

Eric J. Griffith*
Srinivas Akella

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12180, USA
griffe@cs.rpi.edu
sakella@cs.rpi.edu (corresponding author)

Coordinating Multiple Droplets in Planar Array Digital Microfluidic Systems

Abstract

In this paper we present an approach to coordinate the motions of droplets in digital microfluidic systems, a new class of lab-on-a-chip systems for biochemical analysis. A digital microfluidic system typically consists of a planar array of cells with electrodes that control the droplets. The primary challenge in using droplet-based systems is that they require the simultaneous coordination of a potentially large number of droplets on the array as the droplets move, mix, and split. In this paper we describe a general-purpose system that uses simple algorithms and yet is versatile. First, we present a semi-automated approach to generate the array layout in terms of components. Next, we discuss simple algorithms to select destination components for the droplets and a decentralized scheme for components to route the droplets on the array. These are then combined into a reconfigurable system that has been simulated in software to perform analyses such as the DNA polymerase chain reaction. The algorithms have been able to successfully coordinate hundreds of droplets simultaneously and perform one or more chemical analyses in parallel. Because it is challenging to analytically characterize the behavior of such systems, simulation methods to detect potential system instability are proposed.

KEY WORDS—digital microfluidic system, lab-on-a-chip, droplet coordination, layout design, routing

1. Introduction

The field of biochemical analysis systems has been revolutionized recently by the creation of miniature biochemical analysis systems using microfabrication technology. These systems are often termed “micro total analysis systems” or

“lab-on-a-chip” systems. These systems offer a number of advantages, including size reduction, power reduction, and increased reliability. However, current systems are typically tailored to a specific task. Therefore, an important goal is to create reconfigurable and reprogrammable systems capable of handling a variety of analysis tasks.

Digital microfluidic systems (DMFS) that use techniques such as electrowetting and dielectrophoresis are promising candidates for reconfigurable systems (Pollack, Fair, and Shenderov 2000; Jones et al. 2001; Cho, Moon, and Kim 2003; Paik, Pamula, and Fair 2003). We focus on microfluidic systems that manipulate discrete droplets by electrowetting, where the interfacial tension of the droplets is modulated with a voltage (Paik, Pamula, and Fair 2003). Droplets are microliters in volume, and have been moved at 12–25 cm s⁻¹ on planar arrays of 0.15 cm wide electrodes (see Cho, Moon, and Kim 2003 and Fair et al. 2003 for details). The ability to control individual droplets on a planar array enables complex analysis operations to be performed in biochemical “lab-on-a-chip” systems (Figure 1). For example, they can be used to perform polymerase chain reactions for DNA sequence analysis. 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. These systems have the potential to process hundreds of samples quickly. While there are important engineering challenges in fabricating and demonstrating the feasibility of these systems, the primary computational challenge with using droplet-based systems is developing algorithms for the simultaneous coordination of a potentially large number of droplets. Planning optimal paths through the array for each droplet would be computationally intractable for a large number of droplets.

In this paper we describe an approach to creating a general-purpose DMFS. First, we explain a semi-automated approach to design the array layout in terms of modular components, along with the motivating design choices. Next, we discuss simple algorithms to select destination components for the

*Eric Griffith is currently in the Computer Graphics Group at the Delft University of Technology, the Netherlands.

The International Journal of Robotics Research
Vol. 24, No. 11, November 2005, pp. 933-949
DOI: 10.1177/0278364905059067
©2005 SAGE Publications

Figures 2–6 and 10 appear in color online: <http://ijr.sagepub.com>

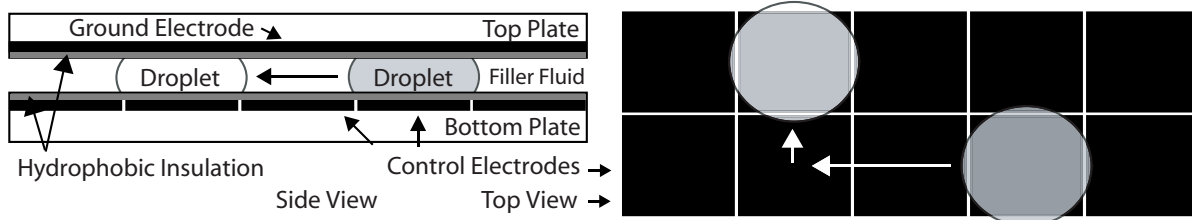


Fig. 1. Droplets on an electrowetting array (side and top views). The droplets are in a medium (usually oil or air) between two glass plates. The gray and white droplets represent the same droplet in two different positions. The gray droplet represents the droplet's initial position. Over a period of three clock cycles, the droplet is moved into the position represented by the white droplet. 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 Paik, Pamula, and Fair (2003).

droplets and a decentralized scheme for components to route the droplets. These are then combined into a reconfigurable system that has been simulated in software to perform DNA polymerase chain reaction and other analyses. The algorithms have been able to successfully coordinate hundreds of droplets simultaneously and perform one or more chemical analyses in parallel. Because it is challenging to analytically characterize the behavior of such systems, methods to detect potential instabilities due to congestion are proposed. An earlier version of this work was presented in Griffith and Akella (2005).

2. Related Work

Digital Microfluidic Systems. Pollack, Fair, and Shenderov (2000) demonstrated rapid manipulation of discrete microdroplets by electrowetting-based actuation. Ding, Chakrabarty, and Fair (2001) described an architectural design and optimization methodology for scheduling biochemical reactions using electrowetting arrays. They identified a basic set of droplet operations and used an integer linear programming formulation to minimize completion time. Droplet paths and areas on the array for storage, mixing and splitting operations are predefined by a human. Fair et al. (2003) describe experiments on injection, dispensing, dilution, and mixing of samples in an electrowetting DMFS. Cho, Moon, and Kim (2003) demonstrated creating, merging, splitting, and move operations using electrodes covered with dielectrics, and identified conditions under which these operations can be performed in an air environment. Fan, Hashi, and Kim (2003) developed a cross-reference grid of single layer electrodes to manipulate droplets with limited row-column addressing. Paik, Pamula, and Fair (2003) studied the effects of droplet aspect ratios and mixing strategies on the rate of droplet mixing. Jones et al. (2001) demonstrated dielectrophoresis-based liquid actuation and nanodroplet formation. Zhang, Chakrabarty, and Fair (2002) described hierarchical techniques for the modeling, design, performance evaluation, and optimization of microfluidic systems. In particular, they performed a compar-

ative performance analysis of a continuous flow system and a droplet-based system and showed that the droplet-based system has a less complex design that provides higher throughput and processing capacity. Our approach of imposing a layout on a digital microfluidics array to suit a given chemical reaction is similar to programming a reconfigurable field programmable gate array (FPGA; Maxfield 2004).

Our work is motivated by the above body of work, as well as the work of Böhringer and Donald (1998), who developed an algorithmic approach to the design and control of arrays of microelectromechanical systems (MEMS) actuators for coordinated micromanipulation tasks. Each droplet in a DMFS can be viewed as a simple robot that moves on a four-connected array (Böhringer 2003, 2004). Böhringer outlines an approach for moving droplets from start to goal locations, subject to droplet separation constraints, obstacles, and control circuitry limitations. He uses an A* search algorithm to generate optimal plans for droplets. To overcome the exponential complexity of this approach, he plans the droplet motions in prioritized order (Erdmann and Lozano-Perez 1987). However, note that in a DMFS the droplets must additionally sometimes combine for different durations to mix, and then be split.

Multiple Robot Coordination. The coordination of droplets in a DMFS is closely related to multiple robot motion coordination. However, very few motion planning algorithms can, with any guarantee, coordinate more than 10 or 20 robots. Hopcroft, Schwartz, and Sharir (1984) showed that even a simplified two-dimensional case of motion planning for multiple robots is PSPACE-hard. Reif and Sharir (1985) and Kant and Zucker (1986) developed techniques to plan motions of a single robot among moving obstacles. This can be generalized to obtain a heuristic solution for planning the motions of multiple robots. Erdmann and Lozano-Perez (1987) order robots by assigned priority and sequentially search for collision-free paths. Recent efforts have focused on probabilistic approaches. Švestka and Overmars (1998) developed a PRM planner for path coordination of multiple car-like robots.

Sanchez and Latombe (2002) use lazy PRM variants for coordinated path planning of multiple robot arms.

