Routing-Based Synthesis of Digital Microfluidic Biochips

Elena Maftei
em@imm.dtu.dk

Paul Pop
pop@imm.dtu.dk

Jan Madsen
jan@imm.dtu.dk

DTU Informatics
Technical University of Denmark
DK-2800 Kgs. Lyngby, Denmark

ABSTRACT

Microfluidic biochips are replacing the conventional biochemical analyzers, and are able to integrate on-chip all the basic functions for biochemical analysis. The "digital" microfluidic biochips are manipulating liquids not as a continuous flow, but as discrete droplets on a two-dimensional array of electrodes. Basic microfluidic operations, such as mixing and dilution, are performed on the array, by routing the corresponding droplets on a series of electrodes. So far, researchers have assumed that these operations are executed on rectangular virtual devices, formed by grouping several adjacent electrodes. One drawback is that all electrodes are considered occupied during the operation execution, although the droplet uses only one electrode at a time. Moreover, the operations can actually execute by routing the droplets on any sequence of electrodes on the array. Hence, in this paper, we eliminate the concept of virtual modules and allow the droplets to move on the chip on any route during operation execution. Thus, the synthesis problem is transformed into a routing problem. We propose an approach derived from a Greedy Randomized Adaptive Search Procedure (GRASP) and we show that by considering routing-based synthesis, significant improvements can be obtained in the application completion time.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Performance, Design

Keywords

Microfluidics, biochips, synthesis, routing

1. INTRODUCTION

Microfluidic biochips (also referred to as lab-on-a-chip) represent a promising alternative to conventional biochemical laboratories, and are able to integrate on-chip all the necessary functions for biochemical analysis using microfluidics, such as, transport, dispensing, mixing, and detection [6].

Biochips offer a number of advantages over conventional biochemical procedures. By handling small amount of fluids, they provide higher sensitivity while decreasing reagent consumption, hence reducing cost. Moreover, due to their miniaturization and automation, they can be used as point-of-care devices, in areas that lack the infrastructure needed by conventional laboratories [16].

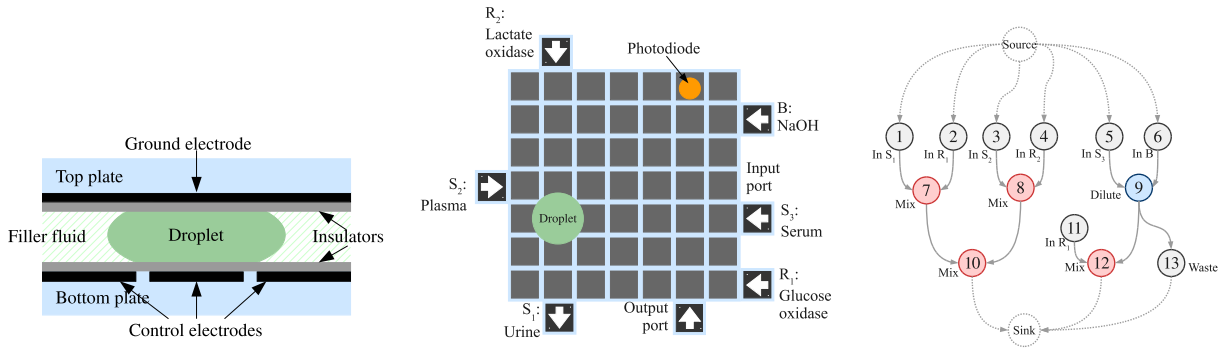
Due to these advantages, biochips are expected to revolutionize clinical diagnosis, especially immediate point-of-care diagnosis of diseases. Other emerging application areas include drug discovery, DNA analysis (e.g., polymerase chain reaction and nucleic acid sequence analysis), protein and enzyme analysis and immuno-assays.

