High-throughput dilution engine for sample preparation on digital microfluidic biochips

Sudip Roy¹, Bhargab B. Bhattacharya², Sarmistha Ghoshal³, Krishnendu Chakrabarty⁴

¹Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India
²Nanotechnology Research Triangle, Indian Statistical Institute, Kolkata, India
³School of Materials Science and Engineering, Bengal Engineering and Science University, Howrah, India
⁴Department of Electrical and Computer Engineering, Duke University, Durham, USA

E-mail: sudipr@cse.iitkgp.ernet.in

Abstract: The new generation of digital microfluidic biochips is capable of implementing complex biochemical laboratory assays (bioprotocols) on a tiny device for automatic and reliable analysis of fluid samples. Dilution of a sample fluid is one of the basic steps required in almost all bioprotocols. The authors design a dilution engine for sample preparation that can produce a stream of target droplets with a desired concentration factor by reusing the waste droplets. Also, a scheduling scheme is presented for mapping the dilution steps into the dilution engine. Next, an architecture is proposed to implement the dilution engine using only one (1:1) mix-split microfluidic module (mixer). The authors consider two schemes: (i) the base approach, where no on-chip storage unit (additional electrode) is available, and (ii) when a constant number of storage units is provided. Simulation results show that the second scheme saves the expensive reactants most efficiently, produces a stream of target droplets very fast and is highly suitable for supplying the required number of droplets as needed.

1 Introduction

A digital microfluidic (DMF) biochip can implement [1] a wide range of biochemical laboratory assays (bioprotocols) on an electrode array of a few square centimetres in size with low consumption of sample/reagent fluids and with high sensitivity [2, 3]. In almost all bioassays, reactants are needed with several concentration factors (CF). Sample preparation steps, such as dilution and mixing of biofluids, are automated and integrated on-chip for high-throughput applications of DMF biochips [4–13]. Existing dilution (mixing) algorithms construct a dilution (mixing) tree which governs the mix-split sequence needed to produce one/two target droplet(s) of desired CF. The input fluids are assumed to have supplied with a sample of CF = 100% and a buffer solution with CF = 0%.

In many applications, a stream of target droplets with a desired CF may be required. For example, (a) a bioassay may need to be repeated on samples for screening several patients in a point-of-care diagnostic environment, (b) an assay may require repeated execution to enhance reliability of test results and (c) the same reagent with a given target CF is needed for conducting various biochemical tests. However, till date, no optimisation technique has been reported that is suitable for catering to multiple demand of a target droplet with a given CF. This problem of automatic and efficient emission of a stream of target droplets is referred to as multiple target droplet generation (MTDG). Assume that the target CF is approximated (rounded-off) as a d-bit binary fraction. Hence, in (1:1) mixing model, the target droplets can be generated with a maximum error of \(1/2^{d+1}\) in the desired CF by following a sequence of at most \(d\) mix-split steps; where \(d \in \mathbb{Z}^+\) and is referred to as the desired accuracy level in CF. To determine the sequence of (1:1) mix-split steps that generates the first two target droplets, any existing dilution algorithm such as twoWayMix [6] can be used. It may be noted that all the earlier methods [5–8] used for dilution were designed for generating only one or two target droplets. In order to produce multiple target droplets, the dilution/mixing tree obtained by twoWayMix can be executed repeatedly, each time producing two such droplets. However, repeated execution of these algorithms not only produces a huge number of waste droplets, but also takes a long time to fulfil multiple demand of a target CF. In this paper, we present, for the first time, an efficient algorithm and a scheduling scheme for solving the MTDG problem with or without any available on-chip storage units.

We use the term dilution engine to represent a biochip that is capable of supplying multiple droplets with a given CF. Here, we present two architectural designs of DMF biochips and the relevant scheduling schemes for implementing the proposed dilution engine – (a) a simple layout with one (1:1) mix-split microfluidic module (mixer) and no storage unit, and (b) a layout with one mixer and a constant number \((d−1)\) of on-chip storage units. Simulation results show that the second scheduling scheme is more suitable for implementing a low-cost dilution engine that reduces the use of reactants as well as the number of electrode actuations and production time.
The organisation of remainder of the paper is as follows: Section 2 describes the related previous work. In Section 3, the dilution mechanism of MTDG is characterised and two architectural layouts are presented. The resource scheduling schemes are presented in Section 4. Sections 5 and 6 provide discussions and simulation results. Finally, conclusions are drawn in Section 7.