When the paths of the robots are specified, a path coordination problem, first studied by O'Donnell and Lozano-Perez (1989) for two robots, arises. LaValle and Hutchinson (1998) addressed a similar problem, where each robot was constrained to a C-space roadmap during its motion. Abrams and Ghrist (2002) have studied the topology of configuration spaces of robots restricted to motions on graphs. Simeon, Leroy, and Laumond (2002) coordinate over 100 car-like robots, where robots with intersecting paths are partitioned into smaller sets. Akella and Hutchinson (2002) developed a mixed integer linear programming (MILP) formulation for the trajectory coordination of 10–20 robots by changing robot start times. Peng and Akella (2005) developed an MILP formulation to coordinate many robots with simple double integrator dynamics along specified paths. Conflict resolution among multiple aircraft in a shared airspace (Tomlin, Pappas, and Sastry 1998; Bicchi and Pallottino 2000; Schouwenaars et al. 2001) is also closely related to multiple robot coordination.

Flexible Manufacturing Systems. Flexible manufacturing systems for product assembly have been modeled and analyzed using several techniques including Petri nets (Desrochers 1990). Of particular interest to flexible manufacturing systems is the issue of deadlock avoidance, which has been analyzed for certain classes of systems (Reveliotis, Lawley, and Ferreira 1997; Lawley 1999). Rizzi, Gowdy, and Hollis (2001) developed a reconfigurable, automated precision assembly system that uses cooperating, modular, robotic agents. Gowdy and Rizzi (1999) describe a model for programming such assembly systems. Inspired by this work, Klavins (2000) developed a method to automatically compile a representation of a distributed, hybrid factory from specifications of a product assembly.

Networking. We view the DMFS as a network. While this system differs from typical networking systems in non-trivial ways, techniques for network flow and rate control (Bertsekas and Gallager 1992; Tanenbaum 1996) may be modified for a DMFS. Methods from queueing theory and networking (Maxemchuk 1987; Brassil and Cruz 1991; Bertsekas and Gallager 1992; Gross and Harris 1998) can help analyze and improve the stability and performance of these systems. Related research in networking includes work on hot-potato or deflection routing (Choudhury and Li 1993; Busch, Herlihy, and Wattenhofer 2000) for different classes of networks, and work on rate control to ensure stability (Kelly, Maulloo, and Tan 1998).

3. Components for Array Layout

Many common operations for biochemical analyses can be performed on an array without additional special purpose

hardware. These operations include dispensing droplets onto the array, collecting droplets from the array, transporting droplets around the array, mixing droplets together, and splitting droplets apart. The array layout design presented here uses a system of “virtual” components. Each component type is responsible for performing one or more types of operation. Multiple instances of the same type of component can be present, and each component instance is allotted an area of the array to perform its operations in. By linking components that can perform all the operations in an analysis, a DMFS can be created to perform that analysis.

In our system, we have defined six component types. The “street”, “connector”, and “intersection” components transport droplets around the array. The “source” component adds droplets to the array, and the “sink” component removes droplets from the array. The “work area” component manages mixing and splitting of droplets. New component types can be defined and integrated into the system for operations that do not require special purpose hardware.

3.1. The Components

In this section we provide an overview of the functionality of each component type. Each component is responsible for managing all of the droplets that are in its portion of the array. At each clock cycle, components attempt to move all droplets in their section of the array. Each component maintains connections with its neighboring components, which are used to pass control of droplets when they move between components. The connections are entrance/exit pairs where the exit from one component is adjacent to the entrance in another. Components may have to wait to move their droplets until the components they are connected to have moved theirs. Figure 2 shows an example system with each of the component types present. The details of the layout itself are presented in Section 5. Figure 3 depicts the individual components.

Street Component. The street component is the general-purpose droplet transportation component. It moves droplets in one direction through at least two array cells. Streets are one-way to prevent two droplets from moving in opposite directions through the component. A street attempts to advance all droplets within it in synchrony at each clock cycle. If moving any droplet would cause it to be adjacent to another, it is not allowed to move. In Figure 3(a), the middle row of cells are the cells that droplets move through. The extra rows above and below this row provide part of the required buffer of empty cells around droplets.

Connector Component. The connector component is similar to a street component, except that a droplet only moves through a single cell in it (Figure 3b). The distinction is made because droplets in the connector are adjacent to two components simultaneously. Thus, when a droplet attempts to enter a connector, the connector must ensure that there is no other droplet adjacent to the connector.

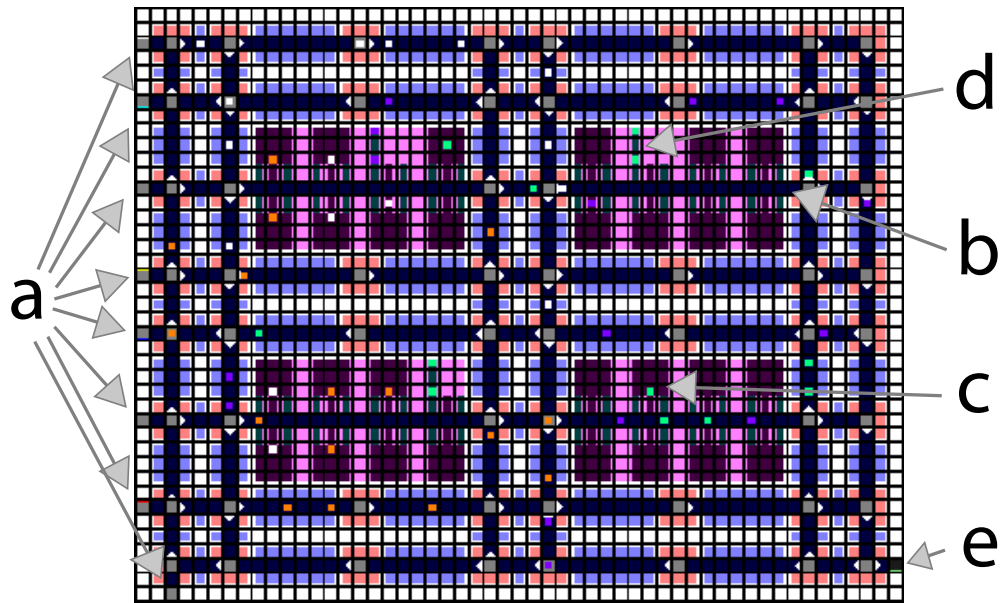


Fig. 2. Array layout for the DNA polymerase chain reaction (PCR) described in Section 7. Each cell of the layout is depicted as a square. On the left side of the array are (a) eight sources, which supply the input sample droplets to the system. There are (b) four work areas on the array, in which droplets are (c) mixed together and (d) split apart. In the lower-right corner of the array is (e) a sink, which collects the droplets of the final products and moves them off the array.

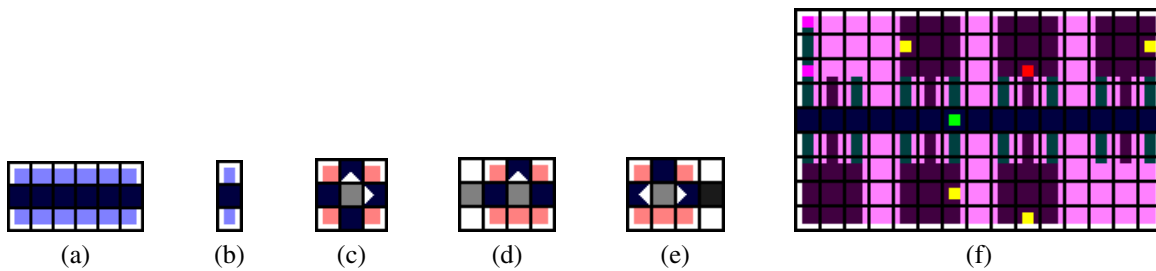


Fig. 3. The components. (a) A street component. Droplets move through the middle row of dark squares. (b) A connector component. (c) An intersection component with two entrances and two exits. The arrowheads indicate the exits. (d) A source component connected to an intersection component. (e) An intersection component connected to a sink component. (f) An active work area, showing several work units with droplets. Each work unit can function as a mixer and as a splitter.

Intersection Component. The intersection component is vital to the droplet routing system. Droplets enter an intersection and then move into its middle cell. Once there, the intersection routes the droplet to the appropriate exit based on the algorithm described in Section 4.3. In Figure 3(c), the exits from the intersection are denoted by white arrows, which are, in this case, up and right. The entrances to the intersection are from the left and from the bottom. The four corner cells provide part of the required buffer of empty cells around droplets.

Work Area Component. The work area component is where mixing and splitting take place. Each work area has a transit area and multiple “work units”. Each work unit may function as a “mixer” and/or as a “splitter”. Mixers merge two droplets into a new droplet. The mixer moves the droplet around to speed up mixing, as in Paik, Pamula, and Fair (2003). Splitters split the mixed droplet into two droplets. A work unit may be used as a mixer until a mixing operation is completed, and then part of that same unit may be used as a splitter. A work

area can mix and split multiple droplets at the same time. When a droplet gets to a work area, the work area sends it to a specific work unit. The work units manage the droplets assigned to them. Once a mix and split operation is complete, the resulting droplets are sent out of the work area.

