Demand-Driven Mixture Preparation and Droplet Streaming using Digital Microfluidic Biochips

Sudip Roy[1], Srijan Kumar[1], Partha P. Chakrabarti[1],
Bhargab B. Bhattacharya[2] and Krishnendu Chakrabarty[3]
[1]Indian Institute of Technology Kharagpur, India. {sudipr, ppchak}@cse.iitkgp.ernet.in
[2]Indian Statistical Institute Kolkata, India. bhargab@isical.ac.in
[3]Duke University, Durham, USA. krish@ee.duke.edu

ABSTRACT

In many biochemical protocols, such as polymerase chain reaction, a mixture of fluids in a certain ratio is repeatedly required, and hence a sufficient quantity of the mixture must be supplied for assay completion. Existing sample-preparation algorithms based on digital microfluidics (DMF) emit two target droplets in one pass, and costly multiple passes are required to sustain the emission of the mixture droplet. To alleviate this problem, we design a streaming engine on a DMF biochip, which optimizes droplet emission depending on the demand and available storage. Simulation results show significant reduction in latency and reactant usage for mixture preparation.

1. INTRODUCTION

Automatic mixture preparation, which is a basic preparatory step in many biochemical laboratory protocols, refers to the task of producing a desired ratio of concentration factors (*CF*s) of a set of reactant fluids. In recent years, several techniques have been developed for sample preparation using digital microfluidics to enable biochemistry-on-a-chip [11, 16–18, 23–25]. These solutions have opened up exciting new applications for electronic design automation in health-care and biomedical analysis. Existing dilution and mixing algorithms used in digital microfluidic (DMF) biochips construct a dilution or mixing tree that governs the mix-split sequence needed to produce at most one or two target droplets with a desired ratio of *CF*s.

However, in many real-life applications, a target mixture may be needed repeatedly for successful assay completion. For example, an additional bioassay may be carried out using the same samples or mixtures as confirmatory screening tests on patients in a point-of-care environment. Moreover, an assay may be repeated to ensure higher confidence in the test results. For instance, in the polymerase chain reaction (PCR) used for DNA amplification, a master-mixture of seven fluids (reactant buffer, dNTPs, forward primer, reverse primer, DNA template, optimase and water) is required with a volumetric ratio {10%:8%:0.8%:0.8%:1%:1%:78.4%} [14]. The resultant mixture is next used in several reactions, each requiring a certain amount of master-mix as determined by the assay. In a flow-based lab-on-a-chip, the supply can be met by maintaining the flow for a desired time [9]. In digital microfluidics, the conventional mixing algorithms [5, 11, 16–18, 20, 23–25] emit two target droplets in one pass, and multiple passes are required to sustain the emission of the desired mixture droplet. However, a drawback of this approach is the wastage of expensive reactants and the excessive latency involved in sample preparation. Thus, in DMF biochips, the emission of a multiplicity of target droplets remains as an open problem, thereby highlighting a major drawback of known sample-preparation algorithms.

Roy et al. [20] presented a *dilution engine* for producing multiple droplets of a dilution sample. However, till date, no optimization technique is known that produces a multiplicity of droplets of a target mixture of three or more different fluids. We refer to the problem of automatic emission of a stream of target droplets for a mixture as Multiple Droplets of Single Target (*MDST*).

In this paper, we present two algorithms for implementing *MDST* on a DMF biochip that can produce a stream of target droplets with reduced mix-split operations and usage of input reactant fluids. We use the term *mixture-preparation engine* to refer to a biochip that implements *MDST*, i.e., which is capable of supplying multiple droplets of a mixture M of N fluids with a given ratio, $N \geq 3$. In order to meet a given demand D (the required number of droplets) of M, we introduce the concept of a *mixing forest* for executing the two droplet generation algorithms proposed here. Given a number of on-chip mixer modules, the algorithm *M_Mixer_Schedule* (*MMS*) optimally schedules the mixing forest for implementing the mixing engine. Another algorithm called *Storage_Reduced_Schedule* (*SRS*), can be used to implement the engine when there is a constraint on on-chip storage availability. For validating the proposed algorithms, we have simulated a droplet streaming engine to produce the PCR master-mixture along with scheduling, placement of resources (mixers, reservoirs, storage units), and layout optimization. Simulation results show that, on average, *MMS* can produce multiple target droplets 72.5% faster and with 75% reduction in the reactant usage compared to baseline approaches used earlier for mixture preparation [16, 18, 24, 25].

The remainder of the paper is organized as follows. Section 2 describes related prior work and the problem formulation. The scheduling schemes and design solutions for *MDST* are presented in Section 4. Simulation results are provided in Section 6. Finally, conclusions are drawn in Section 7.

2. RELATED PRIOR WORK

2.1 Mixing Algorithms

In the (1 : 1) mixing model, two unit-volume droplets are mixed and then split to produce two unit-volume droplets; a *mixing tree* is a binary tree representation of (1 : 1) mix-split steps that are necessary to prepare a desired mixture from its constituent fluids on a

DAC '14, June 01 - 05 2014, San Francisco, CA, USA

DMF biochip [24].

Dilution is a special case of mixing of two input fluids (i.e., $N = 2$), one of which is a buffer solution (e.g., distilled water). Recently, several dilution algorithms have been reported in [6, 11, 17, 19, 23, 24] to produce one/two droplet(s) of a single or multiple target concentration factors (CFs). In order to prepare a target mixture M of N fluids with a given ratio of its constituents, where $N \geq 3$, any mixing algorithm [16, 18, 24, 25] can be used. In all these dilution/mixing algorithms, the $(1:1)$ mix-split model is used, which is suitable for DMF biochips. Also, a CF in the target ratio is approximated (by rounding-off) in a scale of 2^d, where the ratio-sum $L = 2^d$. The integer d is user-defined and it determines the accuracy level of target CFs. By constructing a mixing tree of depth d, these algorithms generate the target mixture with a maximum error of $\frac{1}{2^d}$ in CF in each of the constituent fluids.

