

---

# Minimum Resource Characterization of Biochemical Analyses for Digital Microfluidic Biochip Design

Lingzhi Luo<sup>1</sup> and Srinivas Akella<sup>1</sup>

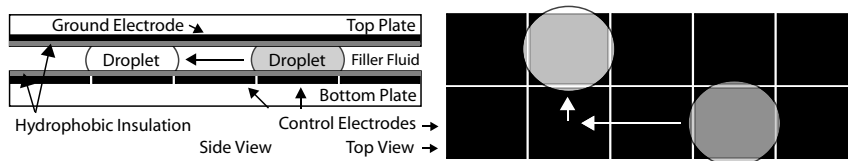
Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA. [luo12@cs.rpi.edu](mailto:luo12@cs.rpi.edu), [s.akella@ieee.org](mailto:s.akella@ieee.org)

**Abstract:** Digital microfluidic systems (DMFS) are a class of lab-on-a-chip systems that manipulate individual droplets of chemicals on an array of electrodes. The biochemical analyses are performed by repeatedly moving, mixing, and splitting droplets on the electrodes. In this paper, we characterize the tree structure of biochemical analyses and identify their minimum resource requirements, towards the design of cost and space-efficient biochips. Mixers and storage units are two primary functional resources on a DMFS biochip; mixers mix and split droplets while storage units store droplets on the chip for subsequent processing. Additional DMFS resources include input and output units and transportation paths. We present an algorithm to compute, for a given number of mixers  $M$ , the minimum number of storage units  $f(M)$  for an input analysis using its tree structure, and design a corresponding scheduling algorithm to perform the analysis. We characterize the variation of the  $M$ -depth of a tree (i.e., its minimum number of storage units  $f(M)$ ) with  $M$ , and use it to calculate the minimum total *size* (the number of electrodes) of mixers and storage units. We prove that the smallest chip for an arbitrary analysis uses one mixer and  $f(1)$  storage units. Finally, we demonstrate our results on two example biochemical analyses and design the smallest chip for a biochemical analysis with a complete tree structure of depth 4. These are the first results on the least resource requirements of DMFS for biochemical analyses, and can be used for the design and selection of chips for arbitrary biochemical analyses.

## 1 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, point-of-care diagnostics, and biochemical sensing [2,5,13,18]. *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 an array of electrodes by using electrowetting (or dielectrophoresis). We focus on microfluidic systems that manipulate

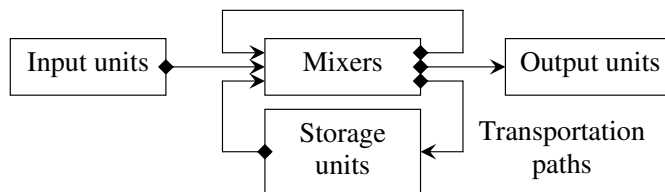
droplets by electrowetting [11]. The chemical analysis is performed by repeatedly moving, mixing, and splitting droplets on the electrodes (Figure 1).



**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 [12].

The ability of DMFS devices to control discrete droplets enables complex analysis operations to be performed in a flexible manner. There are several classes of resources (i.e., functional components consisting of electrodes) in a DMFS: mixers, storage units, input/output units, and transportation paths. Figure 2 illustrates their functions. We focus on performing biochemical analyses in batch mode, where the goal is to produce one droplet of the final product. We previously [10] developed algorithms to compute minimum time schedules based on the tree structure of chemical analyses. Here we focus on determining minimum resources based on the tree structure of the chemical analysis and use it to design (or select) the smallest chip for a given analysis.

Biochip users want versatile, yet low cost system. Hence it is important to identify the class of biochemical analyses a given chip can perform, and further, the smallest chip that can perform a given set of analyses. In this paper, we characterize the structure of different biochemical analyses and classify them according to their least resource requirements. Since we can reduce the fabrication cost of a DMFS chip by reducing the number of electrodes, this work is motivated by two practical questions: For any biochemical analysis, how do we identify the (smallest) biochip with sufficient number of resources to perform it? For a given biochip, what kind of biochemical analyses can be performed on it? We assume that the resources (e.g., mixers and storage units) are fixed at the start of the analysis and do not change over the course of the analysis. First, we compute the least resource requirements based on the tree structure of biochemical analyses and design corresponding algorithms to perform them. Then we define the  $M$ -depth of a tree to describe such resource requirements and characterize the variation of the  $M$ -depth with the number of mixers  $M$ . This is the first work on DMFS to compute the least resource requirements for biochemical analyses and it can be used to guide the selection and design of chips for arbitrary biochemical analyses.



