Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

Optimal Scheduling of Biochemical Analyses on Digital Microfluidic Systems

Lingzhi Luo and Srinivas Akella

Abstract— **Digital microfluidic systems (DMFS) are an emerging class of lab-on-a-chip systems that manipulate individual droplets of chemicals on a planar array of electrodes. The biochemical analyses are performed by repeatedly moving, mixing, and splitting droplets on the electrodes. In this paper, we focus on minimizing the completion time of biochemical analyses by exploiting the parallelism among the operations. We consider a binary tree representation of chemical analyses to schedule operations. Using pipelining, we overlap mixing operations with input and transportation operations. We find the lower bound of the mixing completion time according to the tree structure of given reactions, and calculate the minimal number of mixers *S* required to achieve the lower bound. We present a scheduling algorithm for the case with a specified number of mixers no more than *S*, and prove it is optimal to minimize the mixing completion time. We also analyze resource constraint issues for two extreme cases. For the case with one mixer, we prove that all schedules result in the same mixing completion time as long as the mixer is kept busy at all times and then design a scheduling algorithm to minimize the number of storage units. For the case with zero storage units, we find the minimum number of mixers required. Finally, we demonstrate the benefits of our scheduling methods on an example of DNA polymerase chain reaction (PCR) analysis.**

I. Introduction

Low-cost, portable lab-on-a-chip systems capable of rapid automated biochemical analysis can impact a wide variety of applications including biological research, genetic analysis, point-of-care diagnostics, and biochemical sensing [1]–[4]. *Digital microfluidic systems* (DMFS) are an emerging class of lab-on-a-chip systems that manipulate discrete droplets. A digital microfluidic system manipulates individual droplets of chemicals on a planar array of electrodes by using electrowetting (or dielectrophoresis). The chemical analysis is performed by repeatedly moving, mixing, and splitting droplets on the electrodes.

An important advantage of DMFS devices is their reconfigurability and flexibility in performing various biochemical analyses. We focus on microfluidic systems that manipulate droplets by electrowetting [5]. Droplets are nanoliters in volume, and have been moved at 12-25 cm/s on planar arrays of 0.15 cm wide electrodes [6], [7]. The ability to control discrete droplets on a planar array enables complex analysis operations to be performed in DMFS devices (Fig. 1). For simple biochemical analysis operations, no special purpose devices are required aside from the array itself. The array may additionally contain cells that can perform specialized operations, such as heating or optical sensing.

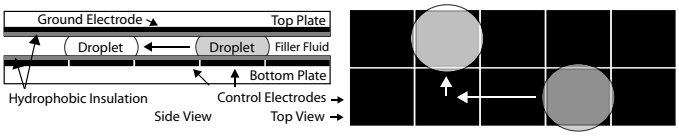

Fig. 1. Droplets on an electrowetting array (side and top views). The droplets are in a medium (usually oil or air) between two glass plates. The gray and white droplets represent the same droplet in initial and destination positions. A droplet moves to a neighboring electrode when that electrode is activated; the electrode is turned off when the droplet has completed its motion. Based on [8].

The completion time of biochemical analyses in batch mode is the time required to produce one droplet of final product. Our goal is to minimize the completion time in batch mode under resource constraints on DMFS. The completion time of a reaction depends on the reaction, the provided resources, and the algorithm to perform it. In this paper, we exploit the tree structure of biochemical analysis to do scheduling and minimize the completion time. A full binary tree structure is introduced to model the biochemical reactions. First, using pipelining we overlap the mixing operations with input and transportation operations. Then we focus on scheduling of mixing operations to minimize the mixing completion time. According to the tree structure, the lower bound of a given reaction is calculated. Also we calculate the minimal number of mixers required to achieve minimal completion time, and present a greedy scheduling algorithm to minimize the completion time under mixer constraint. With the above algorithm, parallelism is fully exploited not only among different mixing steps but also among different kinds of operations. Also, we consider two extreme cases: with one mixer and with zero storage units. In the first case, we prove all active scheduling result in the same completion time, calculate the minimum number of storage units for the given analysis and design corresponding scheduling algorithm. In the second case we compute the minimum number of mixers required.

II. Related work

Design Automation: Su and Chakrabarty presented architecture-level synthesis and geometry synthesis of biochips [9], [10]. They used acyclic sequence graphs to represent the reactions and developed techniques in operation scheduling, resource binding, and module placement.