In Figure 3(f), the middle row of cells are the cells that droplets move through when entering and exiting the work area. Droplets move along this row of cells until they reach the work units, which are both above and below this row. There are six mixer work units and one splitter work unit visible in the figure.

Source Component. The source component represents droplet entry points into the array. Each source introduces specified droplet types at specified intervals. Droplets entering the array are assigned a goal operation. In Figure 3(d), the source is depicted to the left of the intersection.

Sink Component. The sink component represents droplet exit points from the array. Each sink removes specified droplet types from the array. In Figure 3(e), the sink is depicted to the right of the intersection.

4. Droplet Destination Selection and Routing Algorithms

To be practical, a DMFS should be able to handle a large number of droplets simultaneously. Ideally, it should do this in a way that optimizes some quantity, such as throughput or completion time. We significantly reduce the computational cost of planning the droplet motions at the expense of optimality. By dividing the array into components, restrictions are placed on where operations can take place on the array. The interconnection of the components can be viewed as a network with the intersections as the routing devices and the streets and connectors as the “wires”. This reduces the motion planning problem to a network routing problem on a directed graph.

Selecting destinations for droplets is a twofold process. First, the droplet must be assigned an operation to perform. Once the droplet has an assigned operation, a component for that operation may be selected. The methods for these processes, as well as the droplet routing, are discussed in this section.

4.1. Assumptions

We assume the following when modeling the operations of the DMFS.

1. Individual cells of the array are addressable, permitting direct control of individual droplets.
2. No two droplets can occupy adjacent cells unless they are to be mixed.

3. Droplets moving simultaneously in the same direction along a line must have at least two empty cells between each other, and droplets moving simultaneously around a corner must have at least three empty cells between each other.
4. Droplets in the array have identical volumes, except during mixing.
5. Every mix operation is followed by a split operation. A split operation is performed by simultaneously activating the two electrodes on either side of the current cell.

4.2. Droplet Destination Selection

The system is supplied with parameters, described in Section 5, which it uses to maintain a list of components available for certain operations. Work areas can perform a mixing operation with any droplet type, and sinks remove specific types of droplets. Each work area and sink adds itself to an ordered list of components accepting droplets for operations. There is also an ordered list of higher priority containing requests from components for specific droplet types required to complete an operation. Currently, only work areas needing the remaining droplet type for a mixing operation place requests in this list.

When a new droplet enters the system, or is created through a mixing operation, the corresponding source or work area assigns it an operation. This operation is dependent on the system parameters. For example, if a droplet of type A is introduced and the analysis graph specifies that droplets of type A should be mixed with droplets of type B, then that droplet’s goal operation will be to mix with a droplet of type B. If the droplet is an end product of the chemical analysis, it will be assigned the goal operation of leaving the array via a sink.

When the droplet enters an intersection, the intersection tries to find a component to send the droplet to, as described in Algorithm 1. First, it checks the higher priority list for existing requests for the droplet’s type and assigned operation. If any exist, then the droplet is assigned to the first requesting component and that request is removed from the list. Using the above example, a droplet of type A would be assigned to a work area that already has a droplet of type B assigned to it, but does not yet have an assigned droplet of type A for it to mix with.

Should no component be actively requesting that droplet type for its operation, it is assigned to the first component on the lower priority list that can accept droplets of its type, and that component is sent to the end of the list (if it is a sink or is a work area with work units available). If no components are available, then the next intersection the droplet enters attempts to assign it a destination. In the worst-case scenario, this operation takes $O(w + s)$ time for one droplet at one intersection, where w is the number of work areas and s is the number of sinks in the system.

Algorithm 1. Select_Destination_Component.

Input: A droplet d for which a destination is to be selected.
Output: A destination component for d .
 $H \leftarrow \{h_1, h_2, \dots, h_n\}$ // Higher priority ordered list of
// components actively requesting droplet types.
 $L \leftarrow \{l_1, l_2, \dots, l_m\}$ // Lower priority ordered list of
// components accepting droplet types.
for $i = 1$ to n **do**
 if h_i .requested_type = d .type **and** h_i .operation =
 d .requested_operation **then**
 Remove h_i from H
 return h_i // Destination component found in H
 end if
end for
for $i = 1$ to m **do**
 if d .type \in l_i .accepted_type **and** l_i .operation =
 d .requested_operation **then**
 Remove l_i from L
 if d .requested_operation requires droplet type t **then**
 l_i .requested_type $\leftarrow t$
 Add l_i to the end of H
 end if
 if l_i can still accept droplets for l_i .operation **then**
 Add l_i to the end of L
 end if
 return l_i // Destination component found in L
 end if
end for
return \emptyset // No destination component found

4.3. Droplet Routing

After a droplet is assigned a destination component, the droplet must be routed to the component. The routing method we use can be viewed as a deflection routing variant of the Open Shortest Path First (OSPF) network protocol (Brassil and Cruz 1991; Tanenbaum 1996). The routing method relies on each intersection maintaining shortest path information to the other components, computed from the “component graph”. The component graph is a directed graph where each node is a component and each edge is a connection between adjacent components. The directed edge points from the component with the exit to the component with the entrance. The distance along an edge is, generically, taken to be the length of the component containing the exit for the edge.

When the system is initialized, each intersection computes and stores its routing table, which maps the shortest path to each component to a corresponding exit to take from the intersection. The specified exit is the first leg of the shortest path to the component in question. Each intersection constructs its routing table by running Dijkstra’s algorithm on the component graph to compute shortest paths from the intersection. The one restriction is that a shortest path should

not travel through work areas, sources, or sinks, unless those components are the destination. Dijkstra’s algorithm on a sparse graph using a standard implementation has a runtime of $O(n \log n)$ where n is the number of nodes in the graph. Because Dijkstra’s algorithm must be run from each of the intersections, the routing system has an initial overhead of $O(i \cdot n \log n)$ where i is the number of intersections in the array.

At each clock cycle, the intersections are processed (along with the other components) in a fixed order to select their droplet routing moves, as described in Section 5.1. The order could vary a little at each cycle based on droplet movement dependences. Subsequently, synchronous motion of droplets is executed. The droplet routing, described in Algorithm 2, is straightforward once the routing table has been constructed. If a droplet entering the intersection has no destination, then the intersection attempts to assign it one. If that fails, then the droplet is sent to a random, valid exit. Otherwise, the intersection finds the destination component in its routing table and selects the exit that is the best choice for the droplet. If the droplet is able to move toward that exit, it does so. Otherwise, the intersection randomly chooses a valid exit for the droplet. If no viable exit is available, then the droplet waits. The amount of time required for selecting an exit for a droplet is $O(1)$ because the intersection must check at most three possible exits and the routing table has constant access time.

Algorithm 2. Route_Droplet

Input: A droplet d which is to be routed by an intersection component c_i .
Output: An exit of the intersection component, if available, or indication of failure to find an available exit for d .
 $E \leftarrow \{e_1, \dots, e_n\}$ // The list of (up to 3) exits from the
// intersection.
 $R_i \leftarrow \{(c_1, e_{c_1}), (c_2, e_{c_2}), \dots, (c_m, e_{c_m})\}$ // The routing
// table for intersection i where component c_j has id j .
// e_{c_j} is the exit of component c_i that leads to the
// shortest path to component c_j .
if d .destination_component = \emptyset **then**
 d .destination_component = Select_Destination_Component(d)
end if
if d .destination_component $\neq \emptyset$ **then**
 $c_d \leftarrow d$.destination_component
 if Is_Free(e_{c_d}) **then**
 return e_{c_d}
 end if
 end if
 $e_{free} \leftarrow$ Choose_Random_Free_Exit()
 if $e_{free} \neq \emptyset$ **then**
 return e_{free}
 end if
 return \emptyset // No exit available, droplet d will wait at
 // intersection i this clock cycle.

5. A General-Purpose Digital Microfluidic System

A general-purpose DMFS can be created by combining the component-based layout design approach (Section 3) with the accompanying algorithms (Section 4). A general-purpose layout must be capable of handling arbitrary analyses that require the movement, mixing, and splitting of different types of droplets. Further, the layout should contain a sufficient number of work units and be able to efficiently transport droplets around the array.

Efficient droplet transportation is crucial because the usage of the system is not known in advance. Therefore, all parts of the array should be accessible. We group street components in pairs to simulate two-way streets, allowing droplets to closely follow the same path between two locations in both directions. We chose this particular two-way configuration, instead of enabling individual street components to operate in both directions, because of its larger capacity for droplet traffic and avoidance of collisions between droplets moving in opposite directions. To allow for intersections between these two-way streets, we group four intersections together in a rotary-like arrangement (see Figure 4).

