

Scheduling of Microfluidic Operations for Reconfigurable Two-Dimensional Electrowetting Arrays¹

Jie Ding, Krishnendu Chakrabarty and Richard B. Fair
Department of Electrical and Computer Engineering
Duke University
130 Hudson Hall, Box 90291
Durham, NC 27708

Submitted: December 2000, Revised: August 2001

ABSTRACT

We present an architectural design and optimization methodology for performing biochemical reactions using two-dimensional electrowetting arrays. We define a set of basic microfluidic operations and leverage electronic design automation principles for system partitioning, resource allocation, and operation scheduling. Fluidic operations are carried out through the electrostatic configuration of a set of grid points. While concurrency is desirable to minimize processing time, the size of the two-dimensional array limits the number of concurrent operations of any type. Furthermore, functional dependencies between the operations also limit concurrency. We use integer linear programming to minimize the processing time by automatically extracting parallelism from a biochemical assay. As a case study, we apply our optimization method to the polymerase chain reaction, which is an important step in many lab-on-a-chip biochemical applications.

Keywords: Architectural optimization, integer linear programming, microelectrofluidics, partition map, reconfigurable architecture, scheduling.

¹ This research was supported in part by DARPA under contract no. F30602-98-2-0140.

1 INTRODUCTION

Composite microsystems that incorporate microelectromechanical (MEMS) and microelectrofluidic systems (MEFS) are emerging as the next generation of system-on-a-chip (SOC) designs. These systems combine microstructures with solid-state electronics to integrate multiple energy domains, such as electrical, mechanical, and fluidic. The combination of microelectronics and microstructures is enabling a new class of integrated systems targeted at environmental sensing, actuation and control, biomedical analyses, agent detection, and precision fluid dispensing.

Microfluidics not only offers size reduction, e.g. in small medical implants and minimal-invasive surgery, but it also reduces power dissipation and increases system reliability. Microfluidics allows us to control small amounts of fluids for precision dispensing (micro-dosing) and reduce reagent consumption for on-line chemical analysis and real-time process monitoring. However, current-generation of MEFS are application-specific and they are incapable of performing a collection of differing analyses or procedures. We now need design methodologies and tools that allow microfluidic devices to be assembled into an SOC that can perform a variety of tasks supporting a diverse set of applications. This SOC can then be readily reconfigured and reused for chemical detection, analysis, diagnostics, and dispensing. While a number of CAD tools for MEMS are now commercially available, these tools are inadequate for current and next-generation MEFS.

Electrowetting-based actuation for microelectrofluidics (MEFS) has recently been proposed for optical switching [1], chemical analysis [2], and rotating yaw rate sensing [3]. Pollack et al recently demonstrated that by varying the electrical potential along a linear array of electrodes, electrowetting techniques can be used to move liquid droplets along this line of electrodes [2]. By carefully controlling the electrical potential applied to the electrodes, fluid droplets can be moved as fast as 3 cm/sec (Figure 2, and movies at <http://www.ee.duke.edu/research/MONARCH/movies.html>).

Electrowetting can also be used to move droplets in a two-dimensional electrode array. By controlling the voltage on the electrodes, fluid droplets can be moved freely to any location

on a two dimensional plane [2]. Fluid droplets can also be confined to a fixed location and isolated from other droplets moving around it.

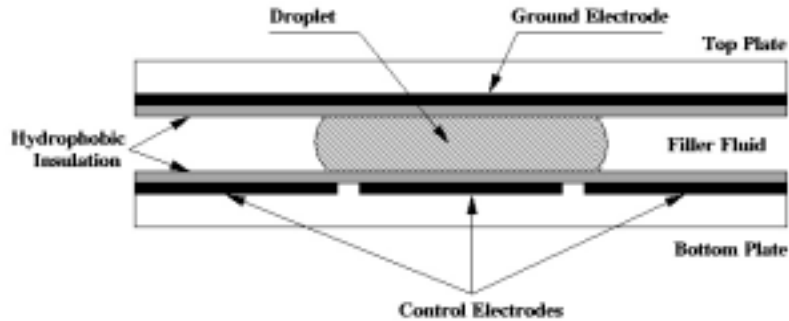


Figure 1: Actuation mechanism for droplet movement [3].