2 Automatic sample dilution: prior work and motivation

In this paper, we consider the (1:1) mixing model, in which a unit-volume droplet \( x_1 \) of CF = \( C_1 \) is mixed with another unit-volume droplet \( x_2 \) of CF = \( C_2 \), followed by a balanced split operation. As a result, two unit-volume droplets of CF = \( (C_1 + C_2)/2 \) each, are produced (see Fig. 1a). One (1:1) mix operation and a subsequent balanced split are together referred to as a mix-split or a dilution step. The two droplets thus produced may be used in the next step or one of them may be discarded as a waste droplet. In order to dilute a fluid to a desired CF \( C_t \) with an accuracy level \( d \), dilution steps are to be executed sequentially at most \( d \) times, and this sequence of steps is represented as a binary tree of depth \( d \) [14]. In the case of dilution, a sample fluid of CF = 1, and a buffer solution with CF = 0 are used, and the corresponding binary tree is called a dilution tree. For example, a dilution tree of depth 5 that produces two target droplets of CF = 34.375% \( \simeq (11/32) \) from a supply of sample (indicated as ‘violet’ circles) and buffer (indicated as ‘blue’ circles) is shown in Fig. 1b. Here, the ‘brown’, ‘green’ and ‘red’ circles indicate the intermediate, target and waste droplets, respectively. Let \( W \) be the total number of waste droplets generated while executing a dilution tree.

\[
C_t = 34.375\% \simeq \frac{11}{32} (\equiv 0.010112)
\]

For automatic dilution of a biofluid, several algorithms were proposed in the literature [5–8, 10–13]. Griffith et al. [5] first proposed a dilution algorithm GAG of \( O(d^3) \) time complexity, to determine a \( d \)-length sequence of mix-split steps for producing two target droplets, given an error tolerance of \( (1/2^d + 1) \) in the desired CF. It used a binary search strategy to determine the required dilution steps; the sequence of mix-split steps was represented in the form of a directed graph. Thies et al. [6] presented a dilution algorithm twoWayMix of \( O(d) \) time complexity to determine a dilution tree of depth \( d \) which produces two target droplets. If the target CF \( C_t \) is approximated (rounded-off) as a \( d \)-bit binary fraction, then the desired target can be achieved with an error in CF not exceeding \( (1/2^d + 1) \). The method does not require to store any intermediate droplets, and only the current droplet is mixed with either the sample or the buffer in the next dilution step. For example, to produce two target droplets of CF = \( (11/32) 0.010112 \) from the supply of sample (CF = 100%) and buffer (CF = 0%) in five steps, twoWayMix scans the five-bit binary fraction 0.010112 from the right end and constructs a dilution tree with \( d = 5 \) and \( W = 4 \) as shown in Fig. 1b. Another method known as dilution/mixing with reduced wastage (DMRW) [7] used a binary-search strategy to determine the dilution graph. It was based on \( (k:k) \) mixing model and reused some of the waste droplets (where \( k \geq 1 \)). Although DMRW significantly reduces \( W \) compared to twoWayMix, it needs a rotary mixer for efficient implementation. Later an improved dilution/mixing algorithm was reported [8] for further reduction of wastage. Other earlier algorithms considered the problem of producing target droplets of different CFs and focused on reducing the number of mix-split steps, reactant usage or wastage [10–13].

For on-chip dilution, all the earlier works are aimed at producing only two target droplets either in minimum time or with reduced wastage. However, design of a dilution engine (MTDG) that can supply a stream of \( D (D > 2) \) droplets with a given CF efficiently has not been studied before. As a naive approach, the dilution tree corresponding to the given target CF can be repeatedly executed \( [D/2] \) times in order to produce \( D \) droplets. However, the total number of required mix-split steps and waste droplets will be multiplied by a factor of \( [D/2] \). Both of these two parameters should be minimised in order to save time and expensive reagents in a bioassay. This motivates us to propose an architectural layout and a scheduling scheme for designing a low-cost and high-throughput dilution engine.

3 Multiple target droplet generation (MTDG)

In this section, we characterise the dilution process for MTDG and present two different schemes for implementing it.

3.1 Characterisation of dilution process for MTDG

We use two types of binary trees and define some notations to characterise the dilution process of MTDG. A full binary tree is a binary tree in which every node other than the leaf nodes has two children [14], whereas a binary tree of depth \( d \) containing exactly \( 2^d \) leaf nodes (with \( 2^d – 1 \) non-leaf nodes) is called a perfect binary tree [14]. In both full and perfect binary trees, the root is assumed at depth 0 or level \( d \).

A dilution tree, \( M(C_t, d) \), is a full binary tree of depth \( d \) in which every non-leaf nodes with in-degree two and out-degree two, represents a (1:1) mix-split step and the two leaf nodes at level 0 denote two target droplets, where the leaf nodes at any other level indicate the waste droplets. Hence, each non-leaf node of a dilution tree has two
Assume that a target CF $C_t$ with an accuracy level of $d$ is to be produced; hence the depth of the dilution tree will be $d$. Let the total number of target droplets required (demand) be $D$.