We group these two-way streets and rotaries with a work area to form a pattern of components. This pattern, shown in Figure 5, is designed to provide a balance between number of work areas and ease of access. The array layout itself is a periodic tiling of this pattern, completed with an alternating sequence of rotaries and streets along its upper and right edges. The extra intersection components in the two-way streets are for connecting sources and sinks around the perimeter of the layout. The intersections in the vertical streets are connected to provide shorter paths to and from work areas.

To generate the layout, the user must specify a set of parameters dependent on the hardware. These are the physical size of the array and the locations of sources and sinks. To fully define the system, the user specifies parameters based on the chemical analyses to be performed, including the chemical analysis graph, the types of droplets introduced at sources, when and how often they are produced, the types of droplets to send to the sinks, information about the various intermediate operations to perform with the droplets on the array, and the number of work units to include in each work area.

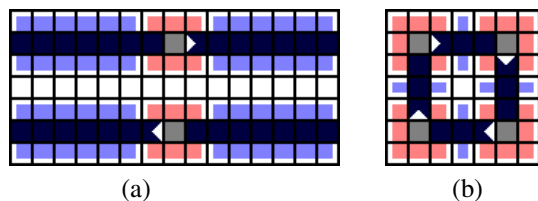


Fig. 4. Simulating two-way transportation: (a) two-way street; (b) rotary.

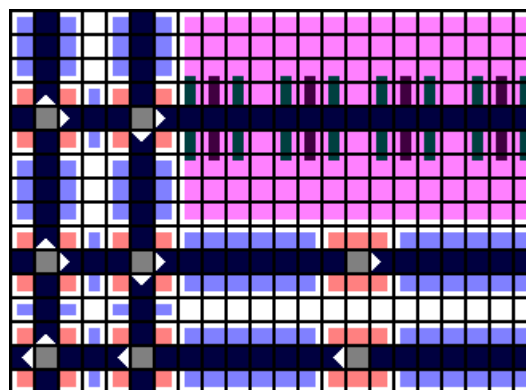


Fig. 5. The layout pattern tile that is a modular building block for the array.

The size of the array is limited to be of width $9 + (7 + w)i$ and height $9 + 16j$, where the layout is i tiles wide, j tiles tall, and w is the width of the work area. For the default case of eight work units per work area, w is 15 and the pattern tile is 22 cells wide and 16 cells tall. The sequence of two-way streets and rotaries to complete the layout adds an additional seven units. The sources and sinks must be placed around the perimeter of the array such that they can be connected to an intersection. Setting aside one cell around all sides of the perimeter of the pattern for sources and sinks gives the remaining two units. A complete example layout with eight sources and one sink can be seen in Figure 2.

5.1. Digital Microfluidic System Control

The approach to a DMFS described here yields a collection of communicating components organized into a network. The components cannot operate wholly independently because they must respect the property that no two droplets can be in adjacent array cells unless they are about to be mixed. Additionally, droplets moving simultaneously must keep at least two or three empty cells between each other if they are moving in a line or around a corner, respectively. Components only have knowledge of droplets within their own portion of the layout. Thus, before moving droplets into a neighboring component or into cells bordering a neighboring component, the component must consult the neighbor to ensure it would not result in two droplets being too close together.

For maximum efficiency, a DMFS should be able to move an arbitrary number of droplets, in parallel, at each clock cycle. Consider a component attempting to move one of its droplets, droplet A, into an array cell adjacent to one of its neighbors. The simplest case is when there is no droplet in the neighbor that prevents droplet A from being moved. The more complicated case occurs when there is a droplet B in

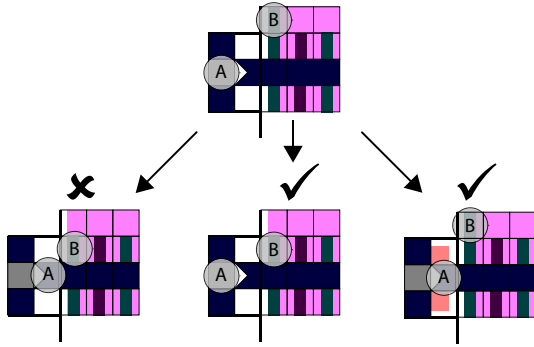


Fig. 6. A situation where two droplets A and B, each in a different component, require knowledge of the next state of the other droplet before knowing if they can move simultaneously. If both droplets move, then they will be diagonally adjacent to each other, which is unacceptable. Therefore, only one of the two droplets may move.

the neighbor component, which in its current position does not prevent the desired movement of droplet A, but droplet B may move into a cell that would prevent droplet A's movement. In this case, droplet A can be moved only if droplet B is not moved into the blocking position. See the example in Figure 6. A similar case arises when two different droplets wish to enter the same component, such as an intersection. In these situations, the component needs knowledge of the next state of the droplets in its neighbor in order to decide if it can move its droplets.

The solution to this problem we have chosen is to first process the components serially at each clock cycle and then perform motion in parallel. An ordered master list of components is maintained, and, at each clock cycle, each component in the list is instructed to attempt to move its droplets (Algorithm 3). The order is generally sources and work areas first and then the remaining components; the order could vary a little at each cycle based on droplet movement dependences. When a particular component wishes to move a droplet into an array cell adjacent to a neighbor component or into the neighbor component, it first asks that neighbor component if the move will result in two droplets being adjacent. If it will, then the component asks the neighbor to attempt to move its droplets so that the droplet is free to move in the next cycle. If the move will still result in adjacent droplets, then it waits to move those droplets that would cause violations (see Algorithm 4). A separate master list is kept containing the current position of all droplets and their desired position in the next clock cycle. When components move the droplets within themselves, the master list of droplets is updated to reflect the new current and desired positions of the droplets. The set of consistent droplet movements can then be collected so motion can be performed in parallel.

Algorithm 3. Move_All_Droplets

Input: C , the ordered list $\{c_1, c_2, \dots, c_n\}$ of all components.
for $i = 1$ to n **do**
 c_i .moved \leftarrow FALSE
end for
for $i = 1$ to n **do**
 c_i .Move_Droplets()
end for

Algorithm 4. Move_Droplets

Input: The component c moving the droplets.
 $N \leftarrow \{n_1, n_2, \dots, n_m\}$ // Neighboring components of c .
 $D \leftarrow \{d_1, d_2, \dots, d_p\}$ // The droplets in component c .
if c .moved **then**
 return
end if
 c .moved \leftarrow TRUE
for $i = 1$ to p **do**
 if d_i wants to move next to neighbor component n_j **then**
 if n_j .Can_Move_Adjacent(d_i) **then**
 Move(d_i)
 else
 n_j .Move_Droplets()
 end if
 else if d_i wants to move into neighbor n_j **then**
 if n_j .Can_Accept_Droplet(d_i) **then**
 Move(d_i)
 else
 n_j .Move_Droplets()
 end if
 else
 Attempt_To_Move(d_i) // component attempts to plan
 // motions for droplet
 end if
end for

5.2. Cell Addressing

In general, we assume that individual cells can be directly addressed by the hardware controlling mechanism for the DMFS. However, row-column addressing schemes, where only entire rows and columns can be addressed and only cells at the intersection of activated rows and columns are activated, are also feasible (Fan, Hashi, and Kim 2003; Böhringer 2004). This approach simplifies the hardware, but provides less flexibility in moving several droplets in synchrony, especially when the row-column activations interfere to cause undesired droplet movement, merging, or splitting. We expect that our algorithms can be modified to handle row-column addressing. The issue here is how to serialize the previously synchronous motion of the droplets at each clock cycle. By

identifying those droplets whose movements may interfere with each other, we can identify sets of droplets that can be safely moved simultaneously.

5.3. Digital Microfluidic System Operation

The DMFS can be operated in two modes, as follows.

1. Batch mode. Here, all the droplets necessary for an analysis are input at the source components in one batch. The droplets are coordinated to complete the analysis, and then the next batch of droplets is processed. The droplets are input in a synchronized manner based on when they are required for the reactions. After each mix and split operation, one of the two resulting droplets is sent to a waste output. An analysis performed in batch mode will typically require a smaller number of tiles in the array since the number of droplets in the system is small.

One advantage of the batch mode is that the droplet routing is almost entirely deterministic, and we can easily analyze the system behavior. Only when no work units are available for a droplet will an intersection component route the droplet randomly.

2. Continuous mode. Here, the source components input the droplets at a fixed rate. (The rate for each droplet type is specified by the human designer.) One advantage of this mode is that it produces a larger volume of product droplets than the batch mode in the same amount of time, especially when no droplets are discarded as waste droplets. A potential disadvantage is that system behavior is harder to analyze. We explore this issue, and in particular, potential system instability, in Section 6.

In both modes, multiple analyses yielding different products can be processed in parallel to better utilize the array.