**Fig. 2.** Five classes of functional resources for DMFS biochips. Transportation paths (shown as arrows starting from diamonds) are used to move droplets from one resource to another. Input units are used to input droplets. Mixers are used to mix two different droplets and split their mixture to produce two new droplets. In batch mode, one of the new droplets will be used and the other will be discarded as a waste droplet at an output unit. The new droplet may be kept idle in a storage unit for some time, stay at the mixer, or be transported to a new mixer for a subsequent mixing. The droplet of final product will be transported to an output unit.

## 2 Related Work

*Resource Requirements:* Su and Chakrabarty presented architecture-level synthesis and geometry synthesis of biochips [14, 15]. They used acyclic sequence graphs to represent the reactions and developed techniques in operation scheduling, resource binding, and module placement. In their papers resource constraints are given in advance by the size of chips or the number of mixers and storage units. However there is no general guideline for selecting chips with proper number of mixers and storage units for biochemical analyses. Griffith and Akella [7] explored the relationship between the number of mixing units and the highest stable input rates in the system. By simulations, it was found that increasing the number of mixing units permits the system to be stable at a higher input rate and the effectiveness of maintaining system stability by increasing the number of mixers decreases. They experimented with a variety of modifications to the resources to gauge the effects on the stability of the system [8]. The work in [7, 8] is based on simulations while in this paper we are formally analyzing the resource requirements for analyses based on the underlying tree structure.

*Layout Design:* Layout design maps 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 [7]. Su and Chakrabarty developed an online reconfigurable technique to bypass the fault unit cells in the microfluidic biochips [16].

*Scheduling Algorithms:* Scheduling algorithms optimize the system performance by properly allocating tasks to resource. Kwok and Ahmad studied the static scheduling of a program on a multiprocessor system to minimize the program completion time in parallel processing [9]. 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

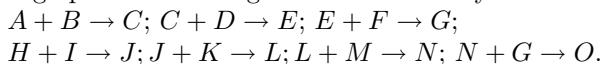
binary tree. Ding et al. [4] and Su and Chakrabarty [14] represent the DMFS reactions using data-flow directed graph and consider the scheduling using Integer Linear Programming (ILP). They solve the problem by general ILP solvers and heuristic algorithms without exploiting the structure of DMFS reactions.

*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 [1]. Griffith and Akella presented a general-purpose DMFS and designed routing algorithm based on Dijkstra’s algorithm [7, 8]. 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 [17].

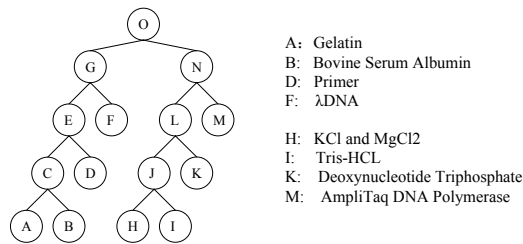
### 3 Model and Problem Formulation

#### 3.1 Full Binary Tree Model

The (biochemical) “analysis graph” provides a representation of the operations of a 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. A mixing operation  $A + B \rightarrow C$  means reagents  $A$  and  $B$  are mixed to produce  $C$ . The mixing operations during a DNA PCR analysis are described as follows.



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 has either zero or two child nodes. A *complete binary tree* is a full binary tree in which all leaf nodes are at the same level. (Levels of nodes are defined in a recursive way as follows: the level of the root node is zero; the level of any other node equals one plus that of its parent node.) 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 by a full binary tree is shown in Figure 3. In the following sections, we will analyze biochemical reactions and design algorithms based on the full binary tree structure.



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

### 3.2 Problem Formulation

The problems we want to solve in this paper are: *Given a biochemical analysis, what are the minimum requirements of resources (e.g., the number and size of mixers, storage units, input/output units and transportation paths) to perform it? How can we design the scheduling algorithm so that the analysis can be performed with the minimum resources?* The number of input and output units is determined by the given biochemical analysis. Transportation paths are constructed so that other function units (the input units, output units, mixers and storage units) are connected. We will first focus on the resource requirements of mixers and storage units, and then consider the requirements of other resources as well in Section 5.

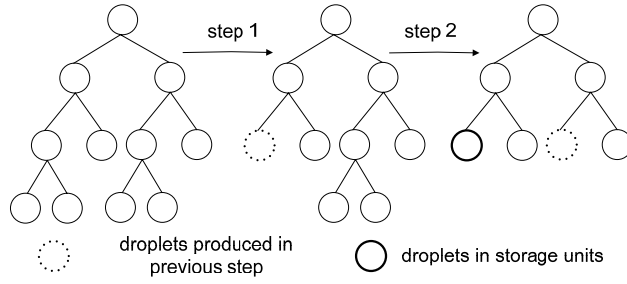
#### Pipelining

For biochemical analyses on a DMFS, the procedure of obtaining a droplet can be divided into several operations: input source droplets, transport droplets, mix (and split) droplets, and output waste droplets. To simplify our analysis, we assume all mixing operations have the same duration. After applying pipelining techniques [10], different operations can be overlapped.