Scheduling Algorithms: Scheduling algorithms optimize the system performance by properly allocating tasks to

L. Luo is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA luol2@cs.rpi.edu

S. Akella is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA sakella@cs.rpi.edu

resource. Kwok and Ahmad studied the static scheduling of a program on a multiprocessor system to minimize the program completion time in parallel processing [11]. Since the general problem is NP-complete, they compared 27 heuristic scheduling algorithms. In contrast to the general problem in parallel computing, the multiple-task reactions in DMFS have certain kinds of precedence, which can be represented by a full binary tree. Ding et al., Su and Chakrabarty represent the DMFS reactions using data-flow directed graph and consider the scheduling using Integer Linear Programming (ILP) [9], [12]. They solve the problem by general ILP solvers and heuristic algorithms without exploiting the structure of DMFS reactions.

Routing: Böhringer modeled the routing problem in DMFS as a multi-robot cooperation problem and used a prioritized A^* search algorithm to generate the optimal plan for droplets [13]. Griffith and Akella presented a general-purpose DMFS and designed routing algorithm based on Dijkstra's algorithm [14], [15]. Su, Hwang and Chakrabarty proposed a two-stage routing method to minimize the number of electrodes used for droplet routing while considering the resource constraint [16].

Layout Design: The purpose of layout design is to map the functional units such as mixers, storage units, and routing paths to the underlying hardware. Griffith and Akella presented a semi-automated method to generate the array layout in terms of components [14]. Su and Chakrabarty developed an online reconfigurable technique to bypass the fault unit cells in the microfluidic biochips [17].

III. Binary tree model of biochemical analysis

The (biochemical) "analysis graph" provides a representation of the operations of DMFS. It is a directed graph, with an input node for each droplet type entering the system, an output node for each droplet type leaving the system, and a mix node for each mixing operation performed in the system. The nodes are connected based on the droplet types they require and produce, and the edges represent transport operations. For example, consider the PCR analysis graph shown in Fig. 2. The mixing operations during the PCR analysis are described as follows.

$A + B \rightarrow C$; $C + D \rightarrow E$; $E + F \rightarrow G$;

$H + I \rightarrow J$; $J + K \rightarrow L$; $L + M \rightarrow N$; $N + G \rightarrow O$.

Here $A + B \rightarrow C$ means reagents A and B are mixed to produce C. To model the dependencies among different mixing operations, we introduce a full binary tree structure. A full binary tree is a binary tree in which every node is either a leaf node or it has two child nodes. Given an analysis R, we construct a full binary tree T as follows. Every reagent in R is represented by a node in T: source reagents, which can be directly fetched from the reservoirs, are represented by leaf nodes in T, and intermediate reagents, which are produced by mixing, are represented by parent nodes of those nodes from which they can be produced. So the analysis R is represented by T, where the final product of R is represented by the root node of T. The representation of the PCR analysis of Fig. 2 by a full binary tree is shown in Fig. 3.

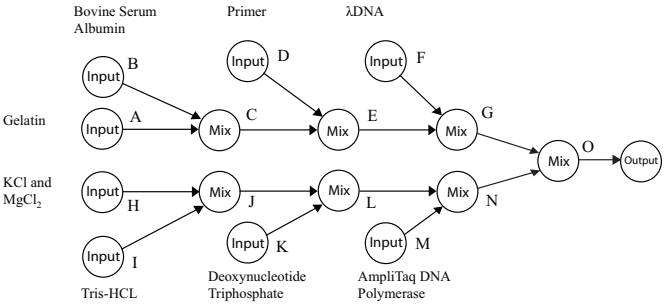

Fig. 2. PCR analysis graph. Input nodes are labeled with the reagents they introduce and alphabet labels for convenience. Mix nodes represent the mixing operations. The output node represents the final product.

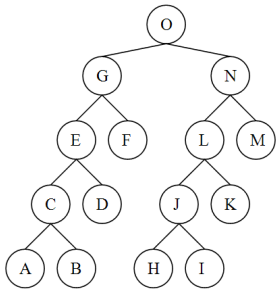

Fig. 3. Representation of PCR analysis by a full binary tree with depth 4.

IV. Scheduling Algorithm Design

For a general biochemical analysis, there are five kinds of basic operations: *input*, *transport*, *mix* (including split), *store*, and *output*. The execution order of these operations should satisfy several rules:

1. *Operation precedence rule*: Consider a mixing for C, $A+B \rightarrow C$. The precedence order of related operations is: *Input* A and B > *Transport* A and B to a mixer > *Mix* A and B for C > *Output*/*Store*/*Transport* C.
2. *Mixing precedence rule*: In the binary tree model, the mixing operation for a parent node should occur later than the mixing operation for its child nodes.
3. *Resource constraint rule*: At any time the utilized resource should not exceed the resources of the biochip.