A complete dilution tree, $C(C_t, d)$, is a perfect binary tree of depth $d$ and each of its $2^d$ leaf nodes denotes a target droplet. For $C_t = (11/32)$, Fig. 2 shows the $C(11/32, 5)$ whose leaf nodes at level 0 represent 32 target droplets. All the non-leaf nodes of a $C$ are labelled by positive integers in pre-order fashion [14] starting from the root node labelled by 1, and let the pre-order labelling of a non-leaf node be denoted as POL($i$). For the tree $C$ shown in Fig. 2, the POL value at the rightmost node at level 1 is 31. In a $C(C_t, d)$, all the $2^d$ leaf nodes at level 0 are labelled by increasing integers from 1 (left-most leaf node) to $2^d$ (right-most leaf node) in a left-to-right order. Such labelling of leaf nodes at level 0 is referred to as demand labelling, which denotes the target droplets demanded (see Fig. 2). Depending on the demand $D$, a subtree with leaf nodes at level 0 denoting the required target droplets is constructed from $C$.

A demand subtree, $D(D, d)$, is a subtree of $C(d)$, in which all the leaf nodes are at level 0, which are labelled as 1, 2, 3, ..., $D$ ($1 \leq D \leq 2d$). If $D = 2^d$, the D-subtree is identical to $C (d)$ and the number of waste droplets ($W$) becomes zero. The green-shaded part of $C$ in Fig. 2 shows $D(18,5)$ for a demand of 18 target droplets.

A demand-waste subtree, $DW(D, d)$, is a minimum size full binary tree and a subtree of $C(C_t, d)$, which fully includes a demand subtree $D(D, d)$. In $DW(D, d)$, D leaf nodes at level 0 denote the target droplets and the leaf nodes at higher levels represent $W$ waste droplets. If $D$ is even, then there is no waste droplet at level 0 and all the waste droplets are at higher levels, otherwise there is only one waste droplet at level 0 along with others at higher levels. For example, $DW(18,5)$ is shown in Fig. 2 as a blue-shaded region.

A waste forest, $WF(D, d)$, is a forest of several perfect binary subtrees of $C(d)$, whose roots are the leaf nodes in $DW(D, d)$ corresponding to the waste droplets in the tree $C(d)$, while producing $D$ targets. Thus, a $DW(D, d)$ consists of $W$ full binary trees with roots at different levels. For example, in Fig. 2, $WF(18,5)$ is indicated within the region outlined in red. All the nodes (leaves and non-leaves except the roots) of the component trees in a waste forest are referred to as virtual nodes.

Let $BS[x]$ represent the bit-sum (total number of 1’s) of the binary representation of $x$. We now present the following results:

**Lemma 1:** The proposed method will produce $D \leq 2^d$ target droplets with a maximum error of $(1/2^{d+1})$ in CF, and the number of waste droplets will be given by $W = BS[2^d - D]$.

**Proof:** The production of target droplets with the desired error bound is achieved by the execution of a dilution tree of depth $d$. For a demand $D$, the number of virtual droplets (leaf nodes) in the complete dilution tree $C$ that are not needed is $(2^d - D)$. If $D = 2^d$, then $(2^d - D) = 0$, and with demand $D$, $D \leq 2^d$, the total number of mix-split operations Mix($D, d$) needed to produce $D$ target droplets, $D \leq 2^d$, is given by

$$Mix(D, d) = D + W - 1 \quad (1)$$

If $D$ is an integral multiple of $2^d$, Mix($D, d$) = $[D/2^d] \cdot (2^d - 1)$. Hence, for any $D > 2^i$, we can verify that Mix($D, d$) = $[D/2^i] \cdot (2^i - 1) + (D' + W - 1)$, where $D' = D - 2^i \cdot [D/2^i]$.

**Theorem 1:** For any target CF $C_t$ and maximum error $\leq (1/2^{d+1})$, and with demand $D$, $D \leq 2^d$, the total number of mix-split operations Mix($D, d$), is given by Mix($D, d$) = $D + BS[2^d - D] - 1$.

**Proof:** The proof immediately follows from Lemma 1 and Observation 1.

**Theorem 2:** In $DW(D, d)$, if $i$ is the parent node of the leaf node with demand label $D \leq 2^d$, the total number of mix-split operations needed to generate $D$ is given by Mix($D, d$) = POL($i$).

**Proof:** Clearly, the node $i$, whose child has demand label $D$, must appear in level 1 of $DW$ and the leaf nodes of $DW$ denoting the target droplets are labelled from left-to-right with the demand labelling. The rest of the proof follows immediately from the properties of pre-order labelling of non-leaf nodes in a tree.