Based on work published thus far, there are four known algorithms that can be used for mixing N fluids, where, $N \geq 3$ — *MM* [24], *RMA* [18], *RSM* [25] and *MTCS* [16]. By "base mixing tree/graph", it is meant that the task graph on which the basic procedure (*MM*, *RMA*, *RSM* or *MTCS*) is run for producing the target droplets. However, all earlier approaches for mixture preparation can be used to produce at most two target droplets in one pass, and hence, they are unsuitable for droplet streaming, i.e., when multiple droplets of a desired mixture are needed.

The earlier work on dilution ($N = 2$) or mixing ($N \geq 3$) can be classified based on the following objectives of producing a target droplet: (a) *SDST*: single droplet of a single target ratio; (b) *MDST*: multiple (more than two) droplet(s) of a single target ratio; (c) *SDMT*: single droplet for multiple target ratios, each. Table 1 indicates the scope of the proposed work against earlier approaches, where yes/no indicates the applicability of the concerned method.

Table 1: Scope and results of earlier work.

Dilution/mixing Algorithm	*SDST*		***MDST***		*SDMT*	
	$N = 2$	$N > 2$	$N = 2$	$\boldsymbol{N > 2}$	$N = 2$	$N > 2$
[6, 17, 19, 24]	Yes	No	No	**No**	No	No
[5, 11, 23]	Yes	No	No	**No**	Yes	No
[20]	Yes	No	Yes	**No**	No	No
MM [24]	Yes	Yes	No	**No**	No	No
RMA [18]	Yes	Yes	No	**No**	No	No
RSM [25]	Yes	Yes	No	**Yes**	Yes	Yes
MTCS [16]	Yes	Yes	No	**No**	No	No
Proposed Method	Yes	Yes	Yes	**Yes**	No	No

2.2 Mapping a Mixing Tree on a DMF Biochip

Without any loss of generality, we assume that all $(1:1)$ mix-split operations are identical and each of them can be implemented by an on-chip mixer in unit time step. In order to schedule a given mixing tree on a biochip, any earlier algorithm, such as genetic algorithm based scheduling [22], path scheduling [8], or optimal mix scheduling (*OMS*) [13] can be used. In this paper, we have used *OMS* [13] to schedule a mixing tree as it produces an optimum solution for a mixing tree.

3. PROBLEM FORMULATION

The objective of the proposed "mixture-preparation engine" is to implement *MDST*, i.e., to produce multiple (more than two) droplets of a given mixture with reduced latency and reactant wastage. We assume that each of the input fluids is supplied at $CF = 100\%$. Our assumptions of using unit volume droplets and $(1:1)$ mix-split steps are in conformity with current DMF technology [7], and because of the limitations of the availability of only $(1:1)$ mix-split operations, management of waste is important.

In order to implement *MDST*, two naive approaches may be used. If the demand is D ($D > 2$), then the base mixing tree can be repeatedly traversed $\lceil \frac{D}{2} \rceil$ times to produce two target droplets each time. This will increase the total number of mix-split steps (T_{ms}), total number of waste droplets (W), and total number of input droplets (I) by $\lceil \frac{D}{2} \rceil$ times relative to the base mixing tree. Another approach is to increase the supply (number of input droplets or the volume of the input droplets) at every leaf node of the base mixing tree. However, this approach requires an increase in the throughput and area of the on-chip mixer(s) or a complete redesign of the electrode-array layout, which is not feasible in practice.

The design problem of *MDST* with a given target ratio can be formulated as follows. **Inputs**: (a) A set of N different fluids, $X = \{x_1, x_2, \ldots, x_N\}$, $N \geq 2$, each supplied at $CF = 100\%$, (b) a target ratio $a_1 : a_2 : \ldots : a_N$ of N fluids, such that the ratio-sum $L = \sum_{i=1}^{N} a_i = 2^d$, where d is the desired accuracy level in CF, (c) the required number of target droplets, i.e., demand D, and (d) the number of on-chip mixers, M_c. **Output**: A schedule of all $(1:1)$ mix-split steps and on-chip mixer allocation in order to produce D droplets of the target mixture M of N fluids with the specified ratio of its constituents. **Objectives**: (a) Minimize the time of completion (in number of time-cycles), T_c, (b) Minimize the total number of input fluid droplets used, I and (c) Minimize the number of storage units q needed on-chip.

4. MULTIPLE DROPLET ENGINE: *MDST*

The key idea behind the mixture-preparation engine *MDST* is to utilize the waste droplets produced by the baseline mixing algorithms [16, 18, 24, 25].

In general, *RMA* [18] constructs a base mixing tree with a larger number of waste droplets compared to other mixing algorithms (*MM*, *RSM*, *MTCS*). Thus, an engine based on *RMA* is likely to produce a steam of target droplets more efficiently with the best utilization of reactant fluids.