There are two generations of microfludic biochips. The first generation is based on the manipulation of continuous liquid through fabricated micro-channels, using external pressure sources or integrated mechanical micro-pumps [16]. Although adequate for many simple biochemical applications, their integrated micro-structures make continuous-flow biochips unsuitable for more complex applications, requiring complicated fluid manipulations [2]. The second generation is based on the manipulation of discrete, individually controllable droplets on a two-dimensional array of identical cells. The actuation of droplets is performed without the need of microstructures, leading to increased scalability and flexibility compared with continuous-flow biochips [12]. This generation is also referred to as "digital microfluidics", due to the analogy between the droplets and the bits in a digital system. In this paper, we are interested in the second generation, droplet-based Digital Microfluidic Biochips (DMBs).

Researchers have so far considered that the execution of operations is constrained to a group of adjacent electrodes forming a rectangular "virtual device". In this context, DMBs have conceptual similarities to dynamically reconfigurable field-programmable gate arrays (DR-FPGAs). However, in the case of DMBs, devices are *virtual*, as operations can be performed on any sequence of electrodes on the microfluidic array. The abstraction of using virtual devices has the advantage that the same synthesis techniques used for DR-FPGAs can be adapted for the synthesis of DMBs. Hence, researchers have addressed the same problems: *allocation* of devices from a module library, *binding* of devices to operations, *scheduling*, *placement* and *routing*.

A unified high-level synthesis and module placement methodology has been proposed in [14], where the focus has been on deriving an implementation that can tolerate faulty cells in the biochip array. Their algorithm was modified in [17] to include droplet-routing-aware physical design decisions. Yuh et al. [18] have proposed a synthesis and placement algorithm which uses a tree-based topological representation and is able to improve on the results from

CASES'10, October 24–29, 2010, Scottsdale, Arizona, USA.



(a) Cell architecture (b) Biochip: array of cells (c) Application graph

Figure 1: Biochip architecture and application model

[14]. In [10] we have proposed an ILP-based architectural-level synthesis and placement approach for DMBs, which although has the advantage of producing the optimal solution, is only feasible for limited problem sizes. In [9] we have proposed a Tabu Search-based synthesis methodology, where we considered that virtual devices can move (change their placement) during their operation, and have shown that significant improvements can thus be obtained.

All of the previous work considers that operations are performed on virtual devices, of rectangular shape, which have a fixed placement on the microfluidic array. One drawback is that all electrodes are considered occupied during the operation execution, although the droplet uses only one electrode at a time. Moreover, the operations can actually execute by routing the droplets on any sequence of electrodes on the microfluidic array. Hence, in this paper, we eliminate the concept of virtual modules and allow the droplets to move on the chip on any route during operation execution. Thus, the synthesis problem is transformed into a routing problem (see Fig. 2a and Fig. 2b for a visual explanation of module-based synthesis, respectively routing-based synthesis).

Routing has been addressed so far as a post-synthesis step, following the placement of modules on the array. Several techniques have been proposed for finding the routes on which droplets move. A prioritized A* search algorithm is presented in [1], where at each time the optimal motion plan is performed for the droplet with the highest priority. In [15], a modified Lee algorithm is proposed for finding the routes on which droplets are transported, while minimizing the number of used cells. A variant of the Open Shortest Path First network protocol is presented in [8], while a network-flow based routing algorithm is proposed in [19]. The results in [19] are improved in [4], by performing bypassibility analysis while routing. Thus, at each time, the droplet less likely to block the movement of the other droplets is chosen to be scheduled.

All these methods consider that routing is performed between virtual devices whose position on the microfluidic array is fixed and determined during the placement step, thus the routes have predefined fixed start- and end-points. In addition, the assumption is that the operation is executed within the virtual device. In our routing-based synthesis approach we eliminate the concept of virtual devices and perform operations while routing, and thus there are no fixed start- and end-points for the routes. Also, to guarantee operation completion, we are not interested in minimizing the routes, but we have to construct routes of a given length. Therefore, the existing algorithms are not directly applicable in our context.

In this paper, we propose a routing-based synthesis approach that, starting from a biochemical application modeled as a sequencing graph and a given biochip array, determines a complete synthesis of the application on the biochip.

Given a route on which the droplets are routed to complete an operation, we first devise a method for determining the percentage of operation completion, considering the length and shape of the route. Then, we present a Greedy Randomized Adaptive Search Procedure (GRASP)-derived routing algorithm for establishing the routes taken by droplets during the execution of operations. We show that by using our proposed routing-based synthesis, significant improvements can be obtained in the application completion time, allowing us to use smaller area biochips and thus reduce costs.