Also, since the transportation time is negligible compared to the mixing time [3, 6], we ignore the transportation time in subsequent scheduling. So when discussing the minimum resource requirements, we can just focus on the mixing scheduling.

#### Mixers and Storage Units

The number of mixers limits how many mixing operations we can perform in a time slot. The number of storage units gives a constraint for droplets that have been produced but cannot immediately be mixed. Figure 4 shows how mixers and storage units are used during the progress of a biochemical analysis.



**Fig. 4.** Progress of a biochemical analysis on a chip with a single mixer. The intermediate reagent produced in the first step is stored in a storage unit since it is not involved in the mixing in the second step.

### Scheduling Representation

Each scheduling step can be regarded as the transition from one full binary tree to another where some sibling nodes are removed (see Figure 4). So the whole schedule  $Sch$  of a biochemical analysis can be represented by a series of full binary trees, starting from the initial full binary tree and ending at the root node, and the transitions between consecutive trees correspond to removing selected sibling nodes.

For a schedule  $Sch$ , let the number of mixers be  $M$  and the number of storage units be  $S$ . We can formulate the scheduling problem as the following two questions:

*Given an initial full binary tree  $T$ , (1) What is the schedule  $Sch$  in which  $S$  is minimized given  $M$ ? (2) What is the schedule  $Sch$  in which the resource requirements (i.e., the number of electrodes for  $M$  mixers,  $S$  storage units, input/output units and transportation paths) are minimized?*

## 4 Minimizing the Number of Storage Units with a Given Number of Mixers

In our previous work [10], we analyzed two extreme cases of resource requirements: with just one mixer and with zero storage units. In the first case, we proved all active schedules (i.e., that keep the mixer busy at all times) result in the same completion time, calculated the minimum number of storage units for a given analysis, and designed a corresponding scheduling algorithm. In the second case, we computed the minimum number of mixers required, and proved the conclusion below.

**Lemma 1.** [10] *The sufficient and necessary condition for performing all the mixing operations without storage units is to use  $\max_{i=0}^K \frac{N_i}{2}$  mixers, where  $K$  is the depth of the tree,  $N_i$  is the number of nodes at the  $i$ th level.*

In this paper, we extend the scheduling algorithm to handle the general case with  $M$  mixers and  $S$  storage units, and characterize the relationship between the number of mixers and the number of storage units required for a biochemical analysis based on its tree structure.

Given an initial full binary tree of a biochemical analysis,  $M$  mixers, and  $S$  storage units, we need to check whether or not the given biochemical analysis can be performed under such constraints. Since two constraints exist here, there are two methods to solve this problem. First, with at most  $M$  mixers, compute the minimum number of storage units  $f(M)$ . If  $f(M) \leq S$ , it is feasible to perform the biochemical analysis under such constraints, and infeasible otherwise. Second, with at most  $S$  storage units, compute the minimum number of mixers  $g(S)$ . Similarly, compare  $g(S)$  and  $M$  to check the feasibility. Since we may reduce the required number of one resource by increasing the number of the other resource, it is obvious that  $f(M)$  and  $g(S)$  are both non-increasing functions. So if we have a solution for the first method, the second one can be solved using binary search in the result of the first method, and vice versa.

In this section, first, we will compute the minimum number of storage units  $f(M)$  and design the scheduling algorithm to perform the biochemical analysis using  $M$  mixers and  $f(M)$  storage units; second, we will define  $f(M)$  as the  $M$ -depth of the tree structure and give an equation to compute the  $M$ -depth for complete trees; third, we characterize the variation with  $M$  of the  $M$ -depth of the tree. So for any biochemical analysis, we can find out whether it can be performed using  $M$  mixers and  $S$  storage units by computing the  $M$ -depth of its tree representation. If two biochemical analyses have tree structures with the same  $M$ -depth, we say they have the same resource requirements.

#### 4.1 Algorithm Design

**Theorem 1** *Algorithm 1 returns the minimum number of storage units for a reaction using at most  $M$  mixers; Algorithm 2 outputs the corresponding schedule in the order of execution.*

*Proof.* Suppose  $f_M(T)$  is the minimum number of storage units for the reaction represented by tree  $T$  using at most  $M$  mixers. First we prove the first part of the theorem. According to Lemma 1, if the maximum number of nodes at the same level is greater than  $2M$ , at least one storage unit is required. Since the algorithm scans nodes from bottom up, the nodes first satisfying this condition need one storage unit. (Base step.)