Our scheduling design is composed of two parts: pipelining and mixing scheduling. Pipelining overlaps mixing operations with other operations and meanwhile satisfies the operation precedence rule, while scheduling of mixing operations reduces the whole mixing time and meanwhile satisfy the mixing precedence rule. Both pipelining and mixing scheduling should satisfy the resource constraint rule.

A. Pipelining

Pipelining is a scheduling method that divides a task into several subtasks and performs different subtasks of multiple-tasks in different function units simultaneously to reduce the overall completion time of all tasks [18]. Using pipelining, the average completion time of a single task will depend on the subtask that takes the longest time.

For biochemical analyses on a DMFS, the procedure of obtaining a droplet can be divided into several subtasks: input source droplets, transport droplets, mix (and split) droplets, and output waste droplets. Among these subtasks, droplet mixing takes the longest time. The mixing duration depends on the mixer size, droplet motion pattern and the chemicals to be mixed [5]. To simplify our analysis, we assume all mixing operations have the same duration. Fig. 4 shows an example of applying pipelining to a part of PCR analysis.

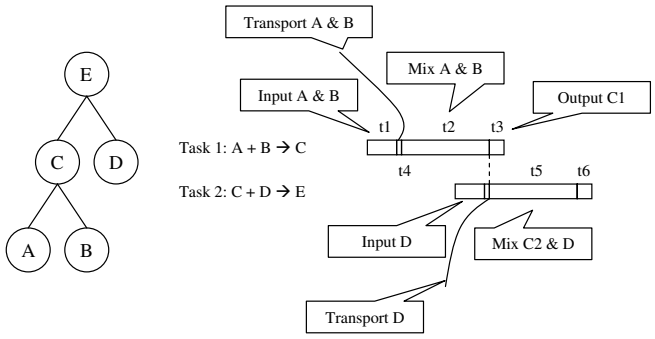

Fig. 4. Application of pipelining to an analysis on DMFS. After mixing operations (including splitting), there will be two destination droplets produced, labeled with 1 and 2 respectively. The output operations are used to dispense one waste droplet (labeled with 1) outside the system. Two mixing tasks are performed in the same mixer, so droplet C2 does not need to be transported for the second mixing task.

Suppose $t_{completion}$ is the completion time, T_{input}, T_{output} and T_{mix} are the times for one input, output, and mixing operation respectively, t_{mix}^{total} is the total time for mixing, and $t_{transport}$ is the time for transportation operations.

$$t_{completion} = T_{input} + t_{mix}^{total} + t_{transport} + T_{output} \quad (1)$$

where t_{mix}^{total} equals $\sum T_{mix}$.

Also, since the transportation time is negligible compared to the mixing time [6], [7], we ignore the transportation time in subsequent scheduling. The approximate completion time:

$$t_{completion} \approx T_{input} + t_{mix}^{total} + T_{output} \quad (2)$$

Since T_{input}, T_{output}, and T_{mix} are constants, we only consider optimization of mixing completion time t_{mix}^{total} below.

B. Scheduling Mixing Operations

The goal of mixing scheduling is to minimize t_{mix}^{total} using a given number of mixers. t_{mix}^{total} depends on three factors:

- The tree structure of the biochemical analysis,
- The number of mixers provided,
- The scheduling algorithm.

In the following subsections, we will answer the questions below:

1) What is the lower bound of the mixing completion time for a biochemical analysis?
2) What is the minimum number of mixers to achieve the lower bound on the mixing completion time?
3) What is the optimal scheduling algorithm with a limited number of mixers for an arbitrary analysis?
4) What if we have only one mixer or zero storage units?

C. Lower Bound of Mixing Completion Time

Suppose T is a full binary tree that represents the biochemical analysis R, the depth of T is K, and the number of nodes at level i is N_i, $i = 0, \ldots K$. (The root node of T is at level 0 and the deepest leaf nodes are at level K.) Here the mixing duration T_{mix} is defined as the duration from the time when the second reagent arrives at the mixer entrance to the time when the mixed product exits the mixer.

Lemma 1: The lower bound $T_{lowerbound}$ of t_{mix}^{total} is the sum of mixing durations along the deepest branch of T.

$$T_{lowerbound} = K \cdot T_{mix} \quad (3)$$

Proof. First, we show that all mixing operations can be completed in $K \cdot T_{mix}$ time with $\max_{i=1}^{K} \frac{N_i}{2}$ mixers. We perform the mixing operations from the deepest nodes to the highest nodes until the root node at level 0. Since the number of mixers $\max_{i=1}^{K} \frac{N_i}{2} \geq \frac{N_i}{2}$, we can schedule all the mixing operations on level i in a mixing duration T_{mix}. After the mixing operation on level 1, we get the root node. In this case $t_{mix}^{total} = K \cdot T_{mix}$. So $T_{lowerbound} \leq K \cdot T_{mix}$.