4.1 Mixing Forest for the *MDST* Engine

Given a target ratio with accuracy level d in CF, the base mixing tree ($\mathcal{T}_1$) of depth d can be constructed by any existing mixing algorithm. The root of the base mixing tree $\mathcal{T}_1$ represents the target droplet. We first introduce the concept of a "mixing forest" ($\mathcal{F}$) to represent the mix-split task graph, when the demand D of the target droplets is more than two ($2 < D \leq 2^d$). The forest $\mathcal{F}$ is constructed with several component mixing trees, each of which is determined by the waste droplets produced from level $(d-1)$ to level 1 in the base mixing tree. Each time a component mixing tree is included in the mixing forest, we get two more target droplets denoted by its root. Hence, by executing a mixing forest generated from a mixing tree of depth d, a total of 2^d target droplets can be produced. For a demand $D > 2^d$, this same process of including new component mixing trees into the mixing forest is repeated from $\mathcal{T}_1$ till the total demand D is met. On the other hand, with $D = p.2^d$ ($p \in \mathbb{Z}^+$), the same mixing forest for $D = 2^d$ is repeated p times in the final mixing forest. Thus, for a given demand D, the mixing forest $\mathcal{F}$ contains $\lceil \frac{D}{2} \rceil$ component mixing trees ordered as $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{|\mathcal{F}|}$, where $|\mathcal{F}| = \lceil \frac{D}{2} \rceil$. Note that for a demand $D = p.2^d$ ($p \in \mathbb{Z}^+$), all the intermediate droplets are fully utilized, and the total number of waste droplets (W) becomes zero.

As an example, consider the PCR master-mix ratio {10%:8%:0.8%:0.8%:1%:1%:78.4%}. This ratio can be approximated as {2:1:1:1:1:1:9} in a scale of 16 (where the accuracy level, $d = 4$), and the droplet demand $D = 2^d$ for this target mixture. A hierarchical view of the construction of the mixing forest is shown in Fig. 1. The input droplets (leaf nodes), the target

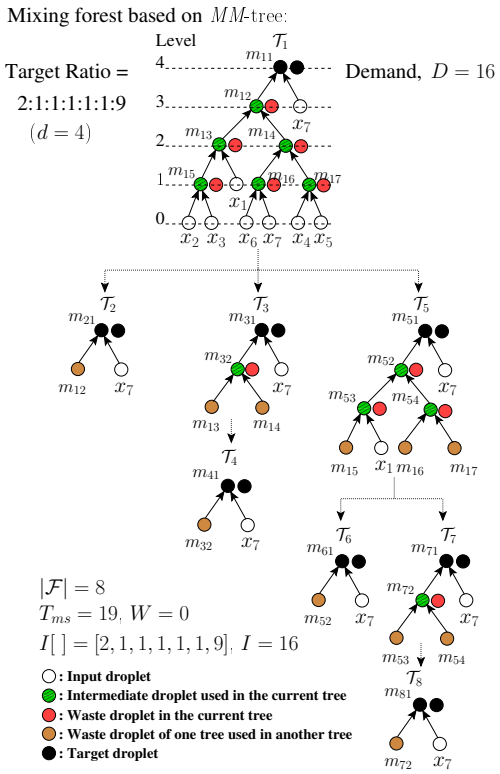


Figure 1: Construction of the mixing forest from the base mixing tree obtained by *MM* [24] for a target ratio 2:1:1:1:1:1:9 (i.e., $d = 4$) with demand $D = 16$.

droplets (root nodes), and two intermediate droplets generated after a mix-split step (non-leaf node) are shown with different colors. Note that out of the two intermediate droplets produced at each interior node, one is used in the current tree (marked in green) and the other one is the waste droplet (marked in red). A brown-colored node indicates the use of an earlier waste droplet in another tree. The color legends are explained in Fig. 1. For a component mixing tree $\mathcal{T}_i$, each non-leaf node is marked by a label m_{ij} denoting the mix-split step, where j is the level obtained in *breadth-first traversal* ordering of $\mathcal{T}_i$ from left to right (see Fig. 1).

In order to satisfy the precedence relation in the underlying mixing process, when a new component mixing tree $\mathcal{T}_i$ is included. Figure 2 shows an example of a mixing forest that produces 20 target droplets of the same mixture. In a mixing forest, let T_{ms} be the total number of mix-split steps (i.e., non-leaf nodes in a mixing forest) and W be the total number of waste droplets. From a mixing forest, the total number of droplets required for each type of input fluid can be computed by adding their numbers over all the component mixing trees. Let $I[\,]$ be the array representing the required numbers of input droplets and I be the total number of input droplets required to implement the mixing forest.

If d is the depth of the base mixing tree $\mathcal{T}_1$, then the following properties of a mixing forest $\mathcal{F}$ hold: (a) the root nodes of all the component mixing trees in $\mathcal{F}$ are at the same level, i.e., at level d; (b) the depth of each of the component mixing trees in $\mathcal{F}$ is less than or equal to d.

4.2 Mapping of *MDST* Mixing Forest

To implement the *MDST* engine on a DMF biochip efficiently, we need to determine an optimum schedule $\mathcal{S}$ of the mixing forest $\mathcal{F}$. In other words, the assignment of time-cycle and allocation of on-chip mixers to each of the mix-split (non-leaf) nodes of the forest are required. In a mixing forest $\mathcal{F}$, if m_{ij} is the j^{th} mix-split step (according to the breadth-first traversal order from left to right) of the i^{th} component mixing tree $\mathcal{T}_i$, then $m_{ij} \longmapsto M_k^t$ indicates that the mix-split step m_{ij} is to be executed in mixer M_k at time-cycle t. For *MDST*, both the time of completion (T_c) and the number of required storage units (q) depend on the target ratio, the accuracy level d, the baseline algorithm used in constructing the mixing tree, and the demand D. Given these parameters, the constructed mixing forest is unique and hence the values of T_{ms}, W and I are also unique. However, depending on the number of available on-chip mixers M_c and the scheduling scheme, the mixing forest can take different values of T_c and q.