6. System Stability

The behavior of a general-purpose DMFS changes with the chemical analysis it performs. The stability of a system operating in continuous mode depends on its parameters, especially the input flow rates. In an unstable system, droplets enter the system faster than the system is able to process them, and a steady-state flow may cease (Gross and Harris 1998). If a system is not stable, in time it will become heavily congested and may finally become deadlocked. A system is deadlocked when droplets are unable to reach their destinations.

We treat congestion as an indicator of instability, and try to identify conditions that lead to or result from congestion. Deadlock, which is the result of severe congestion, is a sufficient condition for a system to be unstable. At least one of two conditions must hold for a system to become congested.

The first is for some droplets to be unable to follow the shortest paths to their destinations. The second condition is for droplets to be unable to be assigned a destination. Either condition is an indicator that the system is not processing droplets fast enough. To identify when these conditions may occur, we employ two methods. At the operation level, we analyze a graph of the operations to be performed based on the system parameters. At the component level, we model droplet flow in the system using the component graph. Using the information from these methods, we are able to design the system to be more stable by selecting system parameters such as input droplet rates and source and sink locations to avoid these conditions.

6.1. Analysis Graph

The (biochemical) “analysis graph” provides a representation of the operations of the system. 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. Each node stores the duration of its operation, and the analysis graph is augmented with additional information (Figure 7).

The first augmentation is the rate at which droplets will enter and leave each node. Droplet rate can be most intuitively described in terms of a source. If a source introduces a particular droplet type once every k cycles, then its rate r would be $1/k$ droplets per cycle. Also, there would be $k - 1$ empty cells between each introduced droplet. This rate is propagated through the graph. If these droplets perform a mixing operation, then, on average, and assuming no unusual delays, one of these droplets would begin its mixing operation every k cycles. Similarly, one droplet would complete the mixing operation every k cycles.

The second augmentation is the best-case expected distance to travel between performing operations. Each node's operation can be performed at one or more components on the array; specific components are chosen during destination selection for the droplets. The expected distance is the average of the shortest path between each possible originating component and each possible destination component. We use this definition because destination components are assigned on a rotating basis, and a droplet leaving a component is equally likely to be assigned to any of the possible destination components. In a system with two work areas and a source component introducing droplets for mixing, the distance from the input node to the mix node would be the average of the lengths of the shortest paths from the source to each of the two work areas.

The third augmentation is the best-case expected time at which the first droplets will enter each node. There is a separate arrival time from each parent node of a given node, and

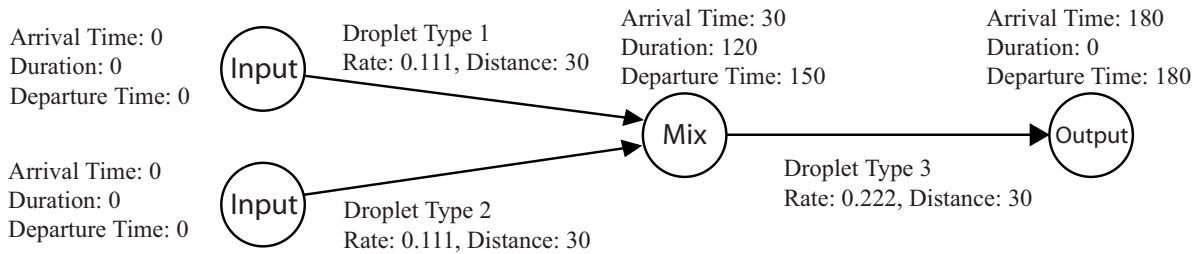


Fig. 7. A simple analysis graph with four nodes: two input nodes, one mix node, and one output node. Droplet types, rates, and expected distances traveled are indicated along the edges. Expected arrival time, duration, and departure time (in units of clock cycles) are indicated at the nodes. Times (and distances) are in units of clock cycles, and rates are in terms of number of droplets per clock cycle.

it is computed as the expected departure time from the parent node plus the best case expected travel time from the parent node to the current node. The expected departure time from a given node is the average of the arrival times, plus the duration of the operation.

The graph is then checked to ensure that the following properties hold:

1. every path from every source node must lead to a sink node;
2. droplets must enter a node at the same rate from each parent node;
3. arrival times to a node should be about the same from each parent node.

If any of these properties are violated, this is likely to result in inefficient processing of droplets. In the event that the properties are violated, the system parameters can be adjusted until they result in a graph without violations.

6.2. Droplet Flow Analysis

The analysis graph provides a reasonable estimate of the overall system stability. However, a more detailed component level analysis of system stability may be required. There may be certain bottleneck components that become congested from too many droplets moving through them, and so slow down droplet flow. These bottlenecks can result in an unstable system when they prevent droplets from reaching their destinations. Some simple experiments have demonstrated that this can arise in larger arrays (for example, arrays of size 295×297), where there is a lot of droplet traffic around the perimeter near the sources and sinks but relatively light traffic in the interior. Modeling the droplet flow through the system can be an effective tool to identify unstable systems, especially those that are unstable due to these bottleneck, congested components.

The droplet flow modeling attempts to predict the expected flow through the component graph (described in Section 4.3)

using the rate information from the analysis graph (see Figure 8). The idea is to determine the rate at which droplets enter and leave each component on the array, which is the flow rate through that component, when the system is in a steady state. To approximate the expected flow rate through components, we developed an iterative analysis. For this analysis, we make the assumption that work areas are uniformly utilized. That is, a droplet being assigned to a work area is equally likely to be assigned to any work area on the array.

Each component is initially assigned inflow and outflow rates of 0. Sources generate a certain amount of flow, dictated by the analysis graph, destined for each work area. For example, if a source produces droplets at a rate of $1/4$ in a system with two work areas, it generates a flow of $1/8$ to each work area. Similarly, work areas generate a certain amount of flow destined for each work area and to each sink. At each iteration, the output of each node in the graph is defined as a function of the input to the node. The input of a node is the sum of the outputs, from the previous iteration, of its parent nodes plus any flow it generates at that iteration. For intersections, the input flow is divided amongst the possible exits based on where the flow is destined. If a particular node becomes congested, nodes sending flow to it try to redirect excess flow away.

Nodes in the graph are assigned a maximum inflow capacity. For all nodes except work areas, this is set as $1/4$, which is the maximum rate that droplets may make a turn through an intersection without inadvertently merging. The maximum inflow rate to a work area is computed based on the duration of its operations in the analysis graph. For example, consider a system where mixing operations take 100 cycles to complete. Each mixing operation requires two droplets and each work area can support eight simultaneous mixing operations. If the droplets entered a work area at a rate of one droplet every seven cycles, the first mixing operation should complete shortly after, or even before, the eighth operation begins. Droplets would likely not be able to enter the work area at a faster rate than this because there would not be any free work units. To respect the maximum inflow rates, parent nodes must adjust their outflows, which may result in some of the inflow

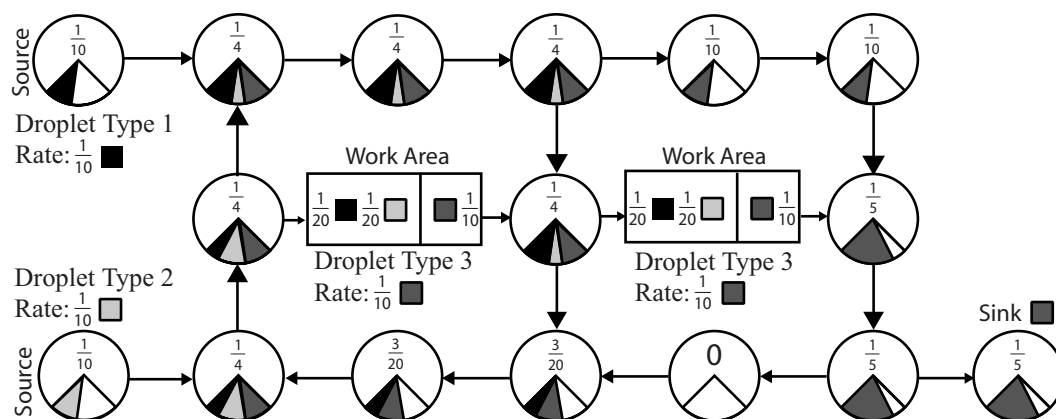


Fig. 8. A component graph for a system with the analysis graph in Figure 7 depicting the steady-state flow through the system. For clarity, this component graph uses a simplified layout. Each circular node contains a pie chart for flow through the component. The actual flow is the shaded portion of the maximum flow rate of $1/4$ in each node. In this example, inflow is equal to outflow. The total flow through each component is depicted by the fraction in the node. The work areas receive flow from both sources and produce flow destined for the sink. The rate of flow and droplet type (color) produced by each source and work area is indicated by the node.

not translating into outflow. A congested node is defined as a node with inflow greater than outflow.