Using two-dimensional electrowetting arrays, many useful microfluidic operations can be performed, such as storing, mixing and droplet splitting. The store operation is performed by applying an insulating voltage around the droplet. This is analogous to a well. The insulating voltage prevents this droplet from mixing with other droplets around it. The mix operation is performed by routing two droplets to the same location, where they are merged into one droplet. Since the size of a droplet is kept small, effective mixing can be achieved by fluid diffusion after merging. Finally, the split operation is performed by creating opposite surface tension at the two ends of a fluid droplet and tearing it into two smaller droplets.

While two-dimensional electrowetting arrays are especially useful for biochemical analysis, system level design methodologies are required to harness this exciting new technology. In this paper, we leverage electronic design automations techniques to develop the first system-level design methodology for reconfigurable MEFS-based lab-on-a-chip.

Reconfigurable computing systems based on field-programmable gate-arrays (FPGAs) are now commonplace [4]. However, the “programmability” of FPGAs is limited by the well-defined roles of interconnect and logic blocks. Interconnect cannot be used for storing information, and logic blocks cannot be used for routing. In contrast, the MEFS architecture that we are developing offers significantly more programmability. The grid points between electrodes can be used for storage, functional operations, as well as for transporting fluid droplets.

Therefore, partitioning, resource allocation, and scheduling have emerged as major challenges for system-level MEFS design targeted at a set of biochemical applications.

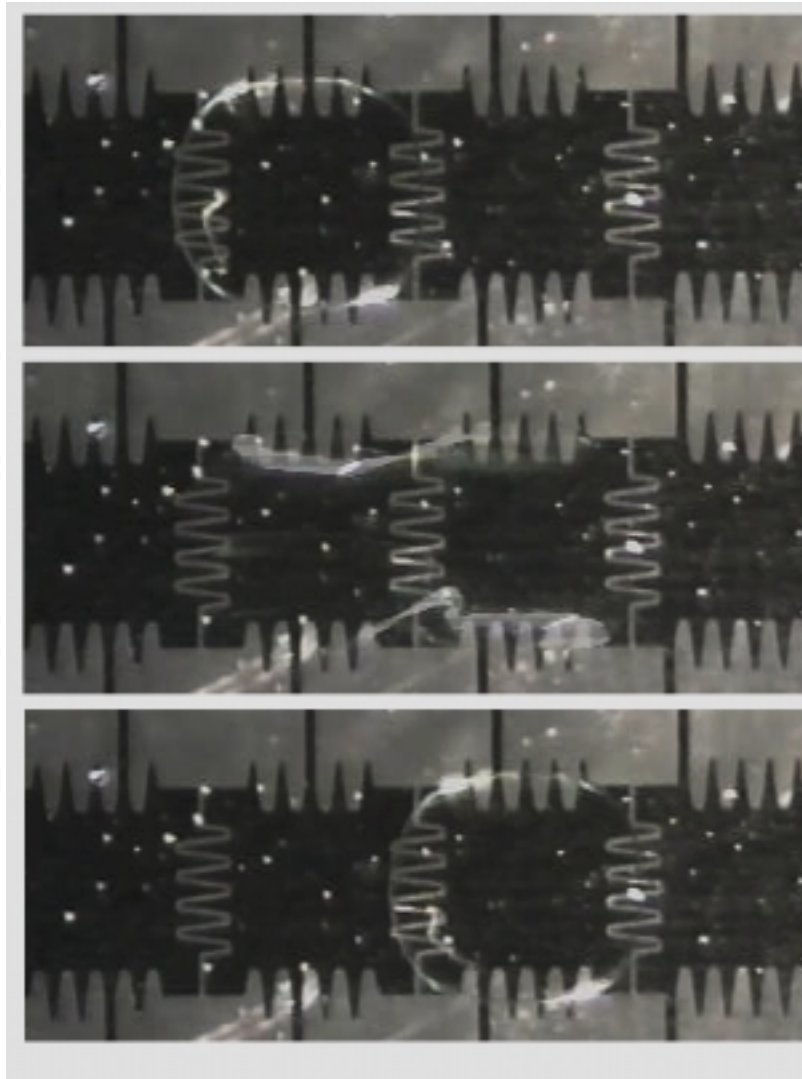


Figure 2: Droplet motion in an electrowetting array [2].