Algorithm 1 *M_Mixers_Schedule* ($\mathcal{F}$, d, M_c)

```
Initialize level ℓ = 1 and time-cycle t = 1.
Initialize queue Q = ∅ and schedule S = ∅.
while ℓ ≤ d do
    Enqueue all new schedulable nodes of F ordered from level ℓ upwards to Q.
    Dequeue M_c nodes from Q.
    for each dequeued node n do
        Assign mixer in increasing order of indices (say M_k) to n at time-cycle t and
        S = S ⋃ {n ⟼ M_k^t}.
    ℓ = ℓ + 1 and t = t + 1.
while Q ≠ ∅ do
    Dequeue M_c nodes from Q.
    for each dequeued node n do
        Assign mixer in increasing order of indices (say M_k) to n at time-cycle t and
        S = S ⋃ {n ⟼ M_k^t}.
    t = t + 1.
return Time of completion T_c = t.
```

Let t_c be the time of completion (required number of clock-cycles), when a base mixing tree is scheduled by *OMS* [13] with M_c on-chip mixers. Note that a baseline mixing tree produces two target droplets of a mixture in one pass. Hence, to meet a droplet demand D, we need to run this procedure $\lceil \frac{D}{2} \rceil$ times. Therefore, for a baseline approach, the total number of clock-cycles (T_r), total number of waste droplets (W_r), and total number of input droplets (I_r), will increase $\lceil \frac{D}{2} \rceil$-fold compared to those of the mixing tree. We consider three baseline approaches such as "Repeated *MM*" or *RMM*, "Repeated *RMA*" (*RRMA*) and "Repeated *MTCS*" (*RMTCS*), where the base mixing trees are determined by *RMA* [18] and *MTCS* [16], respectively, and they are scheduled by *OMS* [13] with same number of mixers (i.e., M_{lb} of the corresponding mixing tree obtained by *MM* [24]). In this paper, we have compared our droplet streaming engine with these three baseline approaches, namely *RMM*, *RRMA* and *RMTCS*. Following a similar argument as in [13], we can show that the number of storage units (q_r) required to schedule the mixing forest using M_c on-chip mixers is $d - (\log_2 M_c + 1)$. We denote the total number of storage units required by any baseline approach as q_r. Hence, the savings in time of completion, waste and input reactant by the proposed scheme with reference to a baseline approach can be computed as $\Delta T_c = T_r - T_c$, $\Delta W = W_r - W$ and $\Delta I = I_r - I$, respectively. The additional number of storage units required can be estimated as $\Delta q = |q_r - q|$.

4.2.1 *Scheduling of a Mixing Forest*

Due to precedence among the non-leaf nodes in a mixing forest, a mix/split step can be completed only if those represented by its two children are already executed or available as input. If the number of on-chip mixers M_c is less than the minimum number (lower-bound) of mixers required for the fastest completion of a mixing forest $\mathcal{F}$, then the time of completion (T_c) increases. We present a scheduling algorithm *M_Mixers_Schedule* (*MMS*), given by **Algorithm 1**, to schedule a mixing forest $\mathcal{F}$ with M_c mixers.

We call the set of mix-split (non-leaf) nodes of a mixing forest

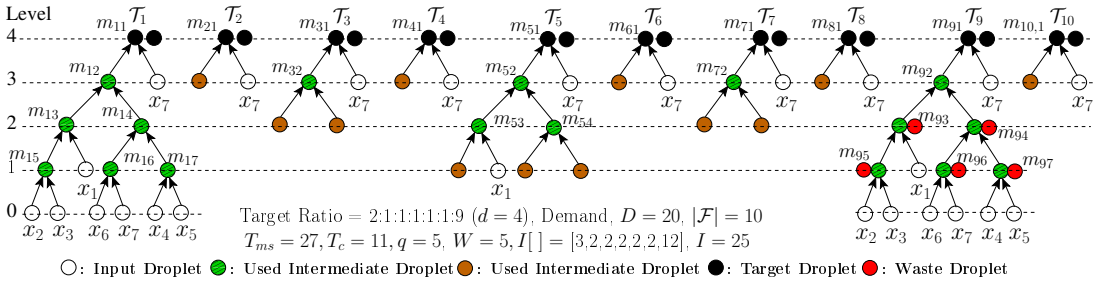


Figure 2: Mixing forest using *MM* [24] for *MDST* ($D = 20$) of the PCR master-mix with target ratio 2:1:1:1:1:1:9.

Algorithm 2 *Storage_Reduced_Scheduling* ($\mathcal{F}$, d, M_c)

```
Initialize level ℓ = 1 and time-cycle t = 1.
Initialize two priority queues, Q_int = ∅ and Q_leaf = ∅.
Initialize schedule S = ∅.
while ℓ ≤ d OR any node is NOT scheduled do
    for each schedulable node n do
        if n is a parent of two leaf nodes then
            Insert node n in Q_leaf according to priority value.
        else
            Insert node n in Q_int according to priority value.
    intNodes = |Q_int|.
    Dequeue min(M_c, |Q_int|) nodes from Q_int.
    Dequeue max(0, min(M_c − intNodes, |Q_leaf|)) nodes from Q_leaf.
    for each dequeued node n do
        Assign mixer in increasing order of indices (say M_k) to n at time-cycle t and
        S = S ⋃ {n ⟼ M_k'}.
    ℓ = ℓ + 1 and t = t + 1.
return Time of completion T_c = t.
```