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

If  $f_M(T.left) \neq f_M(T.right)$ , without loss of generality assume  $f_M(T.left) > f_M(T.right)$ . Schedule as follows: First, perform all mixing operations in the left subtree, when we need  $f_M(T.left)$  storage units. Then perform those in the right subtree, when we need  $f_M(T.right)$  storage units for

**Algorithm 1** Min-Storage-Units-M-Mixers

---

*Input:*  $T, M$  // the binary tree and the number of mixers  
*Output:*  $f_M$  // the number of required storage units

**while** Scan nodes level by level from bottom up: **do**  
  **for** any scanned node  $i$  **do**  
     $i.f_M = 0$  // storage units to produce  $i$   
    **if**  $i.left.f_M == 0$  and  $i.right.f_M == 0$  **then**  
      // zero storage units to produce both child nodes  
      Scan the subtree rooted at  $i$ , set  $i.maxnode$  as the maximum number of nodes at any level in the subtree  
      **if**  $i.maxnode > 2M$  **then**  
         $i.f_M = 1$   
      **end if**  
    **else**  
      // at least one child node needs storage units  
      **if**  $i.left.f_M == i.right.f_M$  **then**  
         $i.f_M = i.left.f_M + 1$   
      **else**  
         $i.f_M = \max\{i.left.f_M, i.right.f_M\}$   
      **end if**  
    **end if**  
  **end for**  
**end while**  
**return**  $T.root.f_M$

---

the nodes in the right subtree and one for the  $T.left$  node. So  $f_M(T) = \max(f_M(T.left), f_M(T.right))$ .

If  $f_M(T.left) = f_M(T.right)$ , we show that  $f_M(T) = f_M(T.left) + 1$ . Schedule all mixing operations in the left subtree, and then the right subtree. We need  $f_M(T.right) + 1$  storage units as discussed above, which is equal to  $f_M(T.left) + 1$ . So  $f_M(T) \leq f_M(T.left) + 1$ . If  $f_M(T) < f_M(T.left) + 1$ , then two conditions should be satisfied: First, in the time slot when  $f_M(T.left)$  storage units are used for the left subtree, no mixing operations in the right subtree have been performed. Second, in the time slot when  $f_M(T.right)$  storage units are used for the right subtree, no mixing operations in the left subtree have been performed. Obviously, they cannot be satisfied at the same time. By contradiction, we conclude  $f_M(T) = f_M(T.left) + 1$ . The induction step is also correct. So  $f_M$  returned by the algorithm is  $f_M(T)$ , the optimal value.

We now show that the second part of the theorem is correct. Since Algorithm 2 outputs the mixing operations in child-node-first order, the mixing precedence rule is satisfied. Also, the algorithm traces back through  $T.root.f_M$  for all subtrees using the recurrence in Algorithm 1, so it outputs the corresponding scheduling results. ■

**Algorithm 2** M-Mixer-Scheduling

---

*Input:*  $T, i$  // the binary tree and the time slot index (1 for the initial tree)  
*Output:*  $F$  // Schedule, global variable initialized as  $\emptyset$

**if**  $T$  has only a root node **then**  
  **return**  
**else**  
  **if**  $T.root.f_M = 0$  **then**  
    // finish mixing operations at one level each time slot  
    **for** all levels of  $T$  from bottom up **do**  
       $F = F \cup$  {Schedule all mixing operations at the same level in time slot  $i$ }  
       $i = i + 1$   
    **end for**  
  **else**  
    // first produce the child node requiring more storage units  
    **if**  $T.root.left.f_M > T.root.right.f_M$  **then**  
      M-Mixer-Scheduling( $T.left, i$ )  
      M-Mixer-Scheduling( $T.right, i$ )  
    **else**  
      M-Mixer-Scheduling( $T.right, i$ )  
      M-Mixer-Scheduling( $T.left, i$ )  
    **end if**  
     $F = F \cup$  {Schedule the mixing operation to get  $T.root$  in time slot  $i$ }  
     $i = i + 1$   
  **end if**  
**end if**  
**return**  $F$

---

**4.2 M-Depth of Tree Structure**

From the algorithm in the last section, we see that with  $M$  mixers, the minimum number of storage units  $f(M)$  for a biochemical analysis only depends on its tree structure. Thus we can define  $f(M)$  as the  $M$ -depth of the tree structure corresponding to  $M$  mixers. From Lemma 1, we can easily see that:

**Lemma 2.** *Given a tree for a biochemical analysis,  $f(M) = 0$  if and only if  $M \geq \frac{T.root.maxnode}{2}$ .*

For complete trees, we derive an equation to compute  $M$ -depth as follows.