**Theorem 3:** The pre-order labelling of $DW$ gives the minimum number of mix-split operations Mix$_{min}$($D, d$) required to produce $D$ target droplets with an accuracy level of $d$; thus, Mix$_{min}$($D, d$) = POL($i$) = $D + BS[2^d - D] - 1$ under dilution tree-based protocol.

**Proof:** This proof comes from the construction of a subtree $DW$ from $C(C_t, d)$ with an accuracy level of $d$. The subtree $DW$ is constructed by selecting the first $D$ leaf nodes at level 0 in $C$. As proven earlier in Lemma 1, $DW$ has $W$ leaf nodes at different levels higher than or equal to level 0 denoting the waste droplets, where $W = BS[2^d - D]$. A full binary tree with $(D + W)$ leaf nodes has $(D + W - 1)$
non-leaf nodes. Hence, the pre-order labelling scheme of non-leaf nodes in \( \mathcal{C} \) and the construction of a \( \mathcal{DW} \) indicate that the number of required mix-split operations is the POL value of the non-leaf node \( i \), which is the parent node of the leaf node labelled as \( D \), and this number is minimum under this dilution protocol. Hence, \( \text{Mix}_{\text{min}}(D) = \text{POL}(i) = D + \text{BS}[2^d - D] - 1 \) is the highest POL of the non-leaf nodes in a \( \mathcal{DW} \).

For the example shown in Fig. 2, the number of waste droplets is \( W = \text{BS}[2^3 - 18] = 3 \), and the required number of mix-split operations is \( \text{Mix}(18, 5) = 18 + 3 = 21 \), which is minimum.

Depending on the values of \( d \) and \( D \), the complete process of dilution for MTDG is captured by \( \mathcal{C}(C, d) \) or/and \( \mathcal{DW}(D, d) \), which are referred to as protocol trees. Three cases may arise.

Case 1: If \( 1 \leq D < 2^d \), only one \( \mathcal{DW}(D, d) \) subtree is considered as the protocol tree.

Case 2: If \( D = 2^d \), the tree \( \mathcal{C}(C, d) \) is considered as the protocol tree.

Case 3: If \( D > 2^d \), then \( \lceil D/2^d \rceil \) copies of \( \mathcal{C}(C, d) \) tree and one \( \mathcal{DW}(D', d) \) subtree, where \( D' = D - 2^d \cdot \lfloor D/2^d \rfloor \), are considered as protocol trees.

Thus, given an instance, we need to schedule the protocol trees appropriately in order to implement the dilution engine.

To produce \( D \) target droplets, first, the protocol trees are determined for the desired CF \( C_i \) with an accuracy level of \( d \). Next, we schedule the protocol trees to execute them on any storage unit. The pseudo-code for SMNS is shown in Algorithm 1 (see Fig. 5).

In SMNS, the dilution tree \( \mathcal{M} \) obtained by twoWayMix [6] is used to produce two target droplets after every \( d \) clock cycles (see Fig. 4a). To produce \( D \) \( (D > 2) \) target droplets, the same dilution tree \( \mathcal{M} \) is repeatedly executed \( \lfloor D/2 \rfloor \) times. Hence, SMNS replicates the execution of twoWayMix [6] without any storage unit. The pseudo-code for SMNS is shown in Algorithm 1 (see Fig. 5).

For this scheme, each execution of \( \mathcal{M} \) of depth \( d \) generates \( (d - 1) \) waste droplets, if \( D \) is even; otherwise the second target droplet is also discarded as a waste droplet. In each execution of \( \mathcal{M} \), the total number of input droplets is \( (d + 1) \), of which the required number of sample droplets (with CF = \( C_i \)) is the bit-sum of the \( d \)-bit binary fraction representing the target CF \( C_i \) and that of buffer droplets (with CF = \( C_b \)) is \( (d - b + 1) \). Hence, for SMNS scheme the closed-form

\[
\text{Mix}_{\text{min}}(D) = C_i + BS[2^d - D] - 1
\]

for the example shown in Fig. 2, the number of waste droplets is \( W = BS[2^3 - 18] = 3 \), and the required number of mix-split operations is \( \text{Mix}(18, 5) = 18 + 3 = 21 \), which is minimum.

Depending on the values of \( d \) and \( D \), the complete process of dilution for MTDG is captured by \( \mathcal{C}(C, d) \) or/and \( \mathcal{DW}(D, d) \), which are referred to as protocol trees. Three cases may arise.

Case 1: If \( 1 \leq D < 2^d \), only one \( \mathcal{DW}(D, d) \) subtree is considered as the protocol tree.

Case 2: If \( D = 2^d \), the tree \( \mathcal{C}(C, d) \) is considered as the protocol tree.

Case 3: If \( D > 2^d \), then \( \lceil D/2^d \rceil \) copies of \( \mathcal{C}(C, d) \) tree and one \( \mathcal{DW}(D', d) \) subtree, where \( D' = D - 2^d \cdot \lfloor D/2^d \rfloor \), are considered as protocol trees.