Algorithm 3 *Counting_Storage_Units* ($\mathcal{F}$, $\mathcal{S}$, T_c)

```
Initialize an array Storage[1...T_c] to all 0s.
for each non-leaf node n in F do
    t_n = scheduled time-cycle of n.
    t_p = scheduled time-cycle of the parent node p of n.
    for i = t_n + 1; i < t_p; i = i + 1 do
        Storage[i] = Storage[i] + 1.
Total number of required storage units, q = max_i Storage[i].
```

as the *schedulable nodes*, if they were not scheduled earlier and are now ready to be performed at the same time-cycle. The schedulable nodes can be processed concurrently, if a sufficient number of mixers are available at that time-cycle. We maintain a queue Q to enqueue all the schedulable nodes in a level-wise bottom-up fashion, and M_c or fewer nodes are dequeued from Q to assign the available mixers at time-cycle t. When all the levels are examined and the queue Q is still non-empty, the remaining nodes are dequeued to assign mixers in next time-cycles without enqueuing any new nodes. In **Algorithm 1**, mixers can be allocated to the dequeued nodes (in the increasing order of indices of the mixers) at time-cycle t.

4.2.2 Storage-Reduced Scheduling of a Forest

Next, we propose a heuristic scheme to reduce storage requirements while scheduling a mixing forest. It prioritizes the scheduling of non-leaf nodes in the mixing forest based on two factors: if the mixing is stalled at a node, then (i) how it affects the total storage requirement, and (ii) what is its impact on the time of completion of the mixing forest.

The first factor is the impact on the storage requirement. In a mixing forest, a schedulable node may be of three types. It may be an internal (non-leaf) node, of which (a) both the children are internal nodes (Type-A), (b) only one (left or right) child is a leaf node (Type-B), or (c) both the children are leaf nodes (Type-C). If a node of Type-A is stalled, it would cost two storage units, if a node of Type-B is stalled, it would cost one storage unit, and if a node of Type-C is stalled, it would cost no storage units, per time-cycle. This is because a leaf node indicates direct input from the fluid reservoir and it does not require any on-chip storage unit. Hence, the priority should be given to the internal nodes whose at least one child node is an internal node (Type-A or Type-B), over the internal nodes whose both the children nodes are leaf nodes (Type-C).

The second factor is the impact on the time of completion. While considering nodes of Type-C, it is more efficient to perform a mix-split node at a lower level, rather than that at a higher level, since a mixing operation at a higher level will not be able to proceed, until the execution of its sibling is performed. Similarly, while considering nodes of Type-A or Type-B, it is better to execute a mix-split node at a higher level than a similar node of a lower level, because this will lead to earlier completion of the mixing forest.

We use two priority queues, Q_{int} and Q_{leaf}, to maintain these priorities of schedulable nodes. The priority queue Q_{int} stores the schedulable nodes of Type-A and Type-B, whereas Q_{leaf} stores the schedulable nodes of Type-C. Within these priority queues, the insertion of nodes is managed according to the priority rule described above. This scheduling scheme is referred to as *Storage_Reduced_Scheduling* (*SRS*) and the pseudo-code is written in **Algorithm 2**. In *SRS*, while scheduling the nodes with M_c mixers, M_c nodes are dequeued from the queue Q_{in} first, then from Q_{leaf}. Both the queues have all the schedulable nodes. Hence, if the number of available schedulable nodes in the queue Q_{in} and Q_{leaf} is less than M_c, then some mixers remain idle. Therefore, for the same problem instance of *MDST*, *SRS* may take longer time than *MMS*. However, *SRS* required less number of storage units than *MMS*.

5. PCR MASTER-MIX ENGINE

Consider the task of producing multiple droplets for the PCR master-mix used in DNA amplification [14]; the details of this mixture are given in Section 4.1. In Fig. 3, the base mixing tree obtained by *MM* [24] is shown as $\mathcal{T}_1$, for which the minimum number of mixer for earliest completion (M_{lb}) is three. For a demand $D = 20$, the mixing forest is scheduled by *SRS* (Fig. 3), in which each non-leaf node is assigned with the time-cycle (denoted by an integer in 'italics') and with a mixer among three available mixers (M_1, M_2 and M_3). A modified Gantt chart of this schedule is shown in Fig. 4, which indicates the droplet emission sequence with the increasing time-cycles. The total number of required storage units can be obtained from the schedule using **Algorithm 3**.

We implement the PCR chip following a design technique presented in [21], as shown in Fig. 5. It requires seven fluid-reservoirs to hold seven reactants (reservoir R_i is loaded with fluid x_i), two waste-reservoirs (W_1 and W_2), three on-chip mixers (M_1, M_2 and M_3) and five storage units ($q_1, \ldots, q_5$). The relative positions of reservoirs and mixers are optimized considering the total droplet-transportation cost. The matrix shown in Fig. 5 indicates the

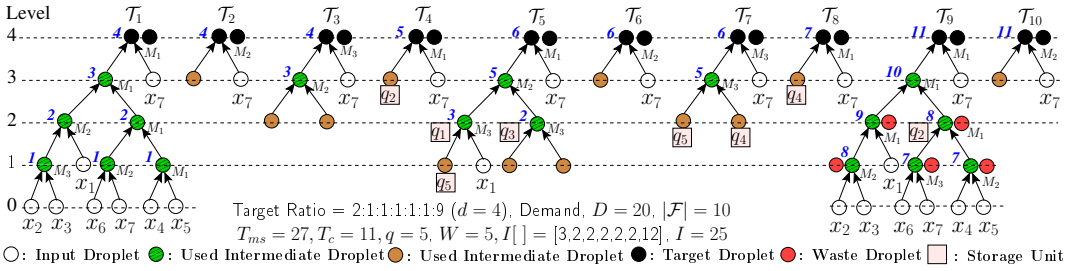
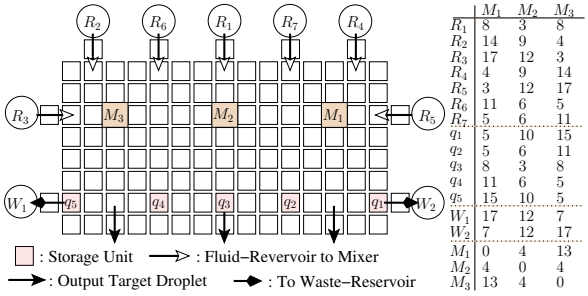