**Theorem 2** *For a complete tree with depth  $D$ , its  $M$ -depth can be computed:*

$$f(M) = D - (\lfloor \log_2 M \rfloor + 1) \quad (1)$$

*Proof.* Suppose in a tree with depth  $D$ , we define the height of a node as  $D$  minus its level. According to Lemma 2, in a complete tree, any node  $i$  with

$i.f_M = 0$  must satisfy that  $i.maxnode \leq 2M$  and thus has a height at most  $\lfloor \log_2 2M \rfloor$ . And for nodes with height more than  $\lfloor \log_2 2M \rfloor$ , increasing height by 1 will increase  $i.f_M$  by 1. So

$$\begin{aligned} f(M) &= T.root.f_M = T.root.height - \lfloor \log_2 2M \rfloor \\ &= D - \lfloor \log_2 2M \rfloor = D - (\lfloor \log_2 M \rfloor + 1). \blacksquare \end{aligned}$$

**Corollary 1** For a binary tree with depth  $D$  for a biochemical analysis, its  $M$ -depth must satisfy:

$$f(M) \leq D - (\lfloor \log_2 M \rfloor + 1) \quad (2)$$

### 4.3 Variation of $M$ -depth with $M$ for a Tree

We first examine the non-increasing property of the  $M$ -depth,  $f(M)$ . For a given biochemical analysis tree, according to Lemma 2, if  $f(M) = 0$ ,  $f(M + 1) = 0$ . There exist nodes where  $i.f_M > 0$ , but  $i.f_{M+1} = 0$ . Considering the same recurrence as in Algorithm 1, for each node  $i$ ,  $i.f_M \geq i.f_{M+1}$ . So  $f(M) = T.root.f_M \geq T.root.f_{M+1} = f(M + 1)$ . That is, with more mixers, we may reduce the minimum number of storage units required.

$$f(M + 1) \leq f(M) \quad (3)$$

Second, there will not be much difference between  $f(M)$  and  $f(M + 1)$ .

#### Theorem 3

$$f(M + 1) \geq f(M) - 1 \quad (4)$$

*Proof.* Use mathematical induction. Base step is trivial: for a leaf node,  $f(M + 1) = f(M) = 0$ .

For a node  $i$  with left and right child nodes, suppose  $f(M)$  is  $M$ -depth of the tree rooted at  $i$ ,  $f_L(M)$  and  $f_R(M)$  are  $M$ -depths of its left subtree and right subtree, respectively. According to the induction hypothesis,

$$f_L(M + 1) \geq f_L(M) - 1, f_R(M + 1) \geq f_R(M) - 1 \quad (5)$$

There are three possible cases below:

1.  $f_L(M) \neq f_R(M)$ : Without loss of generality, assume  $f_L(M) > f_R(M)$ .  $f(M) = f_L(M)$ . So  $f(M + 1) \geq f_L(M + 1) \geq f_L(M) - 1$  (Inequality 5) so  $f(M + 1) \geq f(M) - 1$ .
2.  $f_L(M) = f_R(M) = 0$ :  $f(M) \leq 1$ .  $f(M + 1) \geq 0 \geq f(M) - 1$ .
3.  $f_L(M) = f_R(M) \neq 0$ : According to Theorem 1,  $f(M) = f_L(M) + 1$ 
  - $f_L(M) = f_R(M) \geq 2$ :

$$f_L(M + 1) \geq f_L(M) - 1 \geq 1, f_R(M + 1) \geq f_R(M) - 1 \geq 1 \quad (6)$$

If  $f_L(M + 1) = f_R(M + 1)$ ,

$$\begin{aligned} f(M+1) &= f_L(M+1) + 1 \geq f_L(M) \text{ (according to (6))} \\ &\geq f(M) - 1 \text{ (} f(M) = f_L(M) + 1 \text{)} \end{aligned}$$

If  $f_L(M+1) \neq f_R(M+1)$ , assume  $f_L(M+1) > f_R(M+1)$ .

$$f_L(M+1) > f_R(M+1) \geq f_R(M) - 1 \text{ (according to (6))}$$

$f_L(M+1) > f_R(M) - 1$ . That is  $f_L(M+1) \geq f_L(M)$ .

So  $f(M+1) \geq f_L(M+1) \geq f_L(M) = f(M) - 1$ .