Thus, given an instance, we need to schedule the protocol trees appropriately in order to implement the dilution engine.

To produce \( D \) target droplets, first, the protocol trees are determined for the desired CF \( C_i \) with an accuracy level of \( d \). Next, we schedule the protocol trees to execute them on the chip. We propose two scheduling schemes, SMNS and SMWS, for the layouts shown in Figs. 3a and 3b, respectively. Assume that while scheduling, one mix-split (dilution) step takes one clock cycle, and the time for droplet transportation from a reservoir (storage unit) to the mixer (or reverse) is negligible compared to that for a dilution step [2].

A schedule \( S \) is the set of entries, each of which indicates mixer and timing assignments to a non-leaf node of the protocol tree. For MTDG, we define the performance parameters of a scheduling scheme in terms of \( C_i \), \( d \) and \( D \).

The performance parameters are the total number of clock cycles required \( (T) \), the number of waste droplets generated \( (W) \), the number of sample (with CF = \( C_i \)) droplets \( (I_k) \), and the number of required buffer (with CF = \( C_b \)) droplets \( (I_b) \). The throughput of a scheduling scheme, \( \Theta \), is defined as the average number of target droplets produced per clock cycle computed by \( \Theta = (D/T) \). We define the speed-up of SMWS over SMNS, \( \Sigma \), as the ratio of the total clock cycles required in SMNS \( (T_{\text{SMNS}}) \) to that in SMWS \( (T_{\text{SMWS}}) \). In other words, \( \Sigma = (T_{\text{SMWS}}/T_{\text{SMNS}}) \).

Single mixer with no storage scheme (SMNS)

In SMNS, the dilution tree \( \mathcal{M} \) obtained by twoWayMix [6] is used to produce two target droplets after every \( d \) clock cycles (see Fig. 4a). To produce \( D \) \( (D > 2) \) target droplets, the same dilution tree \( \mathcal{M} \) is repeatedly executed \( \lfloor D/2 \rfloor \) times. Hence, SMNS replicates the execution of twoWayMix [6] without any storage unit. The pseudo-code for SMNS is shown in Algorithm 1 (see Fig. 5).

For this scheme, each execution of \( \mathcal{M} \) of depth \( d \) generates \( (d - 1) \) waste droplets, if \( D \) is even; otherwise the second target droplet is also discarded as a waste droplet. In each execution of \( \mathcal{M} \), the total number of input droplets is \( (d + 1) \), of which the required number of sample droplets (with CF = \( C_i \)) is the bit-sum of the \( d \)-bit binary fraction representing the target CF \( C_i \) and that of buffer droplets (with CF = \( C_b \)) is \( (d - b + 1) \). Hence, for SMNS scheme the closed-form
expressions for the performance parameters will be given by

\[ T = \left\lfloor \frac{d}{2} \right\rfloor, \quad W = (d - 1) \left\lfloor \frac{D}{2} \right\rfloor + (D \mod 2) \]

(2)

(3)

where \( C_t \) is approximated as \((c/2^d)\) and \( c \in \mathbb{Z}^+ \). Therefore, the throughput of SMNS can be estimated as \( \Theta_{\text{SMNS}} = (D/T) \).

### 4.2 Single mixer with storage scheme (SMWS)

In SMWS, first \( d \) mix-split steps of a protocol tree are executed sequentially (corresponding to the non-leaf nodes with POL value from 1 to \( d \)), thus producing the first two target droplets (see Fig. 4b). Then, it utilises intermediate droplets of earlier steps, which were saved on the storage units as shown in Fig. 3b. The pseudo-code for SMWS is shown in Algorithm 2 (see Fig. 6). Execution of Algorithm 2 (see Fig. 6) requires the procedure ScheduleOneMixer to be called (see Fig. 7) Algorithm 3.

The SMWS scheme uses a stack data structure (works like a last-in-first-out or LIFO fashion) \( U \) (shown in Fig. 4b). The physical storage units are maintained and managed like a stack on the supporting layout, and are used to store the intermediate droplets, whenever needed. The stack \( U \) supports two operations: push(\( i \)) (pushes a node \( i \) of protocol tree into the stack) and pop(U) (pops out the last node pushed into it). The total number of storage units \( q \) required in the SMWS scheme depends on the size of stack and the accuracy level \( d \) of the target CF. We will show later that in order to produce multiple target droplets with accuracy level of \( d \), we require a layout with \( q = d - 1 \). For example, when \( d = 10 \), a layout with nine storage units denoted by \( U_1, U_2, \ldots, U_9 \) is required as shown in Fig. 3b.