Figure 3: Mixing forest obtained by *MM* [24] for the target ratio 2:1:1:1:1:1:9 with $D = 20$ and scheduled by *SRS* with three on-chip mixers M_1, M_2 and M_3.

t	1	2	3	4	5	6	7	8	9	10	11
M_1	m_{17}	m_{14}	m_{12}	m_{11}	m_{41}	m_{51}	m_{81}	m_{94}	m_{93} q_2	m_{92}	m_{91}
M_2	m_{16}	m_{13}	m_{32}	m_{21} q_2	m_{52}	m_{61}	m_{97}	m_{95}	—	—	$m_{10,1}$
M_3	m_{15}	m_{54} q_5	m_{53} q_3 q_4	m_{31} q_3 q_4 q_1 q_2	m_{72}	m_{71} q_1	m_{96}	—	—	—	—

$T_c = 11$ time cycles, $q = 5$, $D = 20$, $W = 5$

Figure 4: Gantt chart showing scheduling of the mixing forest (Fig. 3), storage requirement, and emission sequence of target and waste droplets.



	M_1	M_2	M_3
R_1	8	3	8
R_2	14	9	4
R_3	17	12	3
R_4	4	9	14
R_5	3	12	17
R_6	11	6	5
R_7	5	6	11
q_1	5	10	15
q_2	5	6	11
q_3	8	3	8
q_4	11	6	5
q_5	15	10	5
W_1	17	12	7
W_2	7	12	17
M_1	0	4	13
M_2	4	0	4
M_3	13	4	0

Figure 5: An example layout for PCR Master-Mix preparation; the matrix shows the droplet-transportation costs (in terms of the number of electrodes used).

droplet-transportation costs (in terms of number of electrodes used) among the on-chip modules. For the mixing forest shown in Fig. 3, the total droplet-transportation cost can be computed by adding the costs for the transportation of all the droplets. It is observed that if the respective mixing forest scheduled by *SRS* is implemented on the layout of Fig. 5, a total of 386 electrodes needs to be actuated. On the other hand, repeated application of the baseline mixing algorithm *MM* [24] will require as many as 980 electrode actuations for the same demand of target droplets. Excessive electrode actuation leads to reliability problems and reduced lifetime for biochips [10]. Hence, with the proposed optimization method for *MDST*, the implementation of the PCR engine can be handled efficiently.

6. SIMULATION RESULTS

We evaluate the performance of our emission engine on several target ratios used in real-life bioprotocols with demand $D = 32$. We assume that all the ratios are approximated in a scale of 256; Ex.1: {26:21:2:2:3:3:199}, the PCR master-mix used for DNA amplification [3, 14]; Ex.2: {128:123:5}, a mixture of phenol, chloroform and isoamylalcohol used in the One-Step Miniprep method [4]; Ex.3: {25:5:5:5:5:13:13:25:1:159}, a mixture of 10 fluids is used in the Molecular Barcodes methods [12]; Ex.4: {9:17:26:9:195}, a mixture of five fluids used in the Splinkerette PCR method [1]; Ex.5: {57:28:6:6:6:3:150}, a mixture used in the Miniprep protocol [15]. The comparative results of the two scheduling schemes for *MDST* are shown in Table 2. For a basis of comparison, we have considered the repeated applications of three baseline algorithms (*MM* [24], *RMA* [18], *MTCS* [16]), which are named as *RMM*, *RRMA*, *RMTCS*, respectively.

We carry out simulations with a large set of different target ratios to evaluate the scheduling schemes. In a typical real-life bioassay, as many as 12 different fluids may need to be mixed in a target mixture [2]. Our test data set consists of 6058 synthetic target ratios of N ($2 \le N \le 12$) different fluids with ratio-sum, $L = 32$. In order to determine the base mixing tree for a target ratio, we consider each of the three existing mixing algorithms such as *MM* [24], *RMA* [18] and *MTCS* [16]. For *MDST* with a given ratio, this base mixing tree can be scheduled using the schemes *MMS* or *SRS*. The base mixing tree is optimally scheduled by the scheme *OMS* [13] with M_{lb} mixers of the corresponding *MM*-tree (i.e., with the minimum number of mixers needed for fastest execution).

We study the variations in average time of completion (T_c) and average total input requirement (I) by varying the demand D from 2 to 32. The corresponding results shown in Fig. 6 demonstrate that the use of *MMS* can save a significant volume of reactant fluids compared to the baseline mixing algorithms. The completion time can also be significantly reduced, when a certain number of storage units (as determined by the scheduler) is available.

As summarized in Table 3, *MMS* can on average reduce the time of completion (T_c) and total input reactant fluid requirements (I) by 72.5% and 75%, respectively, over any baseline mixing algorithm by using at most 30 on-chip storage units. On the other hand, for the same savings in I, on average, *SRS* can reduce the required number of storage units (q) by 25.5%, if we slightly compromise the production time (4.6% more) compared to *MMS* running the same mixing algorithm.

