pp.550–555, 2020.

[7] G. Liu, Y. Zhu, W. Guo, and X. Huang, "Fault-tolerance-oriented physical design for fully programmable valve array biochips," 2023 60th ACM/IEEE Design Automation Conference (DAC), pp.1–6, 2023.

[8] T. Fu and Y. Ma, "Bubble formation and breakup dynamics in microfluidic devices: A review," Chemical Engineering Science, vol.135, no.2, pp.343–372, 2015.

[9] C. Liu, B. Li, B.B. Bhattacharya, K. Chakrabarty, T.Y. Ho, and U. Schlichtmann, "Testing microfluidic fully programmable valve arrays (fpvas)," Design, Automation Test in Europe Conference Exhibition (DATE), 2017, pp.91–96, 2017.

[10] S. Poddar, G. Fink, W. Haselmayr, and R. Wille, "A generic sample preparation approach for different microfluidic labs-on-chips," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.41, no.11, pp.4612–4625, 2021.
[11] Y.H. Lin, T.Y. Ho, B. Li, and U. Schlichtmann, "Block-flushing: A block-based washing algorithm for programmable microfluidic devices," 2019 Design, Automation Test in Europe Conference Exhibition (DATE), pp.1531–1536, 2019.
[12] S. Bhattacharjee, B.B. Bhattacharya, and K. Chakrabarty, Algorithms for sample preparation with microfluidic lab-on-chip, CRC Press, 2022.
[13] 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," 2020 Design, Automation Test in Europe Conference Exhibition (DATE), pp.1335–1338, 2020.
[14] M. Hirai, D. Kundu, S. Yamashita, S. Roy, and H. Tomiyama, "Transport-free placement of mixers for realizing bioprotocol on programmable microfluidic devices," 2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID), pp.193–198, 2023.



Masataka Hirai received his bachelor's degree from College of Information Science and Engineering, Ritsumeikan University in 2022. He received his master's degree from Graduate School of Information Science and Engineering, Ritsumeikan University in 2024. He is currently with LIXIL Corporation.

Debraj Kundu received the bachelor's degree in computer science and engineering from the West Bengal University of Technology in 2014 and the master's degree in computer science and engineering from the University of Calcutta in 2016. In 2023, he received Ph.D. degree from the Indian Institute of Technology Roorkee, India. His primary research interests include algorithm designing for the automation of bioprotocols on advanced microfluidic biochips. He was a JSPS postdoctoral fellow at Ritsumeikan University, Japan and is currently working as a postdoctoral fellow at the Technical University of Munich, Germany.

Shigeru Yamashita is a professor at the Department of Computer Science, College of Information Science and Engineering, Ritsumeikan University. He received his B.E., M.E. and Ph.D. degrees in Information Science from Kyoto University, Kyoto, Japan, in 1993, 1995 and 2001, respectively. His research interest s include new types of computation and logic synthesis for them. He received the 2000 IEEE Circuits and Systems Society Transactions on Computer-Aided Design of Integrated Circuits and Systems Best Paper Award, SASIMI 2010 Best Paper Award, 2010 IPSJ Yamashita SIG Research Award, and 2010 Marubun Academic Achievement Award of the Marubun Research Promotion Foundation. He is a senior member of IEEE and a senior member of IPSJ.

Sudip Roy is currently an associate professor in the department of computer science and engineering, (Indian Institute of Technology (IIT) Roorkee, India after joining in July 2014. He received the B.Sc. degree (honors) in physics and the B.Tech. degree in computer science and engineering from the University of Calcutta, India, in 2001 and 2004, respectively, and the MS (by research) and Ph.D. degrees in computer science and engineering from the IIT Kharagpur, India, in 2009 and 2014, respectively. He has authored one book, one book chapter, two granted U.S. patents and one grated Indian patent including 32 research articles in international peer-reviewed journals and 57 papers in international peer-reviewed conference proceedings. He is a recipient of the Japan Society for the Promotion of Science (JSPS) Invitational Fellowship in the" Long-Term "category in 2021, the Early-Career Research (ECR) Award from Department of Science and Technology (DST), Govt. of India in 2017, and Microsoft Research India PhD Fellowship Award in 2010. He is a member of IEEE and ACM. His current research interests include computer-aided-design for digital systems, optimization techniques, internet of things (IoTs), and application of machine learning and artificial intelligence.

Hiroyuki Tomiyama received his B.E., M.E., and D.E. degrees in Computer Science from Kyushu University in 1994, 1994, and 1999, respectively. He has held positions as a Visiting Researcher at UC Irvine, a Researcher at ISIT/Kyushu, and an Associate Professor at Nagoya University. Since 2010, he has been a Full Professor at the College of Science and Engineering, Ritsumeikan University. In 2024, he assumed the role of Senior Executive Director of the Research Organization of Science and Technology at Ritsumeikan University. He has served on program and organizing committees of several prestigious conferences, including General Chair of MPSoC 2013, Program Chair of RTCSA 2023, and Steering Committee Chair of ATAIT. He has also served as Editor-in-Chief of IPSJ TSLDM, Associate Editor of ACM TODAES, IEEE ESL, and Springer DAEM, and Chair of the IEEE CS Kansai Chapter and IEEE CEDA Japan Chapter. His research interests include design methodologies for embedded and cyber-physical systems.