Second, we show that $T_{lowerbound} \geq K \cdot T_{mix}$. Suppose we can perform all mixing operations in less time, that is, $T_{lowerbound} < K \cdot T_{mix}$. Then according to the *pigeonhole principle*, there must be at least two mixing operations along the deepest branch of the tree performed in the same time slot, which contradicts the mixing precedence rule.■

D. Minimal Number of Mixers S to Achieve the Lower Bound of Mixing Completion Time

Before computing the minimal number of mixers, we introduce a lemma first. If we remove the reagent nodes of performed mixing operations after each time slot, the scheduling of mixing operations can be represented by a sequence of full binary trees. Let $Tree_i$ denote the full binary tree after the ith mixing duration, T_s denote the number of mixing durations for the final product. So $Tree_0$ is the initial tree, and $Tree_{T_s}$ is the root node.

Lemma 2: The number of mixing operations that can be performed for $Tree_i$, $i = 0, \ldots, T_s$, will change in non-increasing order.

Proof. A mixing operation in $Tree_i$ can be performed only when both reagents for that mixing are leaf nodes. Let P denote the number of such mixing operations. Each mixing operation belongs to one of three cases, based on its parent mixing operation as shown in Fig. 5.

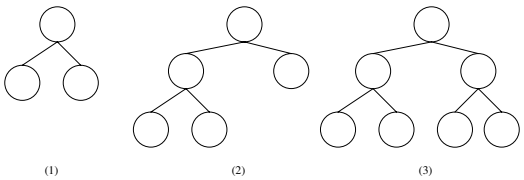

Fig. 5. Classification of mixing operations. Case 1: It is not followed by subsequent mixing operations. Case 2: Its parent's other reagent is a leaf node. Case 3: Its parent's other reagent results from a mixing operation.

The second case will not influence the size of P, while the first and third case will decrease the size of P by 1.■

Next we calculate the minimal number of mixers required to achieve the lower bound of mixing completion time. The sufficient and necessary condition is that the mixing operations along the deepest branch can be performed one by one in each mixing duration T_{mix}.

Let S denote the minimal number of mixers to achieve the lower bound of mixing completion time, (i, j) denote the jth node in the ith level, $M_{(i,j)}$ denote cumulative number of mixing operations from leaf nodes to (i, j), and N_i denote the number of nodes on the ith level.

Theorem 1: To achieve the lower bound of mixing completion time, the sufficient and necessary condition is

$$\sum_{j=1}^{N_i} M_{(i,j)} \leq S \cdot (K - i) \qquad \text{for all } i \in \{0, 1, \cdots, K-1\} \tag{4}$$

where

$$M_{(i,j)} = \begin{cases} 0 & \text{if } (i,j) \text{ is a leaf node;} \\ M_{(i+1,j_1)} + M_{(i+1,j_2)} + 1 & \text{if } (i+1,j_1) \text{ and } (i+1,j_2) \text{ are left and right child nodes of } (i,j) \end{cases}$$

That is,

$$S = \max_{i=0}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil) \tag{5}$$

Proof. When the deepest branch is processed up to the ith level, the required number of mixing operations is $\sum_{j=1}^{N_i} M_{(i,j)}$ and the number of mixing operations that can be performed by S mixers during the elapsed $K-i$ time slots is $S \cdot (K - i)$. So the necessary condition is that Formula 4 holds true.

Next, we show that Equation 4 is also the sufficient condition for $t_{mix}^{total} = T_{lowerbound}$ by proving that the number of mixers $S_z = \max_{i=z}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$ can guarantee that nodes in the level i can be produced before the end of time slot $K - i$ for all $i = K - 1, \ldots, z$.

Base Step: $z = K - 1$. $S_{K-1} = \sum_{j=1}^{N_{K-1}} M_{(K-1,j)}$ is the number of mixing operations to produce all nodes at level $K - 1$. The conclusion holds true.

Induction Step: For some level v, suppose that all mixing operations for nodes in levels $l > v$ can be performed using $S_{v+1} = \max_{i=v+1}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$ mixers. $S_v = \max_{i=v}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$. $S_v \geq S_{v+1}$. So all mixing operations for nodes in levels $l > v$ can be performed using S_v mixers. If in some time slot $K - l$, there are less number of mixing operations than S_v that can be performed, according to Lemma 2, the situation will continue to time slot $K - v$. So all mixing operations for nodes in all levels will be performed. Otherwise, if in all time slots from 1 to $K - v$, there are enough mixing operations to be performed, all mixing operations can also be performed in such case since $S_v = \max_{i=v}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil) \geq (\lceil \frac{\sum_{j=1}^{N_v} M_{(v,j)}}{K-v} \rceil)$.