SMWS utilises the intermediate droplets generated in the protocol tree in a bottom-up fashion to produce two new target droplets. A storage unit which becomes free can be used again to store the intermediate droplets produced later. As SMWS reuses the intermediate droplets, it reduces the number of waste droplets significantly. This, in turn, reduces the consumption of expensive reactants (sample and buffer solutions).

**Theorem 4:** In order to produce \( D \) target droplets with accuracy level of \( d \) in the desired CF, the maximum number of storage units required by SMWS is \((d-1)\).

**Proof:** Let \( 2^{d-1} < D \leq 2^d \); in this case, the first two target droplets are produced after \( d \) mix-split steps and \((d-1)\) intermediate droplets are stored. Each of these stored droplets is associated with a non-leaf node of \( DW \) at level \( i \), where \( 1 \leq i \leq (d-1) \). Therefore, to store the intermediate
Bit(throughput of SMWS is estimated as $\Theta$)

is to be executed to produce $2$ requires only $(i)$ any stored droplet in a protocol tree is $(i)$ number of storage units required is $(i)$ we need to process the stored droplet at depth $(i)$

be written as follows

$x_t = 2$ target droplets, the $x_t$ mix-split steps $\text{Mix}(i)$

the number of required clock cycles $T$ is equal to the total number of mix-split steps $\text{Mix}(D, d)$ in the protocol trees (given by Theorem 1); the total number of waste droplets $W$ is given by Lemma 1. It may be noted that, for MTDG with a target CF $C_t$, the number of input droplets used at level $i$ of the protocol tree is $[D/2^i]$ for $i = 1$ to $d$. From the structure of a protocol tree, it is apparent that in order to produce $D$ target droplets, the first mix-split step at level $d$ is to be executed $[D/2^d]$ times. Therefore, for SMWS, the closed-form expressions for the performance parameters can be written as follows

\begin{align}
T &= \text{Mix}(D, d) = (2^d - 1) \left( \frac{D}{2^d} \right) + M_p \tag{6}
\end{align}

\begin{align}
W &= \text{BS}[\left(2^d - D'\right) \mod 2^d] \tag{7}
\end{align}

\begin{align}
I_k = \sum_{i=1}^{d} \left( \frac{D}{2^i} \right) \cdot \text{Bit}(C_t, i) \tag{8}
\end{align}

\begin{align}
I_x = \sum_{i=1}^{d} \left( \left\lfloor \frac{D}{2^i} \right\rfloor \cdot \text{Bit}(C_t, i) \right) + \left( \frac{D}{2^d} \right) \tag{9}
\end{align}

\begin{align}
q = (d - 1), \quad \text{if } D \geq 2^d \tag{10}
\end{align}

\begin{align}
q = 3 - \left\lfloor \log_2 D \right\rfloor - 1, \quad \text{if } 3 \leq D \leq 2^d \tag{11}
\end{align}

\begin{align}
q = 0, \quad \text{if } D = 1 \quad \text{or} \quad 2 \tag{12}
\end{align}

where $M_p = D' + W - 1$, if $D' \neq 0$, or $0$, otherwise, where $D' = D - 2^d \cdot \left\lfloor D/2^d \right\rfloor$. Bit($C_t$, $i$) is the $i$th bit (counted from the least-significant-bit or LSB) of $d$-bit binary representation of $C_t$ and Bit($C_t$, $i$) is the complement of Bit($C_t$, $i$). The throughput of SMWS is estimated as $\Theta_{\text{SMWS}} = (D/T)$ and its speed-up over SMNS is $\Sigma_{\text{SMWS}} = (T_{\text{SMNS}}/T_{\text{SMWS}})$. As $T_{\text{SMWS}}$ is less than $T_{\text{SMNS}}$, $\Theta_{\text{SMWS}} > \Theta_{\text{SMNS}}$.

## 5 Discussions

### 5.1 Gantt charts and emission spectra for MTDG

Comparative results for the above-mentioned two scheduling schemes can be visualised with a variant of Gantt charts [16], as shown in Fig. 8. In this figure, we consider an example for MTDG which is required to produce $D = 11$ target droplets of $C(11/32)$ with an accuracy level of $d = 5$. In order to produce $11$ target droplets, SMNS takes $5.11/2$ or $30$ clock cycles and generates $(5 - 1).11/2 + (11 \mod 2) = (4.6) + 1$ or $25$ waste droplets (Fig. 8a). The SMWS scheme, on the other hand, requires $11 + \text{BS}_{[32 - 11]} = 13$ clock cycles and generates BS$_{[32 - 11]mod32} = \text{BS}_{[10101]} = 3$ waste droplets (Fig. 8b).

We can demonstrate the process of target droplet generation with an emission spectrum against time, as shown in Fig. 9. To generate $20$ target droplets of a desired CF with $d = 5$, SMNS takes $50$ clock cycles, since after every fifth clock cycle two target droplets are produced (see Fig. 9a), whereas SMWS needs only $21$ clock cycles (see Fig. 9b).