Once the computed flow through the system converges, the component graph is analyzed. If the system is stable, the inflow will equal the outflow at every street, connector, work area, and intersection node. Otherwise, the system is likely to be unstable. A simplified example system with steady-state flow information is depicted in Figure 8. The system flow equations are more effective in estimating the steady-state behavior of the system than for the analysis of transient behavior and unstable behavior of the system.

7. System Simulation

We have simulated example systems operating in batch mode and in continuous mode. We now describe an example system based on the DNA polymerase chain reaction (PCR) operations outlined in Ding, Chakrabarty, and Fair (2001). The reaction involves eight input droplet types and seven mixing operations. See Figure 9 for an analysis graph of the system. (Note that the PCR reaction requires heating once the sample is prepared. We assume that droplets may be routed off-chip for heating.) Immediately following each mixing operation, the resulting droplet is split into two droplets. The layout is set up with four work areas, eight sources each introducing an input droplet type, and one sink to collect the final product (Figure 2). The size of the layout in this example is 53×41 cells. The system parameters were set with the aid of the stability analysis described above. In continuous mode, the system has an average of 66 droplets on the array. The routing computations for this array with 2×2 tiles are performed at a

rate of about 60,000–70,000 cycles a second, enabling rapid simulation of the system to verify stability. Animations of the PCR analysis, in both batch mode and continuous mode, as well as animations of multiple analyses in parallel, can be seen at <http://www.cs.rpi.edu/~sakella/microfluidics/> and in Appendix A.

When the system is in its stable operating range, there is a linear relation between the input droplet rate and output droplet rate, since no droplets are accumulating on the array (Figure 9a). Once a critical input rate is exceeded, there is a rapid drop off in the number of cycles at which instability occurs (Figure 9b). Note that the “input rate” referred to in these and subsequent graphs is the rate at which each of the four chemicals on the left of Figure 9 are introduced. The subsequent input chemicals are introduced at correspondingly higher multiples of the input rate.

7.1. Simulation Analysis

Our most reliable tool for predicting system stability is the implemented software used as a simulator. On a 1.7 GHz Pentium machine, it is possible to simulate 10,000–20,000 cycles per second when dealing with an array that is 3×3 tiles in size. Even assuming a very rapid actual operation speed of the order of 50–100 Hz, we are able to simulate 24 hours of operation in approximately 6 minutes. Significantly less time is usually needed to identify unstable systems because they tend to result in deadlock quickly. In fact, we can use binary search to rapidly identify the input rate at which instability starts to occur.

The simulation approach has also provided us with insights into the behavior of the system. We have observed sharp

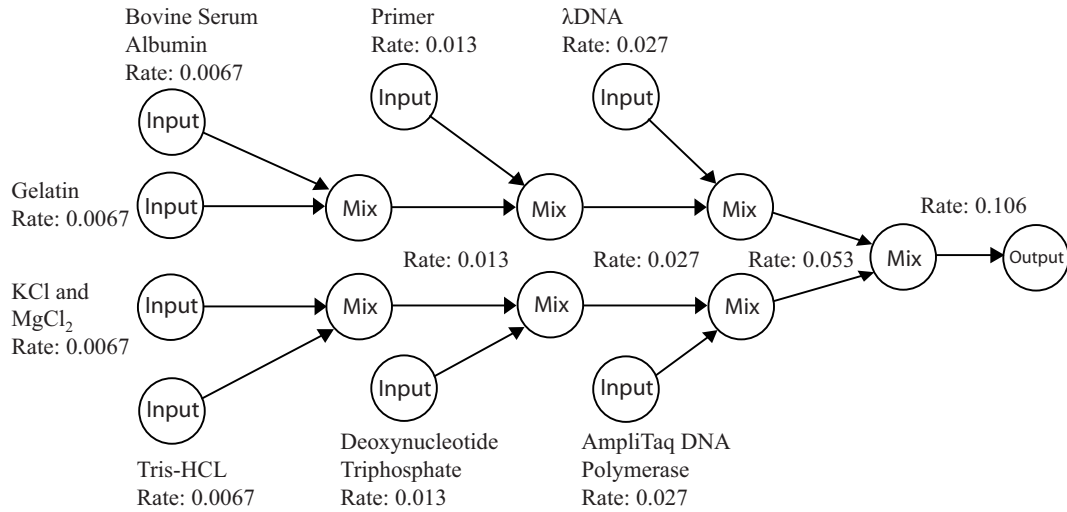


Fig. 9. PCR analysis graph. Input nodes are labeled with the samples they introduce and the rate at which they introduce them. Edges out of mix nodes are labeled with the droplet rate resulting from the operation.

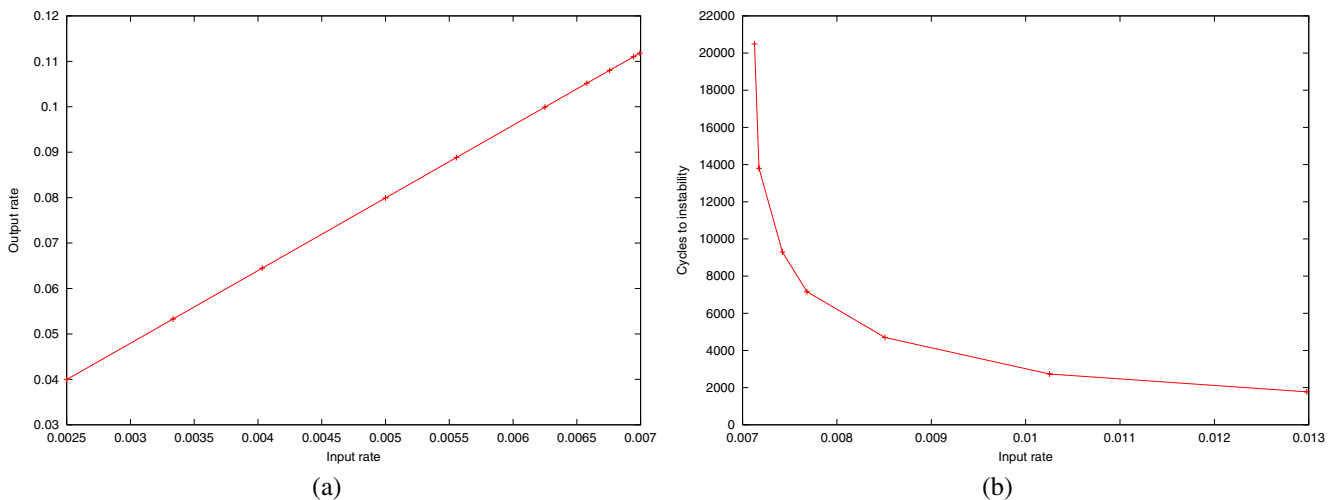


Fig. 10. Simulation data for the PCR reaction illustrating (a) the variation of droplet output rate with input rate in the stable range, and (b) the number of cycles at which the system goes into deadlock, as input rate is increased in the unstable range. For this example, mixing time is 128 cycles, the number of work units per work area is 8, and the tiles are in a 2 × 2 pattern.

variations in behavior when simulating systems that are on the borderline between stability and instability. Small changes in the rate at which droplets enter the system can mean the difference between becoming deadlocked in 5,000 cycles, becoming deadlocked in 2,000,000 cycles, or running continuously for 10,000,000 cycles without deadlock. Figure 11 shows the number of cycles at which deadlock occurs, as the input droplet rate is varied.

In the same tests, we examined the effects of varying the number of work units in each work area for a layout with a

fixed number of tiles. As expected, increasing the number of work units permits the system to be stable at a higher input rate. However, Figure 11 indicates that as the input rate increases, the effectiveness of maintaining system stability by increasing the number of work units decreases. This suggests that the performance bottleneck is shifting from the work areas to the transportation components as the total number of work units in the array increases.