We have developed the syntax and semantics of microfluidic operations such as *MOVE*, *MIX*, and *SPLIT* that can be used to describe biochemical processes such as Polymerase Chain Reaction (PCR) [8]. The various fluid samples represent the operands. Such a *microfluidic program* must then be mapped to the two-dimensional array that represents the datapath of a *microfluidic computer*. (A separate electronic control unit drives the electrodes.) The execution

of microfluidic operations requires the availability of datapath resources (set of grid points) that can be appropriately configured. For example, the *MIX* operation requires that a set of grid points be properly configured to act as a mixer. The size of the two-dimensional array limits the number of concurrent operations of any type that can be carried out. Furthermore, functional dependencies between the operations in a microfluidic program also limit concurrency.

The organization of the paper is as follows. In Section 2, we describe the two-dimensional electrowetting array and introduce the concepts of virtual microfluidic components and partition maps. Section 3 presents the scheduling problem for biochemical analysis and describes an integer linear programming approach for scheduling under resource constraints in two-dimensional electrowetting arrays. Finally, in Section 4 investigate the PCR reaction as a case study. The PCR reaction is an important step in LOC biochemical processing. Processing time must be minimized for a number of critical LOC applications such as the detection and identification of biochemical agents and health monitoring during surgery. Efficient scheduling techniques not only reduce processing time, but they also offer better resource utilization in two-dimensional electrowetting arrays.

2 TWO-DIMENSIONAL ELECTROWETTING ARRAYS

A two-dimensional electrowetting array consists of a grid of electrodes on a two-dimensional plane (Figure 1). Fluid droplets are introduced to the device from the I/O ports on the boundary of the array. Droplets in the array have identical volumes. Hence, this type of device is also called a *unit-flow device*. It is desirable to maintain the unit-flow constraint since the rate of chemical and biomedical reactions grows exponentially with the growth of droplet volume [2].

Operations such as *STORE*, *MOVE*, *MIX*, and *SPLIT* are performed by controlling the electrical potential applied to the electrodes. It is easy to see that some of these operations violate the unit-flow assumption. For example, the fluid droplet size is likely to double as a result of a mixing operation. Therefore, we always perform a split operation after mixing to maintain the droplet volume.

In a continuous-flow MEFS system, mixing is performed using a micromixer. This is a specific device located at a fixed place in the microfluidic system. In unit-flow systems however, mixing operations can happen anywhere on the array, not necessarily at a specific location. If we

define a mixer as the location where fluids mix, then a unit-flow mixer can be located at any arbitrary cell in the electrode grid. This property is referred to as *reconfigurability*, and it is in many ways similar to the reconfigurability provided by FPGAs. However, as discussed in section 1, unit-flow devices allow a higher degree of reconfigurability than FPGAs. Storage cells, mixers and splitters can be created, removed, and relocated at runtime. This allows us to create extremely flexible and efficient biochemical analysis systems.

An abstract model of the unit flow system with a two-dimensional grid of electrodes is shown in Figure 1. A ground plane is positioned above the electrode array at a spacing that is less than the diameter of the droplets. I/O ports are placed at the boundary of the system.

2.1 Virtual devices and partition maps

In the unit-flow environment, the routes that droplets travel and the rendezvous points of fluid droplets are programmed into a micro-controller that controls the voltages of electrodes. The storage and interconnect on the datapath are viewed as virtual devices by the controller.

A virtual device is defined to have three regions. The first is the *functional region*, where a particular function is performed. The second type of region is called the *segregation region*, which wraps around the functional region. This insulates the functional region from its environment. The outer-most region of the device is the *inherited communication path*. This provides a one-cell wide communication path for fluid droplet movement. Figure 3 shows a unit-flow storage cell. One droplet of a fluid sample is stored in each functional cell.

A *partition map* shows the time-varying positions of all the virtual devices inside the defined area. It is generated by the designer, and pre-loaded into the microcontroller, which then controls the electrode voltages according to partition map.