- $f_L(M) = f_R(M) = 1$ :  $f(M) = 2$ . From Lemma 2,  $i.left.maxnode > 2M$  and  $i.right.maxnode > 2M$ . So  $i.maxnode > 2(M+1)$  (At least two nodes of right (or left) subtree will be at the same level with more than  $2M$  nodes in left (or right) subtree). By Lemma 2 again,  $f(M+1) > 0$ . So  $f(M+1) \geq 1 = f(M) - 1$ .

So the induction step is also correct. Thus  $f(M+1) \geq f(M) - 1$ . ■

Considering Inequalities 3 and 4, we see that for a full binary tree,  $f(M) \geq f(M+1) \geq f(M) - 1$ . This means that using one additional mixer to perform a biochemical analysis either keeps the minimum number of storage units the same or reduces it by one. So we get the conclusion below.

**Corollary 2** *The total number of mixers and storage units  $F(M) = M + f(M)$  is a non-decreasing function of  $M$ .*

#### 4.4 Example

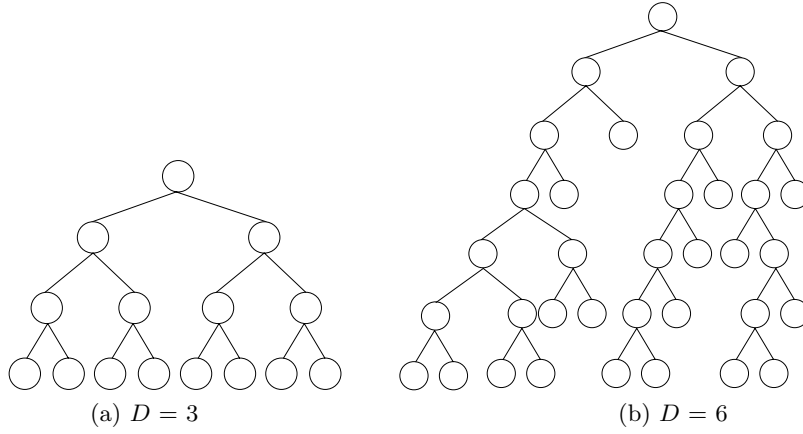
We apply the above algorithms and conclusions to characterize biochemical analyses based on resource requirements. The two trees in Figure 5 have different structures, but share the same resource requirements, as illustrated by their identical characteristic resource curve  $f(M)$  in Figure 6.

## 5 Towards Smallest Chip Design

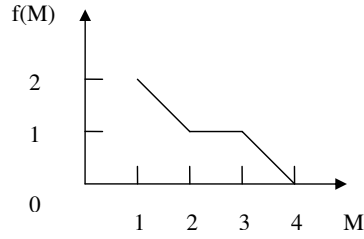
We discussed mixers and storage units in Section 4, without considering their geometry and other resources such as input/output units and transportation paths. In this section, we will consider all resources towards designing the smallest DMFS chip for biochemical analyses. The smaller the number of electrodes in a DMFS chip, the easier the fabrication and the lower the cost.

### 5.1 Mixers and Storage Units

Mixers and storage units are the two primary functional components on a DMFS biochip, where mixers are used to mix and split droplets while storage units are used to store droplets on chip for future usage. They should be large enough to prevent droplets inside them inadvertently mixing with other



**Fig. 5.** Two analysis trees that have the same least resource requirements.



**Fig. 6.** Characteristic resource curve of  $f(M)$  shared by the two trees in Figure 5.

droplets outside them. As shown in Figure 7, a storage unit needs only one electrode to hold one droplet, while a mixer needs more electrodes to move the mixed droplet inside it and thus needs more surrounding electrodes to keep the droplet inside separate from other droplets outside. It follows that the *size* of one mixer  $S_{mix}$  (the number of electrodes in one mixer) is bigger than the size of a storage unit  $S_{store}$ .



**Fig. 7.** The sizes of a mixer and a storage unit. (a) A mixer with 3 electrodes for droplets to move.  $S_{mix} = 15$ . (b) A storage unit with 1 electrode for droplets to stay.  $S_{store} = 9$ .

From the above observation and Corollary 2, we obtain the following result for a full binary tree whose  $M$ -depth is  $f(M)$ .