So the number of mixers $S_z = \max_{i=z}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$ can guarantee that the mixing operations for nodes in level i can be performed before the end of time slot $K - i$ for all $i = K - 1, \ldots, z$. So the root node, which is in level 0, can be produced at the end of time slot K. So the lower bound of mixing completion time is achieved using S mixers. ■

E. Optimal Mixing Scheduling with Given Number of Mixers

The mixing scheduling in Algorithm 1 are basically bottom to top (this scheduling is in fact the CP rule for parallel scheduling [19]).

- First perform the mixing operations at the deepest level.
- Perform as many mixing operations as possible using given number of mixers.

Algorithm 1 Optimal-Scheduling-with-N-Mixers

Input: T, N *// The binary tree and the number of mixers*
Output: F *// Schedule*
$i = 1$ // time slot index
$F = \emptyset$
while $T \neq \emptyset$ **do**
  $C = \emptyset$ *// mixing that can be performed currently*
  Find pairs of leaf nodes with the same parent nodes in T *// identify mixing operations ready to be performed*
  Store them in C, ordered from deepest level upwards
  if $|C| < N$ **then**
    $F = F \bigcup \{$Perform all mixing operations in C in the time slot $i\}$
  else
    $F = F \bigcup \{$Perform the first N mixing operations of C in the time slot $i\}$
  end if
  Update T by removing leaf nodes involving the performed mixing operations
  $i = i + 1$
end while
return F

Theorem 2: Algorithm 1 is the optimal scheduling algorithm with N mixers to minimize mixing completion time.

Proof. Suppose algorithm O is optimal with N mixers minimizing mixing completion time and T_O, T_Q are the completion time of O and our algorithm, respectively.

First consider our algorithm described above. Let i be the time slot when our algorithm finally encounters $|C| > N$ and all operations in C mix nodes belonging to the same level. Let d denote the depth of T at the beginning of time slot i. According to Lemma 2 and the bottom-top execution order of Algorithm 1, $|C| > N$ was always satisfied before the i^{th} time slot and no node at level higher than d was mixed. Also for every subsequent time slot, the algorithm will finish mixing operations of all nodes at subsequent level. That is, $T_Q = i + d$.

Suppose T_1 is the number of time slots spent mixing all nodes at level d or deeper by O, U is the number of mixing operations which need to be performed for nodes at level d or deeper and $T_2 = T_O - T_1$.

At the end of time slot i, our algorithm performed fewer

mixing operations than U using N mixers, so $U > N \cdot i$, $T_1 \geq U/N \geq i+1$. According to Lemma 1, $T_2 \geq d-1$. Thus, $T_O = T_1 + T_2 \geq i + d$.
So $T_O \geq T_Q$. That is, our algorithm is optimal with N mixers to minimize the mixing completion time. ■

F. Resource Constraint Analysis for Two Extreme Cases

The resource constraints come from two aspects: the number of mixers and storage units. The former limits how many mixing operations we can perform in a time slot. The latter gives a constraint for the number of droplets that have been produced but cannot immediately be mixed.

Case One: Scheduling with Only One Mixer

At least one mixer is required to be able to perform any mixing operations. Let M denote the number of mixing operations to perform for the biochemical analysis.

Lemma 3: With one mixer, all scheduling will have the same t_{mix}^{total} as long as we keep the mixer busy:

$$t_{mix}^{total} = M \cdot T_{mix} \tag{6}$$

Proof. No matter which scheduling method we use, only one mixing operation can be performed in each time slot due to the mixer resource constraint. As long as the mixer keeps busy, the mixing completion time is the sum of durations of all mixing operations. $t_{mix}^{total} = M \cdot T_{mix}$. ■

Given a reaction, we design Algorithm 2 to compute the minimum number of storage units, and Algorithm 3 to generate the corresponding schedule. Suppose $T.root$ is T's root node, $T.left$ and $T.right$ are the left and right subtrees of T. Since the tree is a full binary tree, either $T.root$ is a leaf node, or both $T.left$ and $T.right$ are non-null.