A partition map is similar to a virtual device, in that it is also a virtual map, and it only exists in the microcontroller specification. It is also dynamic in nature since it may change with time. Reconfiguration occurs when a new partition map is loaded into the controller. Figure 4 shows a partition map containing two storage cells, one input cell, and one mixer. (The labels A, B, ..., J, K will be explained later.) The inherited communication paths of adjacent devices are combined to form a single channel in the electrode array. This channel is used for fluid droplet transfer, and is called a communication path. It forms the main network for fluid movement. Researchers have recently shown that it is possible to move the fluid droplets at a speed of 20 grids/second along

minutes and a *SPLIT* operation takes 2 minutes, then the time slot is set to 1 minute. In this case, the *MIX* operation will take 3 slots, and the *SPLIT* operation will take 2 slots. In this way, we digitize the continuous fluid operation and the controller starts or completes an operation at the end of each time slot.

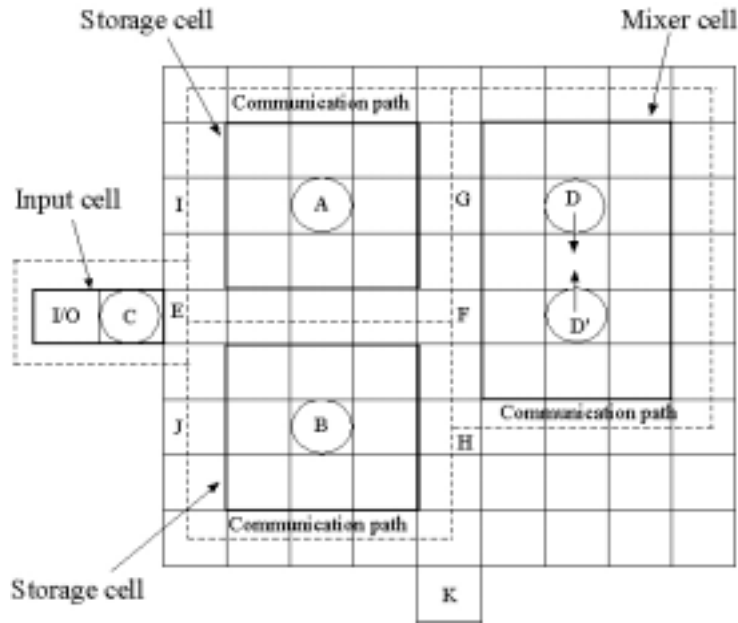


Figure 4: Partition map with two storage units, one input cell, and one mixer.

3 SCHEDULE OPTIMIZATION

The order of execution of microfluidic operations must be determined after carefully considering the dependencies between the operations and the availability of resources. While dependencies are imposed by the biochemical application, the resource constraints are imposed by the size of the two-dimensional electrowetting array and the availability of virtual devices. In this section, we use the dataflow graph model of high-level synthesis [6] to represent the scheduling problem and solve it using integer linear programming (ILP). The motivation for using ILP lies in the fact that it is a well-understood optimization method and we can leverage a number of public domain solvers [7].

First, each step of a biochemical process is represented using either a single microfluidic operation or a series of basic microfluidic operations. Each such instance of an operation forms a node in the dataflow graph. A directed edge from node u to node v indicates a dependency between the operations corresponding to u and v , i.e. the operation corresponding to u must be carried out before the operation corresponding to v . The goal of the scheduling problem is to determine the start times (time slots) of each operation so that the total completion time is minimized.

Let $x_{i,j}$ be a binary variable defined as follows:

$$x_{i,j} = \begin{cases} 1, & \text{if operation } i \text{ starts at time slot } j \\ 0, & \text{otherwise} \end{cases}$$

where $1 \leq i \leq N$, the number of operations (nodes in the dataflow graph), and $1 \leq j \leq M$, the maximum possible index for a time slot. Note that M can be trivially obtained by adding up the number of time slots required for all the operations. Note also that since each operation is scheduled exactly once, $\sum_{j=1}^M x_{i,j} = 1, 1 \leq i \leq N$.

The starting time S_i for operation i can now be expressed in terms of the set of variables $\{x_{i1}, x_{i2}, \dots, x_{im}\}$. Assuming that each time slot is of length 1 unit, we get $S_i = \sum_{j=1}^M jx_{ij}$.

Each operation i has an associated execution time d_i . If there exists a dependency edge between operation i and operation j , then $S_j \geq S_i + d_i$. Such dependencies generally arise from the fluid samples that are used in each step of the biochemical reaction. These fluid samples are similar to variables in traditional architectural synthesis.