Simulation results reveal that when *RMA* [18] is used as the baseline algorithm, the scheduling schemes (*MMS* or *SRS*) save significant amounts of reactant fluids and produce multiple target droplets at a faster rate compared to other schemes. This can be verified from the results shown in Table 3 and Fig. 7, where the variations of T_c and q with the number of on-chip mixers M are shown for the

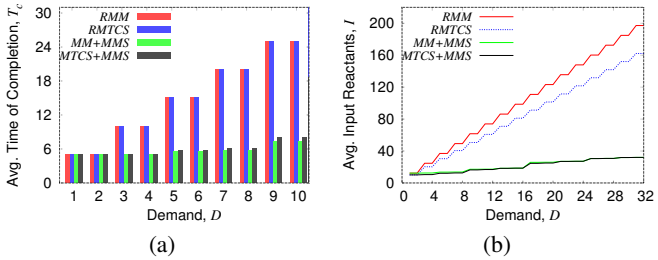


Figure 6: Variations of average (a) T_c and (b) I, with varying demand D over 6058 target ratios (where $L = 32$) of N different fluids ($2 \le N \le 12$).

Table 2: Comparison of T_c, q and I for *MDST* with a Combination of Three Scheduling Schemes and Three Mixing Algorithms*.

Ratio	# Clock Cycles, T_c (Time of Completion)									# Storage Units Required, q									# Reactant (Input) Droplets, I					
	A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***A***	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***I***	***A***	***B/C***	***D***	***E/F***	***G***	***H/I***
Ex.1	128	15	16	128	12	12	128	15	16	1	13	8	0	12	8	2	13	8	272	41	304	43	240	39
Ex.2	128	34	34	128	34	34	128	34	34	0	15	4	0	15	4	0	15	4	144	35	144	35	144	35
Ex.3	128	12	13	128	12	14	128	11	13	1	9	9	0	10	9	2	10	11	432	45	464	47	288	39
Ex.4	128	20	20	128	15	15	128	20	20	1	13	6	0	12	8	1	13	8	208	37	256	40	160	37
Ex.5	128	17	17	128	17	19	128	24	24	2	13	9	1	12	13	1	13	14	304	40	320	41	208	36

**A*: *RMM*, *B*: *MM+MMS*, *C*: *MM+SRS*, *D*: *RRMA*, *E*: *RMA+MMS*, *F*: *RMA+SRS*, *G*: *RMTCS*, *H*: *MTCS+MMS*, *I*: *MTCS+SRS*.

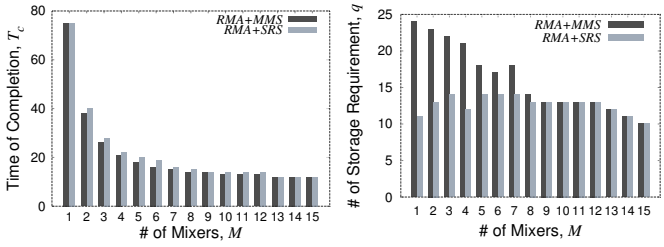
Table 3: Average % Improvements in T_c, I and q over 6058 Target Ratios of $L = 32$.

Parameter	Relative Schemes	Average % Improvement		
		MM [24]	*RMA* [18]	*MTCS* [16]
Time of Completion (in # Time Cycles), T_c	*MMS*‖*R*	73.0%	73.5%	71.1%
	SRS‖*R*	72.0%	72.1%	69.8%
Total Input Requirements (# Reactant Droplets), I	*MMS*‖*R*	76.0%	76.6%	72.4%
	SRS‖*R*	76.0%	76.6%	72.4%
# Storage Units, q	*SRS*‖*MMS*	23.2%	26.0%	27.4%
Time of Completion, T_c	*SRS*‖*MMS*	−3.9%	−5.5%	−4.4%

MMS‖*R*: *MMS* over *Repeated*, *SRS*‖*R*: *SRS* over *Repeated* and *SRS*‖*MMS*: *SRS* over *MMS*.

Table 4: Results for PCR Master-Mix using Three On-Chip Mixers and a Fixed Number of Storage Units.

D	# Passes; (Total # Time Cycles, Total # Waste Droplets)								
	$d = 4$			$d = 5$			$d = 6$		
	$q = 3$	$q = 5$	$q = 7$	$q = 3$	$q = 5$	$q = 7$	$q = 3$	$q = 5$	$q = 7$
2	One (4,6)	One (4,6)	One (4,6)	One (5,9)	One (5,9)	One (5,9)	One (6,9)	One (6,9)	One (6,9)
16	Two (10,7)	One (7,0)	One (7,0)	Two (12,13)	One (8,3)	One (8,3)	Two (13,14)	One (9,5)	One (9,5)
20	Two (11,5)	One (11,5)	One (11,5)	Two (13,11)	Two (13,11)	One (11,5)	Two (14,13)	One (10,6)	One (10,6)
32	Three (17,7)	One (14,0)	One (14,0)	Three (20,16)	Two (16,6)	Two (18,10)	Three (21,19)	Two (17,12)	Two (17,12)



(a) (b)

Figure 7: Variations of (a) T_c and (b) q with M by *MMS* and *SRS* for the PCR master-mix ratio {2:1:1:1:1:1:9} with $D = 32$.

PCR master-mix with $D = 32$.