### 5.2 SMWS with fewer storage units

For a given $d$ and $D$, the number of storage units (say, $q'$) is less than $q$ (i.e., $d - 1$, when $D \geq 2d$ or $(\log 2) - 1$, when $1 \leq D < 2^d$), then the demand-waste subtree(s) can be constructed depending on the values of $d$, $D$ and $q'$. If $q' \geq (d - 1)$, then the same earlier scheduling scheme SMWS is used. To schedule the protocol trees with a bounded number of

### Algorithm 3

1. Schedule $S = \emptyset$; Time $t = 1$; # of target droplets $i = 0$;
2. Node $n \leftarrow T.\text{root}$; Stack $U = \emptyset$;
3. while $i < D$ do
4. while $n \neq \text{leaf node}$ do
5. $n.\text{mizer} = M$; // $M$ is the mixer
6. $n.\text{time} = t$; // timing assignment
7. $S = S \cup \{n.\text{mizer}, n.\text{time}\}$;
8. $t = t + 1$;
9. if $n.\text{right} \neq \text{leaf node}$ then
10. $U = \text{push}(n.\text{right})$;
11. if $n.\text{right} = \text{leaf node}$ at Level $\neq 0$ then
12. Discard node $n.\text{right}$ as waste droplet;
13. $n \leftarrow n.\text{left}$;
14. $i = i + 2$;
15. $n \leftarrow \text{pop}(U)$;
16. Return $S$;

---

**Fig. 7** Algorithm 3: ScheduleOneMixer(Protocol Tree $T$)

---

Fig. 8 Gantt charts for MTDG of $D = 11$ target droplets of a CF with accuracy level $d = 5$

Mixer assignment shows the POL-values of non-leaf nodes assigned to $M$ along with the timing assignments by two scheduling schemes

- a SMNS
- b SMWS

---

**Fig. 9** Gantt charts for MTDG with $D = 11$ target droplets of a CF with accuracy level $d = 5$

- a SMNS
- b SMWS

---

© The Institution of Engineering and Technology 2013

doi: 10.1049/iet-cdt.2013.0060
storage units, we propose a modified scheduling scheme SMWS\_StorageBound, given by Algorithm 4 (see Fig. 10). Thus, if we have fewer than \((d - 1)\) storage units, for an accuracy level \(d\), more number of protocol trees are to be scheduled by SMWS\_StorageBound. Execution of Algorithm 4 (see Fig. 10) requires the procedure ScheduleOneMixer to be called (Algorithm 3, see Fig. 8). As a result, the total time required for MTDG and the total number of waste droplets generated while producing \(D\) target droplets with \(q'\) on-chip storage units \((q' < q)\) will be more compared to the case when \((d - 1)\) storage units are available.

5.2.1 Suitability of the proposed layout for SMWS: The proposed layout fits well with the execution of the scheduling algorithm. The design is simple and various microfluidic modules (dispensers, mixer, storage units) are so placed that the routing paths of input, output, intermediate and waste droplets are short and congestion conflicts can be resolved easily. As a result, the total time of completion for MTDG can be significantly reduced.

### 6 Simulation results

In our simulation experiments, we have considered at most 10 mix-split steps in a dilution tree (i.e. \(d = 10\)), as this provides enough accuracy level in concentration factor. Also, the volumetric error of the intermediate droplets may increase with increasing \(d\) [4]. For several examples of MTDG, the performance parameter values \(T, W, q\) for both the scheduling schemes SMNS [6] and SMWS are shown in Table 1. It is evident that SMWS completes the same MTDG process much faster than SMNS and hence \(\Theta_{\text{SMWS}} > \Theta_{\text{SMNS}}\). Note that the average speed-up of SMWS over SMNS increases with the increasing values of \(d\).

We have simulated both the scheduling schemes SMNS and SMWS and observed the variations of \(T, W, I_h\) and \(I_\ell\) for various values of \(d, D\) and \(C_t\). First, we present the plots for a target CF with an accuracy level of \(d = 4\). Figs. 11a and b show the variations in \(T\) and \(W\), respectively, with increasing demand \((D)\) of target droplets. For both SMNS