Algorithm 2 Min-Storage-Units

Input: T // the binary tree
Output: N // the number of required storage units
if $T.root$ is a leaf node **then**
  $T.value = 0$
else
  $Left$=Min-Storage-Units($T.left$)
  $Right$=Min-Storage-Units($T.right$)
  if $Left = Right$ **then**
    $T.value = Left + 1$
  else
    $T.value = \max(Left, Right)$
  end if
end if
if T is the initial input tree & $T.value \neq 0$ **then**
  $N = T.value - 1$
else
  $N = T.value$
end if
return N

Theorem 3: Algorithm 2 returns the minimum number of storage units for a reaction using one mixer; Algorithm 3 outputs the corresponding schedule in the order of execution.

Algorithm 3 One-Mixer-Scheduling

Input: T, i // the binary tree
// and the time slot index with initial value 1
Output: F // Schedule
$F = \emptyset$
if $T.value = 0$ **then**
  return
else
  if $T.left.value > T.right.value$ **then**
    One-Mixer-Scheduling($T.left$, i)
    One-Mixer-Scheduling($T.right$, i)
  else
    One-Mixer-Scheduling($T.right$, i)
    One-Mixer-Scheduling($T.left$, i)
  end if
end if
$F = F \bigcup \{$Perform mixing to get $T.root$ in time slot $i\}$
$i = i + 1$
return

Proof. Suppose $S(T)$ is the minimum number of storage units for the reaction represented by tree T using one mixer. First we prove the first half of the theorem. If $T.root$ is a leaf node or $T.value = 1$, $S(T) = 0$. (Base step)

If $T.root$ is not a leaf node, $S(T) \geq \max(S(T.left), S(T.right))$.

If $S(T.left) \neq S(T.right)$, without loss of generality assume $S(T.left) > S(T.right)$. Schedule as follows: first, perform all mixing operations in the left subtree, when we need $S(T.left)$ storage units; then perform those in the right subtree, when we need $S(T.right)$ storage units for the nodes in the right subtree and one for the $T.left$ node. So $S(T) = \max(S(T.left), S(T.right))$.

If $S(T.left) = S(T.right)$, we show that $S(T) = S(T.left) + 1$. Perform all mixing operations in the left subtree, and then the right subtree. Here we need $S(T.left) + 1$ storage units. So $S(T) \leq S(T.left) + 1$. If $S(T) < S(T.left) + 1$, then two conditions should be satisfied: first, in the time slot when $S(T.left)$ storage units is used for the left subtree, no mixing operations in the right subtree has been performed; second, in the time slot when $S(T.right)$ storage units is used for the right subtree, no mixing operations in the left subtree has been performed. Obviously, they cannot be satisfied at the same time. By contradiction, we conclude $S(T) = S(T.left) + 1$. The induction step is also correct. So N returned by the algorithm equals $S(T)$, the optimal value.

We now show that the second half of the theorem is correct. Since Algorithm 3 outputs the mixing operations in child-node-first order, the mixing precedence rule is satisfied. Also, the algorithm traces back through $T.value$ for all subtrees using the recurrence in Algorithm 2, so it outputs the corresponding scheduling results. ■

Case two: Scheduling with Zero Storage Units

Theorem 4: The sufficient and necessary condition of performing all the mixing operations without storage units

TABLE I

RESULTS OF PCR ANALYSIS USING ONE AND TWO MIXERS. ALL TIMES ARE IN SECONDS. APPROXIMATE COMPLETION TIME DOES NOT CONSIDER TRANSPORTATION TIME.

Num of Mixers	Num of Storage Units	Total Mixing Time	Completion Time	
			Approx.	Accurate
1	1	35	37	37.8
2	0	20	22	22.54

is to use $\max_{i=0}^{K} \frac{N_i}{2}$ mixers, and the completion time in this case is the lower bound $K \cdot T_{mix}$.

Proof. Actually, we have considered the sufficient condition when discussing the lower bound of mixing completion time in the proof of Lemma 1. There, using $\max_{i=0}^{K} \frac{N_i}{2}$ mixers, we provide a scheduling algorithm, which achieves the lower bound of the completion time and needs zero storage units. So we only need to prove that it is also the necessary condition for performing all mixing operations with zero storage units.

First we show that all mixing operations for non-leaf nodes at the same level must be performed in the same time slot if there are zero storage units.

Base Step: There is only one mixing operation for the root node at level 0, so the conclusion is correct obviously.

Induction Step: For some level i, suppose the conclusion holds true for all levels $j < i$. If the conclusion does not hold true for level i, there must be at least two non-leaf nodes A and B in level i, and say A is produced before B. According to the induction hypothesis, A and B are consumed for other non-leaf nodes at level $i-1$ in the same time slot. Obviously, one storage unit is required for A. By contradiction, we get the conclusion.