In real-life, the number of available on-chip storage is often limited. In order to execute our algorithms with limited resources, several passes may be needed. Let D' be the maximum demand that is achievable in one pass of *SRS* given the number of available on-chip storage units (q'). If $D > D'$, then this pass can be repeated $\lfloor \frac{D}{D'} \rfloor$ times, where in the last pass an incomplete mixing forest is to be scheduled for producing $\left(D - D'.\lfloor \frac{D}{D'} \rfloor\right)$ target droplets. This technique enables us to design a droplet-streaming engine, which is able to fulfill a given demand satisfying the constraint of available on-chip storage electrodes. In our experiment for producing PCR master-mixture on the layout of Fig. 5, we varied the number of storage units (q as 3, 5, and 7), accuracy level (d as 4, 5, and 6) and demand (D as 2, 16, 20, and 32). The results on the number of passes, the time of completion (in total number of time cycles), and the total number of waste droplets are reported in Table 4.

7. CONCLUSIONS

We have presented the first optimization techniques for producing a stream of mixture droplets on a DMF biochip subject to the availability of a given number of on-chip mixers and storage electrodes. Two algorithms have been described for efficient implementation of the underlying mixing forest on a DMF biochip. The key idea is to suitably reuse the intermediate waste droplets, and thereby producing more target droplets, instead of discarding them as waste. Simulation results highlight the role that electronic design automation techniques can play in emerging biotechnology and biomedical applications.

References

[1] A. G. Uren et al. A High-Throughput Splinkerette-PCR Method for the Isolation and Sequencing of Retroviral Insertion Sites. *Nature Protocols*, 4(5):789–798, 2009.

[2] V. Ananthanarayanan and W. Thies. Biocoder: A Programming Language for Standardizing and Automating Biology Protocols. *Journal of Biological Engineering*, 4, 2010.

[3] Bio-Protocol, 2013. http://www.bio-protocol.org/.

[4] K. Chowdhury. One step 'miniprep' method for the isolation of plasmid DNA. *Nucleic Acids Res.*, 19(10):2792, 1991.

[5] D. Mitra et al. On-Chip Sample Preparation with Multiple Dilutions Using Digital Microfluidics. In *Proc. of IEEE ISVLSI*, pages 314–319, 2012.

[6] E. J. Griffith et al. Performance Characterization of a Reconfigurable Planar-Array Digital Microfluidic System. *IEEE TCAD*, 25(2):345–357, 2006.

[7] R. B. Fair. Digital Microfluidics: Is a True Lab-on-a-Chip Possible? *Microfluidics and Nanofluidics*, 3:245–281, 2007.

[8] D. Grissom and P. Brisk. Path scheduling on digital microfluidic biochips. In *Proc. of the IEEE/ACM DAC*, pages 26–35, 2012.

[9] K. E. Herold and A. Rasooly. *Lab-on-a-Chip Technology (Vol. 1): Fabrication and Microfluidics*, volume I. Caister Academic Press, August 2009.

[10] T.-W. Huang, T.-Y. Ho, and K. Chakrabarty. Reliability-Oriented Broadcast Electrode-Addressing for Pin-Constrained Digital Microfluidic Biochips. In *Proc. of IEEE/ACM ICCAD*, pages 448–455, 2011.

[11] J.-D. Huang et al. Reactant Minimization during Sample Preparation on Digital Microfluidic Biochips using Skewed Mixing Trees. In *Proc. of IEEE/ACM ICCAD*, pages 377–384, 2012.

[12] I. Lopez and D. L. Erickson. *DNA Barcodes: Methods and Protocols*. Humana Press, 2012.

[13] L. Luo and S. Akella. Optimal Scheduling of Biochemical Analyses on Digital Microfluidic Systems. *IEEE TASE*, 8(1):216–227, 2011.

[14] PCR Master Mix Calculator. `http://www.mutationdiscovery.com`.

[15] Preparation of Plasmid DNA by Alkaline Lysis with SDS: Minipreparation, Cold Spring Harb Protocols. `http://cshprotocols.cshlp.org/content/2006/1/pdb.prot4084.citation`, 2006.

[16] S. Kumar et al. Efficient Mixture Preparation on Digital Microfluidic Biochips. In *IEEE DDECS*, pages 205–210, 2013.

[17] S. Roy et al. Optimization of Dilution and Mixing of Biochemical Samples using Digital Microfluidic Biochips. *IEEE TCAD*, 29(11):1696–1708, 2010.

[18] S. Roy et al. Layout-Aware Solution Preparation for Biochemical Analysis on a Digital Microfluidic Biochip. In *Proc. of the VLSID*, pages 171–176, 2011.

[19] S. Roy et al. Waste-Aware Dilution and Mixing of Biochemical Samples with Digital Microfluidic Biochips. In *Proc. of IEEE/ACM DATE*, pages 1059–1064, 2011.

[20] S. Roy et al. A High-Throughput Dilution Engine for Sample Preparation on Digital Microfluidic Biochips. *IET Computers & Digital Techniques (IET-CDT)*, page 9, September 2013.

[21] S. Roy et al. Routing-Aware Resource Allocation for Mixture Preparation in Digital Microfluidic Biochips. In *IEEE ISVLSI*, 2013.

[22] F. Su and K. Chakrabarty. High-level Synthesis of Digital Microfluidic Biochips. *ACM JETC*, 3(4):1–32, 2008.

[23] T.-W. Chiang et al. Graph-Based Optimal Reactant Minimization for Sample Preparation on Digital Microfluidic Biochips. In *Proc. of IEEE VLSI-DAT*, pages 1–4, 2013.

[24] W. Thies et al. Abstraction Layers for Scalable Microfluidic Biocomputing. *Natural Computing*, 7(2):255–275, 2008.

[25] Y.-L. Hsieh et al. A Reagent-Saving Mixing Algorithm for Preparing Multiple-Target Biochemical Samples Using Digital Microfluidics. *IEEE TCAD*, 31(11):1656–1669, Nov 2012.