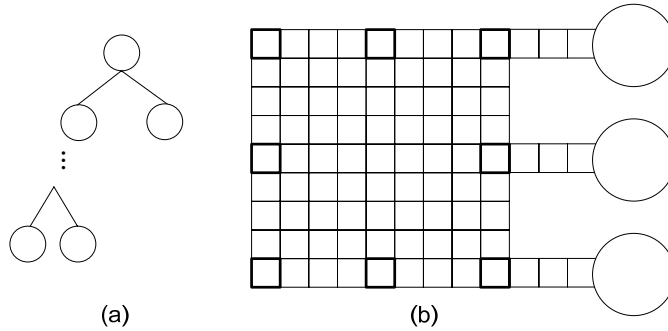
**Theorem 4** *The total size of  $M$  mixers and  $f(M)$  storage units will monotonically increase with the number of mixers.*

*Proof.* We need prove that  $M_1 \cdot S_{mix} + f(M_1) \cdot S_{store} < M_2 \cdot S_{mix} + f(M_2) \cdot S_{store}$  when  $M_1 < M_2$ .

$$\begin{aligned} M_1 \cdot S_{mix} + f(M_1) \cdot S_{store} &= (M_1 + f(M_1)) \cdot S_{store} + M_1 \cdot (S_{mix} - S_{store}) \\ &\leq (M_2 + f(M_2)) \cdot S_{store} + M_1 \cdot (S_{mix} - S_{store}) < (M_2 + f(M_2)) \cdot S_{store} + M_2 \cdot (S_{mix} - S_{store}) \\ &= M_2 \cdot S_{mix} + f(M_2) \cdot S_{store} \blacksquare \end{aligned}$$

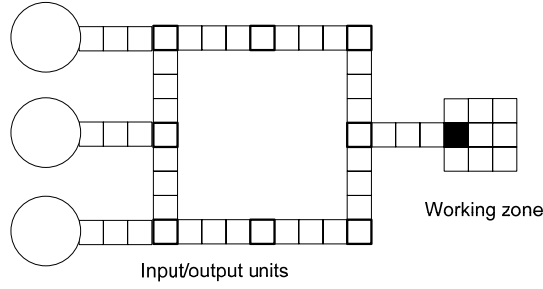
**5.2 Input and Output Units**

Input and output units should be connected to the perimeter electrodes of the chip, which we call connection electrodes, as shown in Figure 8(b). For the tree in Figure 8(a), we need just one mixer and zero storage units for the reaction. If we directly connect the input units to the chip as in Figure 8(b), the number of corresponding connection electrodes will be the number of leaf nodes of the tree in Figure 8(a). Since the connection electrodes should be on the perimeter of the chip, the total number of electrodes on the chip will be proportional to the square of the number of connection electrodes, which is much bigger than the size of one mixer. (This assumes the chip is a rectangular array of electrodes.)



**Fig. 8.** The size of chip for an analysis tree when directly connecting the input units to the chip. (a) A full binary tree of depth 6, where the right child of each node is a leaf node or a null node. (b) The chip containing sufficient connection electrodes, shown bold on the chip perimeter. (A subset of input units are shown.)

An alternative approach is to connect the input and output units to a ring of electrodes (as in [13]) and then connect the ring to a separate working zone of mixers and storage units as shown in Figure 9. The size of the working zone can be selected to have the required functionality, and the working zone can even be in the interior of the ring if it is sufficiently small.



**Fig. 9.** Connect the input and output units to the working zone of mixers and storage units through a ring so that all droplets to the working zone are input from and output to one electrode (filled in black). Here the working zone is a  $3 \times 3$  mixer.

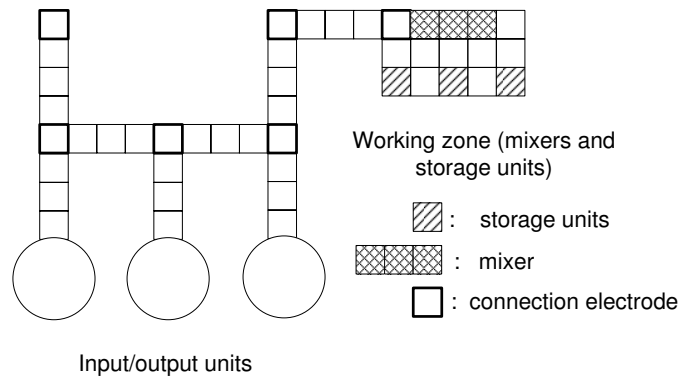
### 5.3 Transportation Paths

**Theorem 5** *Even considering the transportation paths, the smallest chip is constructed using one mixer and  $f(1)$  storage units.*