Because larger arrays are able to handle higher input rates, we compared the area-normalized droplet output rate (com-

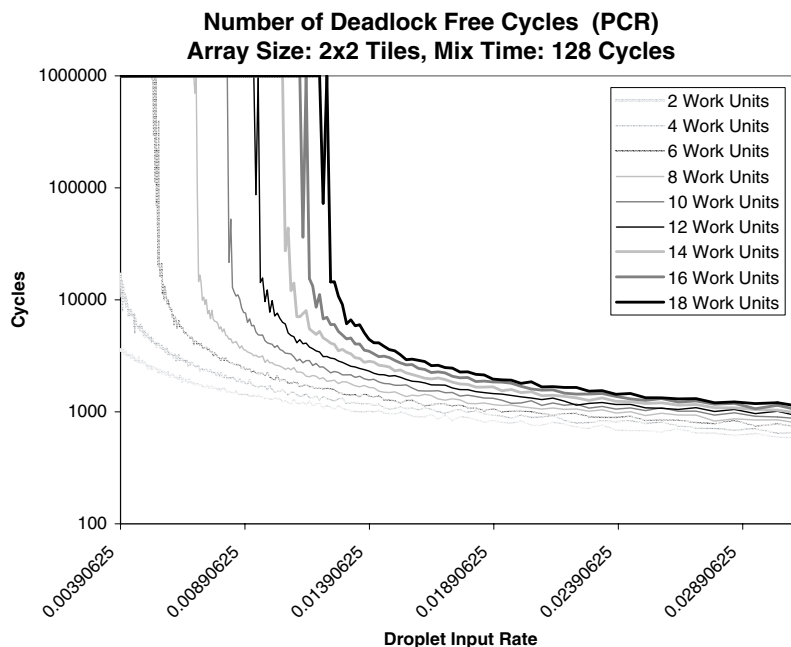


Fig. 11. Number of cycles at which deadlock occurs, as input droplet rate is varied. Each curve corresponds to a different number of work units in each work area. Note that the number of cycles is plotted on a logarithmic scale, and the simulations were run to a maximum of one million cycles. Simulation data are for the PCR reaction.

puted as droplet output rate divided by number of cells in array) as the number of work units is increased (Figure 12). The maximum peak area-normalized output rate occurs with 12 work units per work area. However, the peak value occurs on the borderline between stability and instability. When considering the peak value for 12, 14, 16, and 18 work units, there is a slight downward trend, which suggests that there is a limit to the benefit of adding more work units to the array. In general, the smallest array size for which a given rate is stable yields the best value. The benefit of adding more work units is stability at higher input rates.

To gauge the performance of the array in terms of the work area utilization, we recorded the states of the individual work units during the course of the simulation. For each work unit, there are three possible states: empty, reserved, and active. An empty work unit has no droplets in it (or assigned to it). A reserved work unit has one droplet in it (or assigned to it) and is waiting for another to arrive for a mixing operation. A work unit is active at all other times, as it performs mix or split operations. Figure 13 shows a plot of the average percent of work units in each state for varying input rates. There is a sharp change in the pattern when the system transitions from stable to unstable. When the system is stable, the percentage of empty work units indicates the stress level of the system. In stable systems near the point of instability, the percentage of reserved work units serves as an indicator of what is preventing the system from being able to handle higher input rates.

If the percentage of reserved work units is low, then droplets are moving quickly to work areas, and the performance bottleneck is a shortage of work areas. On the other hand, if the percentage is high, then droplets are having to travel too far to reach their assigned work area, which indicates the bottleneck is either insufficient transportation capacity or inefficient droplet transportation around the array.

8. Conclusions and Future Work

We have described a new approach to creating a general-purpose DMFS by partitioning the planar array into a collection of virtual components and coordinating the motions of droplets by implementing a decentralized routing algorithm. We have explored techniques to identify potentially unstable systems, and applied them to successfully demonstrate a stable DNA polymerase chain reaction in simulation.

The system described here can semi-automatically generate a layout given a set of system parameters, and then perform real-time droplet manipulation. It has successfully coordinated hundreds of droplets, and is a proof of concept that decentralized network-like motion planning can work for a digital microfluidic system. Further, the software can be easily modified to act as a controller for a physical array. The same array can perform a variety of chemical analyses, and has been demonstrated to even perform multiple analyses in parallel.

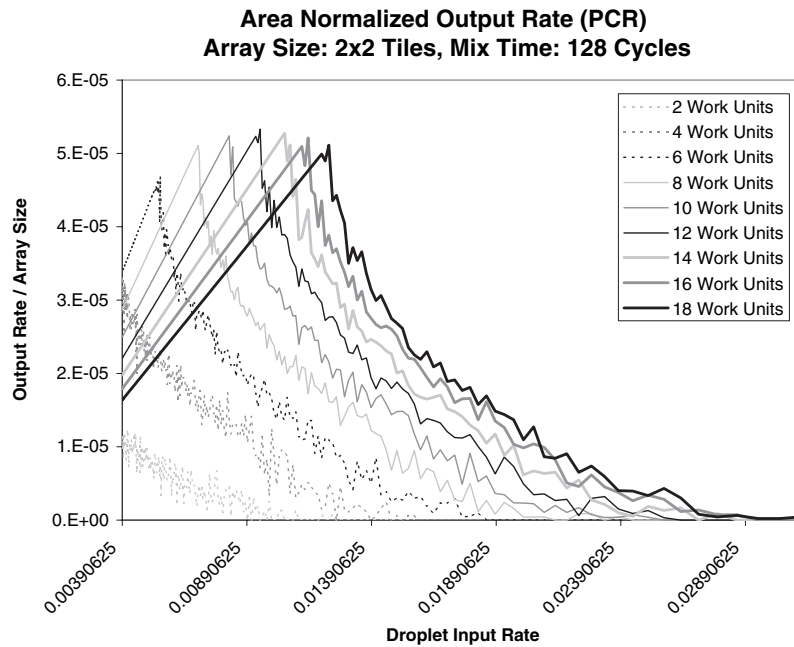


Fig. 12. The plot of area-normalized output rate (output rate/array size) as droplet input rate is varied illustrates the effect of adding more work units. Simulation data are for the PCR reaction.

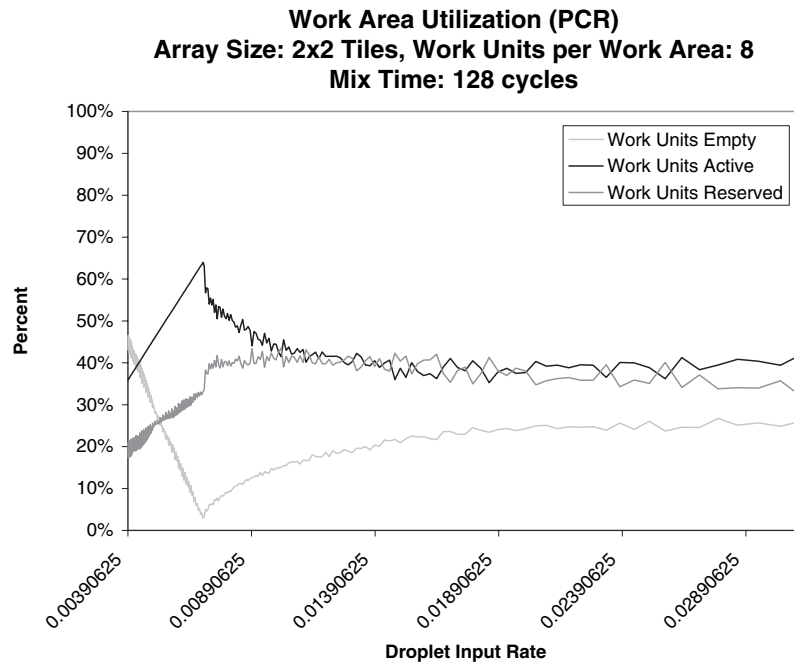


Fig. 13. The work area utilization, measured by the percentage of active, reserved, and empty work units, as input droplet rate is varied. Simulation data are for the PCR reaction.

The current system can be enhanced in a number of ways for greater flexibility and efficiency. The overall design of the components and the system allows for the introduction of new component types. For example, optical sensing components that permit monitoring of the reactions, and storage components, where droplets can be temporarily stored, can be incorporated. The system can also support simpler hardware that permits only limited row-column addressing of electrodes. Automatically sequencing operations to achieve desired droplet concentrations would be a useful extension. Automatically generating the array for a given reaction requires optimizing the number of tiles and their layout, as well as the locations of the sinks and sources on the array. Optimizing tile design to optimize droplet flow rates, and optimizing array layout for a given tile pattern are interesting problems.

Modeling the optimization of system throughput as a network routing and flow control problem and the stability problem as a deadlock avoidance problem can improve system performance and provide guarantees of stability. Further, they can provide insights into changes to the layout design. However, this droplet routing system differs from network routing in several ways. First, the system cannot discard droplets. Secondly, mix and split operations are more complex than packet routing operations. Thirdly, the system has multiple classes of nodes, and has rate constraints on both intersections and streets. Fourthly, only one droplet can enter an intersection at any instant, since there are no buffers for droplets. Therefore, even though the network graph for the system is relatively simple, analysis of the system is difficult.

Techniques for stability and performance analysis, drawing upon methods from queueing theory and networking (Maxemchuk 1987; Brassil and Cruz 1991; Bertsekas and Gallager 1992), can be applied to this system. Detailed analysis of the system can help identify components where congestion or deadlock may occur, and automatically set droplet input rates to avoid such problems. More sophisticated dynamic routing techniques can be explored to more effectively load balance the transport sections of the array and reduce bottlenecks at components. The design and control of fully reconfigurable arrays, where any part of the array may be used for any desired operation, pose particularly interesting challenges.