Since all mixing operations for non-leaf nodes at the same level must be performed in the same time slot, the number of mixers is $\max_{i=0}^{K} \frac{N_i}{2}$ as desired. ■

V. EXAMPLE

In this section, we apply our scheduling analysis and algorithms to the PCR analysis. Let $t_{transport}$ be the switching time of transportation. The parameter values we use are: $T_{mix} = 5\ sec$, $T_{input} = T_{output} = 1\ sec$, $t_{transport} = 0.01\ sec/electrode$. (In the experiments of [6] and [7], $t'_{transport} = 0.006 - 0.013\ sec/electrode$.) The depth of binary tree for PCR analysis is $K = 4$.

The results are shown in Table I. Mixing time takes up most of the completion time since pipelining is used to overlap other operations with mixing. Accurate completion time (calculated by Equation 1) is almost the same as approximate completion time (calculated by Equation 2) since the transportation speed is very fast. (During one mixing duration, droplets can be transported over $\frac{5}{0.01} = 500$ electrodes.) The mixing time using two mixers reduces to 57% of that using one mixer due to parallel mixing operations. (The percentage is larger than 50% since one mixer is idle when the last mixing is performed.) By Theorem 1, $T_{lowerbound} = 20\ sec$, which is the mixing time shown in Table I using two mixers.

We illustrate the scheduling results when one and two mixers are used in Fig. 6. We can see that the pipelining technique is used to overlap different kinds of operations such as *input*, *transport*, *mix*, *store* and *output*. Comparing the results with one and two mixers, we see that more mixers can exploit the parallelism inside the mixing operations of PCR analysis. By Theorem 1, the lower bound of completion time has been achieved, so more mixers than two cannot decrease the completion time any more.

Other cases of much more complicated biochemical analysis can also be handled in a similar fashion. Also the analysis and algorithms can easily be extended to perform several independent biochemical analyses in parallel.

VI. CONCLUSION

In this paper, we focus on scheduling algorithms to optimize the completion time of biochemical analyses on a DMFS by exploiting the tree structure of these reactions. Using pipelining techniques, most input and output operations can be overlapped with mixing operations. So our goal is to complete the mixing operations as soon as possible with given number of mixers while overlapping other operations with mixing. We introduce a full binary tree structure to model the biochemical analysis. We calculate the lower bound on the mixing completion time of a biochemical analysis and compute the minimum number of mixers to achieve such lower bound. Also we design a scheduling algorithm of mixing operations for a given number of mixers, which is proved to be optimal in the measure of completion time. Finally, resource constraint issue is also considered for two extreme cases of having just one mixer and zero storage units. Our results can be applied to various analyses on general-purpose electrowetting-based DMFS platforms. In our future work, we will extend these scheduling algorithms to mixing operations with different durations and implement our algorithms in software to control DMFS biochips.

ACKNOWLEDGMENTS

This work was supported in part by NSF under Award No. IIS-0093233 and Award No. IIS-0541224. We would like to thank the reviewers for their valuable comments.

REFERENCES

[1] P. Dittrich and A. Manz, "Lab-on-a-chip: microfluidics in drug discovery," *Nature Reviews Drug Discovery*, vol. 5, no. 3, pp. 210–218, Mar. 2006.

[2] B. Zheng, C. Gerdts, and R. F. Ismagilov, "Using nanoliter plugs in microfluidics to facilitate and understand protein crystallization," *Current Opinion in Structural Biology*, vol. 15, pp. 548–555, 2005.

[3] X. X. Chen, H. K. Wu, C. D. Mao, and G. M. Whitesides, "A prototype two-dimensional capillary electrophoresis system fabricated in poly(dimenthylsiloxane)," *Anal. Chem.*, vol. 74, pp. 1772–1778, 2002.

[4] V. Srinivasan, V. K. Pamula, and R. B. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab on a Chip*, vol. 5, no. 3, pp. 310–315, Mar. 2004.

[5] P. Paik, V. K. Pamula, and R. B. Fair, "Rapid droplet mixers for digital microfluidic systems," *Lab on a chip*, vol. 3, pp. 253–259, Sept. 2003.

[6] S. K. Cho, H. Moon, and C. Kim, "Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits," *J. Microelectromech. Syst.*, vol. 12, no. 1, pp. 70–80, Feb. 2003.