Finally, we add resource constraints to the ILP model. Let a_k be an upper bound on the number of operations of type k . We now have the following set of constraints for each k :

$$\sum_{i \in T(k)} \sum_{j=l-d_i+1}^l x_{ij} \leq a_k, 1 \leq l \leq M$$

The objective of this optimization problem is to minimize the completion time of the last operation, i.e. **minimize** $\max_i \left\{ \sum_{j=1}^M jx_{ij} + d_i \right\}$. This can be linearized as: **minimize** C subject to $C \geq \sum_{j=1}^M (jx_{ij}) + d_i, 1 \leq i \leq N$.

The ILP model can be easily solved using public-domain solvers. In our work, we used the *lpsolve* package from Eindhoven University of Technology in Netherlands [7].

4 PCR EXAMPLE

In this section, we present a case study for operation scheduling using the PCR reaction. The PCR reaction includes three basic steps. The first is the input section. In this part, a number of fluid samples are input into the system. Next, these samples are combined using a pre-determined set of *MIX* operations. Note that these are implemented by interleaving *MOV*, *MIX*, and *SPLIT* operations. Finally, the sample mixture is sent off-chip for a series of heating steps.

The input samples for PCR include *Tris-HCl (pH 8.3)*, *KCl*, *gelatin*, *bovine serum albumin*, *beosynucleotide triphosphate*, a *primer*, *AmpliTaq DNA polymerase*, and λ *DNA*. The PCR procedure consists of the following series of steps:

1. Introduce *Tris-HCl (pH 8.3)* to storage.
2. Introduce *KCl*.
3. Wait until *KCl* mixes with *Tris-HCl*.
4. Introduce *gelatin*.
5. Wait until it mixes.
6. Introduce *bovine serum albumin*.
7. Wait until it mixes.
8. Introduce *beosynucleotide triphosphate*.
9. Wait until it mixes.
10. Introduce *primer* to the storage.
11. Wait until it is well mixed with the mixture.
12. Introduce *AmpliTaq DNA polymerase*.
13. Wait until it is well mixed with the mixture.
14. Introduce λ *DNA* to the storage.
15. Wait until it is well mixed with the mixture.

4.1 System configuration

The first example system we use is shown in Figure 4. The system can perform moving, mixing and splitting for the PCR reaction. It consists of 9-by-9 array of grid cells. A dedicated I/O port is located at the edge of the system. We assume that the mixing of two fluid droplets takes 2 minutes, while the input operation takes 0.4 minutes. Since the mix operation is always followed by a split operation, the latter is not explicitly considered here. Instead, we assume that the time for a split is included in the time for a mix operation. The speed of fluid movement is assumed to be 20 grid cells per minute.

The partition map for this example is given by Figure 4. In addition to the partition map, the droplet route plan and schedule of operations (to be determined next) must be loaded into the controller.

4.2 Optimal Scheduling

We now describe how an optimal schedule can be derived to minimize the processing time. First, we represent the PCR reaction as a series of basic steps. This corresponds to a specification outlined by a lab technician, and serves as a user program. The user program can either be a sequential enumeration of steps, or it can contain a limited amount of hand-extracted concurrency. We then generate the dataflow graph based on the functional dependencies between the operations (Figure 5). An optimized PCR reaction for the datapath of Figure 1 and the dataflow graph of Figure 5 is given below:

Time (minutes)	Operations	Representation
	<i>Definition section</i> Path path1, C-E-F-G-D Path path2, C-E-F-H-D' Path path3, C-E-I-A Path path4, C-E-J-B Path path5, D'-F-H-K Path path6, A-G-F-D' Path path7, B-H-F-D'	Definition
0	Load partition map <i>INPUT</i> Tris-HCl	I1
0.4	<i>MOVE</i> C, D, path1 <i>INPUT</i> KCl	I2
0.8	<i>MOVE</i> C, D', path2 <i>INPUT</i> gelatin <i>MIX</i> D and D'	I3 M1
1.2	move C, A, path3 <i>INPUT</i> bovine serum albumin	I4
1.6	<i>MOVE</i> C, B, path4	
2.8	<i>MOVE</i> D', K, path5 <i>MOVE</i> A, D', path6 <i>INPUT</i> beosynucleotide triphosphate	I5