Appendix: Index to Multimedia Extensions

The multimedia extension page is found at <http://www.ijrr.org>. These animations include the PCR example run on the array in Figure 2. The layouts have two sinks along the right side. Each droplet is typically depicted as a small square, and the color of a droplet indicates its droplet type. When two droplets are mixed, the mixed droplet type is indicated by its unique color.

Table of Multimedia Extensions

Extension	Type	Description
1	Animation	PCR analysis, in batch mode. Waste droplets are depicted as blue diamonds.
2	Animation	PCR analysis, in continuous mode.
3	Animation	Two analyses running in parallel (continuous mode). Droplets of the PCR analysis are depicted as squares and droplets of the second analysis are depicted as diamonds.
4	Animation	Unstable PCR example (continuous mode), showing the system reaching a deadlock state.

Acknowledgments

Many thanks to Karl Böhringer for introducing us to this problem and providing encouragement and advice along the way. We benefited from discussions on network models with Bulent Yener, Biplab Sikdar, Costas Busch, and Jayasri Akella. This work was supported in part by the National Science Foundation under CAREER Award No. IIS-0093233.

References

- Abrams, A. and Ghrist, R. 2002. Finding topology in a factory: configuration spaces. *American Mathematical Monthly* 109(2):140–150.
- Akella, S. and Hutchinson, S. 2002. Coordinating the motions of multiple robots with specified trajectories. *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, May, pp. 624–631.
- Bertsekas, D. P. and Gallager, R. G. 1992. *Data Networks*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ.
- Bicchi, A. and Pallottino, L. 2000. On optimal cooperative conflict resolution for air traffic management systems. *IEEE Transactions on Intelligent Transportation Systems* 1(4):221–231.
- Böhringer, K. F. 2003. Optimal strategies for moving droplets in digital microfluidic systems. *Proceedings of the 7th International Conference on Miniaturized Chemical and Biochemical Analysis Systems (MicroTAS 2003)*, Squaw Valley, CA, October.
- Böhringer, K. F. 2004. Towards optimal strategies for moving droplets in digital microfluidic systems. *Proceedings of the*

- IEEE International Conference on Robotics and Automation*, New Orleans, LA, April.
- Böhringer, K. F. and Donald, B. R. 1998. Algorithmic MEMS. *Robotics: The Algorithmic Perspective*, P. K. Agarwal, L. E. Kaviraki, and M. T. Mason, editors, A. K. Peters, Natick, MA.
- Brassil, J. and Cruz, R. 1991. Non-uniform traffic in the Manhattan street network. *Proceedings of the IEEE International Conference on Communications (ICC'91)*, Denver, CO, June, pp. 1647–1651.
- Busch, C., Herlihy, M., and Wattenhofer, R. 2000. Hard-potato routing. *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000)*, Portland, OR, May, pp. 278–285.
- Cho, S. K., Moon, H., and Kim, C-J. 2003. Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits. *Journal of Microelectromechanical Systems* 12(1):70–80.
- Choudhury, A. K. and Li, V. O. K. 1993. An approximate analysis of the performance of deflection routing in regular networks. *IEEE Journal on Selected Areas in Communications* 11(8):1302–1316.
- Desrochers, A. A. 1990. *Modeling and Control of Automated Manufacturing Systems*, IEEE Computer Society, Washington, DC.
- Ding, J., Chakrabarty, K., and Fair, R. B. 2001. Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20(12):1463–1468.
- Erdmann, M. and Lozano-Perez, T. 1987. On multiple moving objects. *Algorithmica* 2(4):477–521.
- Fair, R. B., Srinivasan, V., Ren, H., Paik, P., Pamula, V., and Pollack, M. G. 2003. Electrowetting-based on-chip sample processing for integrated microfluidics. *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*, Washington, DC, December, pp. 779–782.
- Fan, S-K., Hashi, C., and Kim, C-J. 2003. Manipulation of multiple droplets on NxM grid by cross-reference EWOD driving scheme and pressure-contact packaging. *Proceedings of the IEEE Conference on MEMS*, Kyoto, Japan, January, pp. 694–697.
- Gowdy, J. and Rizzi, A. 1999. Programming in the architecture for agile assembly. *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, May, pp. 3103–3108.
- Griffith, E. and Akella, S. 2005. Coordinating multiple droplets in planar array digital microfluidics systems. *Algorithmic Foundations of Robotics VI*, In M. Erdmann, D. Hsu, M. Overmars, and A. F. van der Stappen, editors, Springer-Verlag, Berlin, pp. 219–234.
- Gross, D. and Harris, C. M. 1998. *Fundamentals of Queueing Theory*, 3rd edition, Wiley, New York.
- Hopcroft, J. E., Schwartz, J. T., and Sharir, M. 1984. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the “warehouseman’s problem”. *International Journal of Robotics Research* 3(4):76–88.
- Jones, T. B., Gunji, M., Washizu, M., and Feldman, M. J. 2001. Dielectrophoretic liquid actuation and nanodroplet formation. *Journal of Applied Physics* 89:1441–1448.
- Kant, K. and Zucker, S. W. 1986. Toward efficient trajectory planning: The path-velocity decomposition. *International Journal of Robotics Research* 5(3):72–89.
- Kelly, F., Maulloo, A., and Tan, D. 1998. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society* 49:237–252.
- Klavins, E. 2000. Automatic compilation of concurrent hybrid factories from product assembly specifications. *Hybrid Systems: Computation and Control, Lecture Notes in Computer Science* Vol. 1790, Springer-Verlag, Berlin, pp. 174–187.
- LaValle, S. M. and Hutchinson, S. A. 1998. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation* 14(6):912–925.
- Lawley, M. A. 1999. Deadlock avoidance for production systems with flexible routing. *IEEE Transactions on Robotics and Automation* 15(3):497–509.
- Maxemchuk, N. F. 1987. Routing in the Manhattan street network. *IEEE Transactions on Communications* 35(5):503–512.
- Maxfield, C. 2004. *The Design Warrior’s Guide to FPGAs: Devices, Tools, and Flows*, Elsevier, Burlington, MA.
- O’Donnell, P. A. and Lozano-Perez, T. 1989. Deadlock-free and collision-free coordination of two robot manipulators. *Proceedings of the IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, May, pp. 484–489.
- Paik, P., Pamula, V. K., and Fair, R. B. 2003. Rapid droplet mixers for digital microfluidic systems. *Lab on a Chip* 3:253–259.
- Peng, J. and Akella, S. 2005. Coordinating multiple robots with kinodynamic constraints along specified paths. *International Journal of Robotics Research*, 24(4):295–310.
- Pollack, M. G., Fair, R. B., and Shenderov, A. D. 2000. Electrowetting-based actuation of liquid droplets for microfluidic applications. *Applied Physics Letters* 77:1725–1726.
- Reif, J. and Sharir, M. 1985. Motion planning in the presence of moving obstacles. *Proceedings of the 26th Annual Symposium on the Foundations of Computer Science*, Portland, OR, October, pp. 144–154.
- Reveliotis, S. A., Lawley, M. A., and Ferreira, P. M. 1997. Polynomial-complexity deadlock avoidance policies for sequential resource allocation systems. *IEEE Transactions on Automatic Control* 42(10):1344–1357.

- Rizzi, A. A., Gowdy, J., and Hollis, R. L. 2001. Distributed coordination in modular precision assembly systems. *International Journal of Robotics Research* 20(10):819–838.
- Sanchez, G. and Latombe, J. 2002. On delaying collision checking in PRM planning – application to multi-robot coordination. *International Journal of Robotics Research* 21(1):5–26.
- Schouwenaars, T., De Moor, B., Feron, E., and How, J. 2001. Mixed integer programming for multi-vehicle path planning. *Proceedings of the European Control Conference*, Porto, Portugal.
- Simeon, T., Leroy, S., and Laumond, J.-P. 2002. Path coordination for multiple mobile robots: A resolution-complete algorithm. *IEEE Transactions on Robotics and Automation* 18(1):42–49.
- Švestka, P. and Overmars, M. 1998. Coordinated path planning for multiple robots. *Robotics and Autonomous Systems* 23(3):125–152.
- Tanenbaum, A. S. 1996. *Computer Networks*, 3rd edition, Prentice-Hall, Upper Saddle River, NJ.
- Tomlin, C., Pappas, G. J., and Sastry, S. 1998. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control* 43(4):509–521.
- Zhang, T., Chakrabarty, K., and Fair, R. B. 2002. *Microelectrofluidic Systems: Modeling and Simulation*, CRC Press, Boca Raton, FL.