---

**Algorithm 4**

1: Construct \(C\) from \(M \leftarrow \text{twoWayMix}(C_L, C_R, C_T, d);\)
2: Obtain \(POL(i)\) of each non-leaf node and demand labelling of all the leaf nodes at level 0 in \(C;\)
3: if \(D \leq 2^{d+1}\) then
4: Obtain \(DW(D, d)\) from \(C;\)
5: ScheduleOneMixer\((DW(D, d))\);
6: else if \(D > 2^{d+1}\) then
7: for \(i = 1\) to \(\lfloor \frac{D}{2^{d+1}} \rfloor\) do
8: Obtain \(DW(2^{d+1}, d)\) from \(C;\)
9: ScheduleOneMixer\((DW(2^{d+1}, d))\);
10: \(D' = D - 2^{d+1} \lfloor \frac{D}{2^{d+1}} \rfloor;\)
11: if \(D' \neq 0\) then
12: Obtain \(DW(D', d)\) from \(C;\)
13: ScheduleOneMixer\((DW(D', d))\);

---

**Table 1** Comparative results for some example cases of MTDG

<table>
<thead>
<tr>
<th>Example</th>
<th>(d)</th>
<th>(D)</th>
<th>SMNS</th>
<th>(\Theta)</th>
<th>SMWS</th>
<th>(\Theta)</th>
<th>(\Sigma)</th>
<th>(q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>32</td>
<td>80</td>
<td>64</td>
<td>0.40</td>
<td>31</td>
<td>0</td>
<td>1.03</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>35</td>
<td>90</td>
<td>73</td>
<td>0.39</td>
<td>37</td>
<td>4</td>
<td>0.95</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>11</td>
<td>30</td>
<td>25</td>
<td>0.37</td>
<td>13</td>
<td>3</td>
<td>0.85</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>64</td>
<td>192</td>
<td>160</td>
<td>0.33</td>
<td>63</td>
<td>0</td>
<td>1.02</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>68</td>
<td>204</td>
<td>170</td>
<td>0.33</td>
<td>70</td>
<td>4</td>
<td>0.97</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>33</td>
<td>102</td>
<td>86</td>
<td>0.32</td>
<td>37</td>
<td>5</td>
<td>0.89</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>256</td>
<td>1024</td>
<td>896</td>
<td>0.25</td>
<td>255</td>
<td>0</td>
<td>1.00</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>300</td>
<td>1200</td>
<td>1050</td>
<td>0.25</td>
<td>302</td>
<td>4</td>
<td>0.99</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>17</td>
<td>72</td>
<td>64</td>
<td>0.24</td>
<td>23</td>
<td>7</td>
<td>0.74</td>
</tr>
</tbody>
</table>

---


doi: 10.1049/iet-cdt.2013.0060

© The Institution of Engineering and Technology 2013
and SMWS, the variations in $T$ and $W$ against $D$ and $C_t$ are demonstrated in the 3D plots shown in Figs. 12a and 12b, respectively. It is observed that the values of $T$ and $W$ are independent of $C_t$, and depend only on $d$ and $D$. Figs. 13a and 13b show the variations in $I_h$ and $I_\ell$, respectively, by SMNS and SMWS for five different target CFs with $D = 2$, 10 and 100.

We have performed simulation experiments of SMNS and SMWS with varying $d$ and observed that as $d$ increases, the improvement of SMWS becomes more significant compared to that of SMNS. For a given value of $d$, SMWS requires less time to produce $D$ target droplets and consumes lesser amount of expensive reactants compared to those needed by SMNS.

7 Conclusions

In this paper, we have presented an efficient scheme for producing a stream of target droplets with a desired concentration factor. Our design requires a single mixer with only a constant number of storage electrodes, which is determined by the accuracy level of the concentration factor, and is independent of the demand of the target droplet. We have also formulated a scheduling scheme for controlling the dilution sequence. Several theoretical results related to performance analysis have been derived for the two scheduling schemes, SMNS and SMWS. Our results demonstrate that SMWS scheme outperforms the base approach of SMNS significantly, both in mixing time.
(hence, in actuation steps) and in reactant usage, when production of multiple target droplet is required. Thus, SMWS will be highly suitable for designing a low-cost and high-throughput dilution engine for sample preparation with DMF biochips. As an open problem, one may study how to produce, efficiently, multiple droplets of a sample, which is a mixture of more than two reagents on a DMF platform.

8 Acknowledgment

The authors wish to thank the Guest Editor, Dr. Rishad Shafik, and the anonymous reviewers for their critical comments and valuable suggestions. This work of S. Roy was supported by Microsoft Corporation and Microsoft Research India under the Microsoft Research India PhD Fellowship Award (2010–2014). The work of B. B. Bhattacharya was supported by a special grant to Nanotechnology Research Triangle from Indian Statistical Institute, Kolkata. The work of S. Ghoshal was supported in part by the Department of Science & Technology (DST) Women Scientists Scheme (WOS-A), 2009–2012. The work of K. Chakrabarty was supported in part by the US National Science Foundation under grants CCF-0914895 and CNS-1135853.

9 References

14 Aho, A.V., Hopcroft, J.E., Ullman, J.D.: ‘The design and analysis of computer algorithms’ (Addison-Wesley, 1974)