	<i>MIX</i> D and D'	M2
3.2	<i>MOVE</i> C, A, path3	
4.8	<i>MOVE</i> move D', K, path5 <i>MOVE</i> A, D', path6 <i>INPUT</i> primer <i>MIX</i> D and D'	I6 M3
5.2	<i>MOVE</i> C, A, path3	
6.8	<i>MOVE</i> D', K, path5 <i>MOVE</i> A, D', path6 <i>INPUT</i> AmpliTaq DNA polymerase <i>MIX</i> D and D'	I7 M4
7.2	<i>MOVE</i> C, A, path3	
8.8	<i>MOVE</i> D', K, path5 <i>MOVE</i> A, D', path6 <i>INPUT</i> λ DNA <i>MIX</i> D and D'	I8 M5
9.2	<i>MOVE</i> Move C, A, path3	
10.8	<i>MOVE</i> D', K, path5 <i>MOVE</i> A, D', path6 <i>MIX</i> D and D'	M6
12.8	<i>MOVE</i> D', K, path5 <i>MOVE</i> B, D', path7 <i>MIX</i> D and D'	M7
14.8	<i>MOVE</i> D', K, path5	

Table 1: Optimized PCR reaction based on the datapath of Figure 1 and the dataflow graph of Figure 5.

The optimized PCR program of Table 1 was easy to derive since there is only one mixer in the system. The total processing time using this schedule is 14.8 minutes. We next show how the processing time can be decreased further and an optimal schedule derived using ILP.

Consider the partition map shown in Figure 6 with two mixers. This allows greater parallelism and demonstrates the advantage is using ILP to minimize the processing time. The following discussion presents the ILP model for this example in more detail.

The PCR program contains a total of 15 *INPUT* and *MIX* operations. From Table 1, we note that an upper bound on the processing time is 15 minutes. Each time slot is of length 0.4 minutes (the assumed time for an *INPUT* operation); hence an upper bound on the number of time slots is 37. There are 37 lots

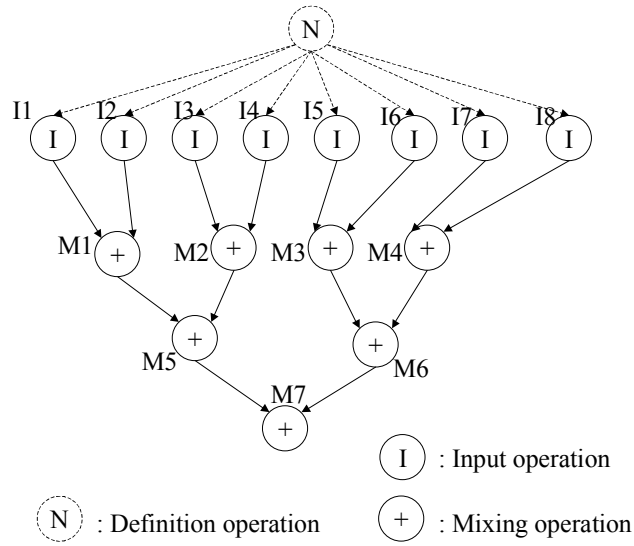


Figure 5: Dataflow graph with input and mix operations.

needed for the schedule. To build the ILP model for this partition map, we define a set of decision variables as discussed in Section 3. Thus our ILP model uses $x_{1,j} \dots x_{15,j}$ as the decision variables, where $j = 1, 2, \dots, 37$. The start time of each operation can be expressed as follows:

$$\begin{aligned}
 S_1 &= x_{1,2} + 2x_{1,3} + \dots + 29x_{1,37} \\
 S_2 &= x_{2,2} + 2x_{2,3} + \dots + 29x_{2,37} \\
 &\dots \\
 S_{15} &= x_{15,2} + 2x_{15,3} + \dots + 29x_{15,37}
 \end{aligned}$$

The dependency between instructions can be denoted using the following set of inequalities:

$$\begin{aligned}
 S_9 &> S_1, S_2 \\
 S_{10} &> S_3, S_4 \\
 S_{11} &> S_5, S_6 \\
 &\dots \\
 S_{15} &> S_{13}, S_{14}
 \end{aligned}$$

Finally, the resource constraints can be represented as:

$$\begin{aligned}
 x_{1,1} + x_{2,1} + \dots + x_{15,1} &< 2 \\
 x_{2,2} + x_{2,2} + \dots + x_{15,2} &< 2 \\
 &\dots \\
 x_{37,15} + x_{37,15} + \dots + x_{37,15} &< 2
 \end{aligned}$$