*Proof.* Proof by contradiction. First, the smallest chip can be constructed using  $M$  mixers and  $f(M)$  storage units since we can always convert extra storage units (if more than  $f(M)$ ) to transport electrodes without increasing the chip size. Second, assume  $M > 1$  in the smallest chip. The corresponding transportation paths should connect all mixers and storage units so that they are reachable from each other. The size of mixers and storage units is  $M \cdot S_{mix} + f(M) \cdot S_{store}$ ; we assume the number of electrodes for input/output units are determined by the analysis tree and therefore constant. Since the size of a mixer is bigger than that of a storage unit, we can retain one mixer and change all other mixers to be storage units. The transportation paths can still connect all mixers and storage units, and the size of mixers and storage units is now reduced to  $1 \cdot S_{mix} + (M - 1 + f(M)) \cdot S_{store}$ . The size of one mixer and  $f(1)$  storage units is  $1 \cdot S_{mix} + f(1) \cdot S_{store}$ . From Corollary 2,  $M + f(M) \geq 1 + f(1)$ , which implies the chip with  $M$  mixers is larger than the chip with one mixer, leading to a contradiction. We conclude that even considering the transportation paths, the smallest chip is constructed using one mixer and  $f(1)$  storage units. ■

### 5.4 Example

Consider a complete tree of depth 4. By Theorem 2,  $f(1)$  for the complete tree is 3. Suppose that mixing and splitting droplets is performed in a mixer with 3 electrodes for droplets to move as shown in Figure 7 (a). Since we can place the mixer and storage units along the perimeter of the chip and overlap a subset of the surrounding electrodes, the size of a mixer and a storage unit can be smaller than that shown in Figure 7. Figure 10 shows the layout of the smallest chip, without showing all input and output units.



**Fig. 10.** The partial layout of the smallest chip for a complete tree of depth 4.

## 6 Conclusion

In this paper, we focused on characterizing the resource requirements of arbitrary biochemical analyses on DMFS biochips from their tree structure. We designed a corresponding scheduling algorithm to perform the biochemical analyses using the least resource requirements. We also defined the  $M$ -depth of an analysis tree to describe such resource requirements and infer the variation in the number and size of resources as a function of  $M$  and the tree structure. We can use these results to design the smallest biochip for any biochemical analysis and also determine whether a particular biochip can be used for a biochemical analysis. These results represent our initial steps towards automated design of digital microfluidic biochips. In our future work, we will use these results to explore the tradeoff between resource requirements and the completion time of biochemical analyses, and combine these results with automated layout design and routing algorithms for DMFS biochips.

## Acknowledgments

This work was supported in part by NSF under Award Nos. IIS-0713517, IIS-0730817, and CNS-0709099.

## References

1. K.-F. Böhringer. Modeling and controlling parallel tasks in droplet-based microfluidic systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(12):1463–1468, Dec. 2001.
2. X. X. Chen, H. K. Wu, C. D. Mao, and G. M. Whitesides. A prototype two-dimensional capillary electrophoresis system fabricated in poly(dimethylsiloxane). *Anal. Chem.*, 74:1772–1778, 2002.

3. 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.*, 12(1):70–80, Feb. 2003.
4. 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*, 25(2):329–339, Feb. 2006.
5. P. Dittrich and A. Manz. Lab-on-a-chip: microfluidics in drug discovery. *Nature Reviews Drug Discovery*, 5(3):210–218, Mar. 2006.
6. 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)*, pages 779–782, 2003.
7. E. J. Griffith and S. Akella. Coordinating multiple droplets in planar array digital microfluidic systems. *International Journal of Robotics Research*, 24(11):933–949, Nov. 2005.
8. 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*, 25(2):340–352, Feb. 2006.
9. Y. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, pages 406–471, 1999.
10. L. Luo and S. Akella. Optimal scheduling for biochemical analyses on digital microfluidic systems. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3151–3157, Oct. 2007.
11. P. Paik, V. K. Pamula, and R. B. Fair. Rapid droplet mixers for digital microfluidic systems. *Lab on a chip*, 3:253–259, Sept. 2003.
12. M. G. Pollack, R. B. Fair, and A. D. Shenderov. Electrowetting-based actuation of liquid droplets for microfluidic applications. *Appl. Phys. Lett.*, 77(11):1725–1726, Sept. 2000.
13. 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*, 5(3):310–315, Mar. 2004.
14. F. Su and K. Chakrabarty. Architectural-level synthesis of digital microfluidics-based biochips. In *Proc. IEEE International Conference on CAD*, pages 223–228, 2004.
15. F. Su and K. Chakrabarty. Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips. In *Proc. IEEE/ACM Design Automation Conference*, pages 825–830, 2005.
16. F. Su and K. Chakrabarty. Module placement for fault-tolerant microfluidics-based biochips. *ACM Transactions on Design Automation of Electronic Systems*, pages 682–710, 2006.
17. 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*, pages 323–328, Munich, Germany, Mar. 2006.
18. B. Zheng, C. Gerdtts, and R. F. Ismagilov. Using nanoliter plugs in microfluidics to facilitate and understand protein crystallization. *Current Opinion in Structural Biology*, 15:548–555, 2005.