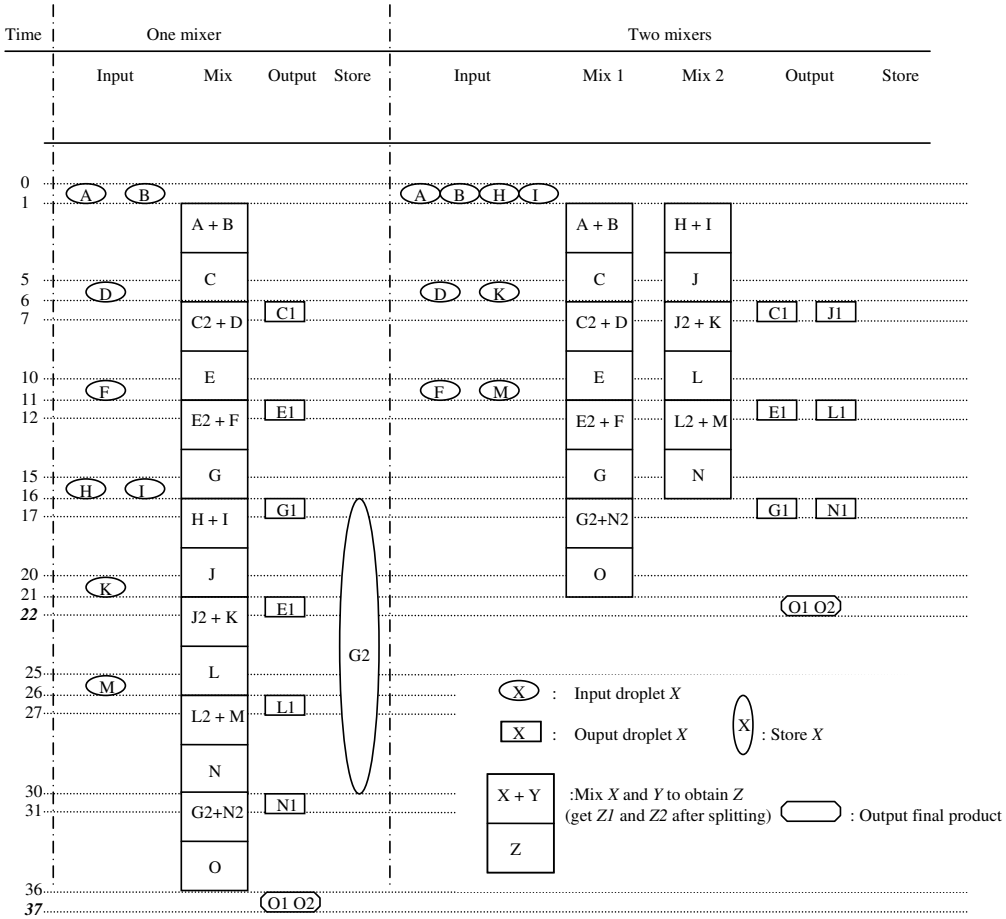

Fig. 6. Scheduling results of performing PCR analysis on DMFS. The left side shows results with one mixer, while the right side shows result with two mixers. Transportation time is ignored here. Accurate completion time considering transportation time is shown in Table I. All times are in seconds.

[7] R. B. Fair, V. Srinivasan, H. Ren, P. Paik, V. Pamula, and M. G. Pollack, "Electrowetting-based on-chip sample processing for integrated microfluidics," in *IEEE International Electron Devices Meeting (IEDM)*, 2003, pp. 779–782.

[8] M. G. Pollack, R. B. Fair, and A. D. Shenderov, "Electrowetting-based actuation of liquid droplets for microfluidic applications," *Appl. Phys. Lett.*, vol. 77, no. 11, pp. 1725–1726, Sept. 2000.

[9] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *Proc. IEEE International Conference on CAD*, 2004, pp. 223–228.

[10] ——, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. IEEE/ACM Design Automation Conference*, 2005, pp. 825–830.

[11] Y. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, pp. 406–471, 1999.

[12] J. Ding, K. Chakrabarty, and R. B. Fair, "Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 329 – 339, Feb. 2006.

[13] K.-F. Böhringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 12, pp. 1463–1468, Dec. 2001.

[14] E. J. Griffith and S. Akella, "Coordinating multiple droplets in planar array digital microfluidic systems," *International Journal of Robotics Research*, vol. 24, no. 11, pp. 933–949, Nov. 2005.

[15] E. J. Griffith, S. Akella, and M. K. Goldberg, "Performance characterization of a reconfigurable planar array digital microfluidic system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 340–352, Feb. 2006.

[16] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Design, Automation and Test in Europe (DATE) Conference*, Munich, Germany, Mar. 2006, pp. 323–328.

[17] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Transactions on Design Automation of Electronic Systems*, pp. 682–710, 2006.

[18] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. Morgan Kaufmann, 2002.

[19] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1995.