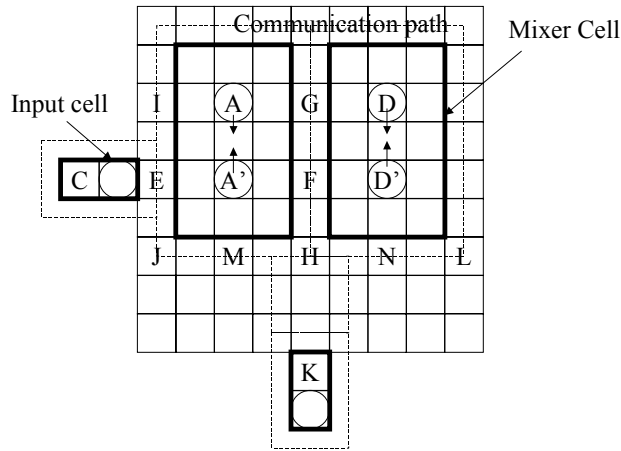


Figure 6: Partition map with two mixers for PCR reaction.

We solved this ILP model using *lpsolve*. It took 10 minutes of CPU time on a Sun Ultra Sparc with a 333 MHz processor and 128 MB of RAM. The optimum processing time is 9.6 minutes, 50% faster than the PCR program of Table 1. The optimized schedule is given below:

Time (minutes)	Operations	Representation
	Path path1, C-E-J-H-G-D Path path2, C-E-J-H-F-D' Path path3, C-E-I-A Path path4, C-E-A' Path path5, D'-F-H-K Path path6, A'-F-H-K Path path7, A-G-F-D'	Definition
0	Load partition map <i>INPUT</i> Tris-HCl	I1
0.4	<i>MOVE</i> C, D, path1 <i>INPUT</i> KCl	I2
0.8	<i>MOVE</i> C, D', path2 <i>INPUT</i> gelatin <i>MIX</i> D and D'	I3 M1
1.2	<i>MOVE</i> C, A, path3 <i>INPUT</i> bovine serum albumin	I4
1.6	<i>MOVE</i> C, A', path4 <i>INPUT</i> beosynucleotide triphosphate <i>MIX</i> A and A'	I5 M2
2.8	<i>MOVE</i> D', K, path5 <i>MOVE</i> A, D', path1 <i>MIX</i> D and D' <i>INPUT</i> primer	M3 I6
3.6	<i>MOVE</i> A', K, path6 <i>MOVE</i> C, A', path4	

	<i>MIX</i> A and A' <i>INPUT</i> AmpliTaq DNA polymerase	M4 I7
4.8	<i>MOVE</i> D', K, path5 <i>MOVE</i> C, D', path1 <i>MIX</i> D and D' <i>INPUT</i> λDNA	M5 I8
5.6	<i>MOVE</i> D', K, path5 <i>MOVE</i> C, D', path1 <i>MIX</i> D and D'	M6
6.8	<i>MOVE</i> A', K, path6	
7.6	<i>MOVE</i> D', K, path5 <i>MOVE</i> A, D', path7 <i>MIX</i> D and D'	M7
9.6	<i>MOVE</i> D', K, path5	

This can be represented using the annotated dataflow graph shown in Figure 7.

Consider next the partition map shown in Figure 3 with four mixers. This allows even greater parallelism and decreases processing time further. In the ILP model, we reformulated the resource constraints as follows:

$$x_{1,1} + x_{2,1} + \dots + x_{15,1} < 4$$

...

$$x_{37,15} + x_{37,15} + \dots + x_{37,15} < 4$$

The partition map for this implementation is shown in Figure 8.

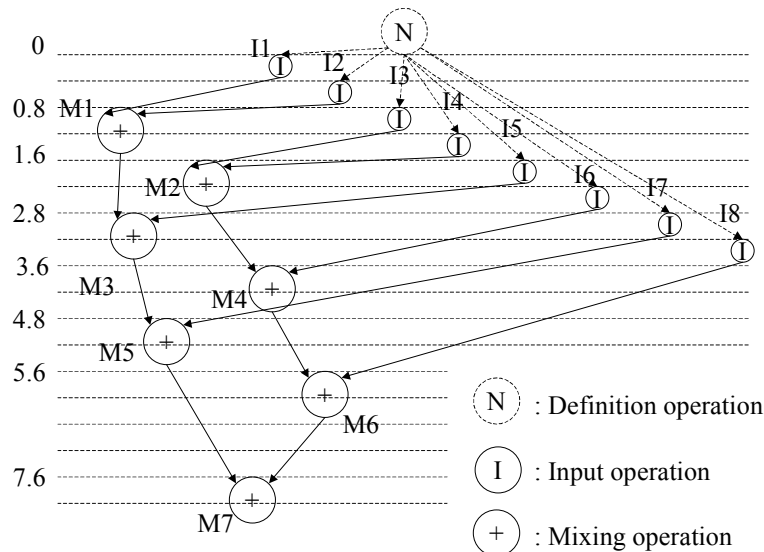


Figure 7: Dataflow graph showing an optimized schedule for 2-mixer partition map.

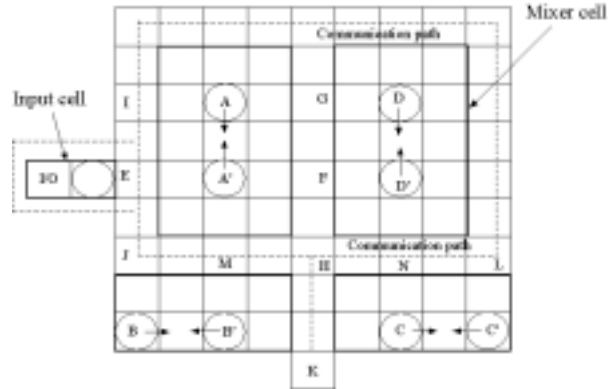


Figure 8: Partition map with four mixers for PCR reaction.

We solved this ILP and obtained the optimum processing time of 9.2 minutes, roughly 5.2% faster than 2-mixer version. Since the speed-up from two mixers to four mixers is insignificant, we conclude the maximum amount of parallelism in the PCR reaction has already been achieved.

5 CONCLUSIONS

We have presented a novel architectural design and optimization methodology for performing biochemical reactions using two-dimensional electrowetting arrays. We have defined a set of basic microfluidic operations and leveraged electronic design automation principles for system partitioning, resource allocation, and operation scheduling. While concurrency is desirable to minimize processing time, it is limited by the size of the two-dimensional array and functional dependencies between operations. We have used integer linear programming to minimize the processing time by automatically extracting parallelism from a biochemical assay. As a case study, we have applied our optimization method to the polymerase chain reaction. The proposed technique leverages known scheduling algorithms from electronic design automation for lab-on-a-chip (LOC) design, and it is expected to aid several LOC applications such as the rapid detection of biochemical agents, and reliable health monitoring during surgery.

REFERENCES

- [1] J. L. Jackel, S. Hackwood, J. J. Veslka, and G. Beni. "Electrowetting Switch for Multimode Optical Fibers", *Applied Optics*, vol. 22, no.11, pp. 1765-1770, 1999
- [2] M. Pollack, R. B. Fair and A. Shenderov, "Electrowetting-based actuation of liquid droplets for microfluidic applications", *Applied Physics Letters*, vol. 77, no. 11, pp. 1725-1726, July 2000.

- [3] R. Yates, C. Williams, C. Shearwood and P. Mellor, "A Micromachined Rotating Yaw Rate Sensor", *Proc. Micromachined Devices and Components II, SPIE Meeting*, pp. 161-168, 1996.
- [4] S. M. Trimberger, ed., *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, Norwell, MA, 1994.
- [5] J. R. Welty, C. E. Wicks and R. E. Wilson, "*Fundamentals of Momentum, Heat, and Mass Transfer*", John Wiley & Sons, Inc, New York 1983.
- [6] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, Inc., New York, NY, 1994.
- [7] M. Berkelaar, *lpsolve 3.0*, Eindhoven University of Technology, Eindhoven, The Netherlands, ftp://ftp.ics.ele.tue.nl/pub/lp_solve.
- [8] L. C. Waters et al., "Multiple Sample PCR Amplification and Electrophoretic Analysis on a Microchip", *Analytical Chemistry*, vol. 70, no. 24, December 1998.