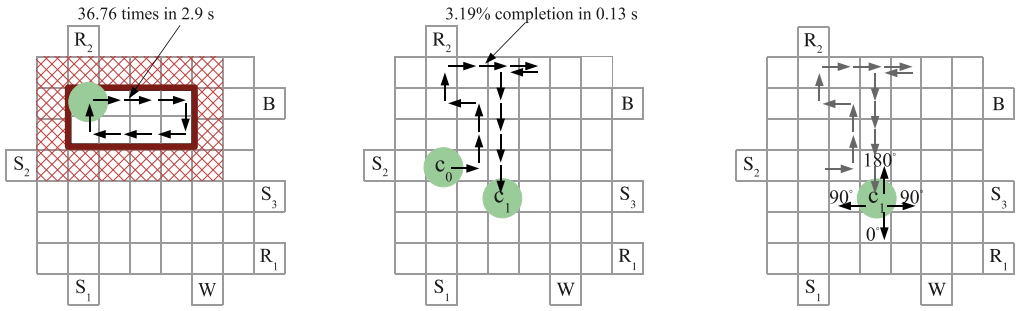
The paper is organized in six sections. Section 2.1 presents the architecture of a digital microfluidic biochip. We discuss the characterization of routing-based operation execution in Sections 2.2 and 2.3. The biochemical application model is presented in Section 2.4. We formulate the problem in Section 3 and illustrate the difference between module-based and routing-based synthesis. The proposed synthesis approach is presented in Section 4 and evaluated in Section 5. The last section presents our conclusions.

2. SYSTEM MODEL

2.1 Biochip Architecture

In a digital microfluidic biochip the manipulation of liquids is performed using discrete droplets. There are several mechanisms for droplet manipulation [6]. Our work considers electrowetting-on-dielectric (EWD) [12], but can be extended to handle other techniques as well. EWD is the most promising technique, and can provide high droplet speeds of up to 20 cm/s [12]. A biochip is composed of several cells, see Fig. 1b. The schematic of a cell is presented in Fig. 1a. The droplet is sandwiched between two glass plates (the top plate and the bottom plate), and moves within a filler fluid. The top plate contains a single ground electrode, while the bottom plate has several control electrodes. The electrodes are insulated from the droplet through an insulation material. With EWD, the movement of droplets is controlled by applying voltages to the required electrodes. For example, turning off the middle control electrode and turning on the right control electrode in Fig. 1a will force the droplet to move to the right. For the details on EWD, the reader is directed to [12].

Several cells are put together to form a two-dimensional array (an example architecture is presented in Fig. 1b). Using EWD manipulation, droplets can be moved to any location without the need for pumps and valves, which are required in a continuous-



(a) Module-based operation (b) Routing-based operation (c) Droplet movement

Figure 2: Execution of a mixing operation

flow biochip. Besides the basic cell discussed previously, a chip typically contains input and output ports and detectors. The detection can be done by using, for example, a light-emitting diode (LED) beneath the bottom plate and a photodiode on the top plate.

2.2 Module- vs. Routing-Based Operations

Using this architecture, and changing correspondingly the control voltages, all of the required operations, such as transport, splitting, dispensing, mixing, and detection, can be performed. For example, mixing is done by bringing two droplets to the same location and merging them, followed by the transport of the resulted droplet over a series of electrodes. By moving the droplet, external energy is introduced, creating complex flow patterns (due to the formation of multilaminates), thus leading to a faster mixing [11]. Mixing through diffusion, where the resulted droplet remains on the same electrode, is very slow. The operation can be executed anywhere on the microfluidic array and is not confined to a certain area, thus we say that mixing is a "reconfigurable" operation. Another reconfigurable operation is dilution, which consists of a sequence of mixing and splitting steps. A biochemical application may also contain "non-reconfigurable" operations, that are executed on real devices, such as reservoirs or optical detectors.

So far, it has been considered that reconfigurable operations are performed inside virtual modules, created by grouping adjacent cells. Such a module is shown in Fig. 2a, where the droplet is routed circularly on a series of electrodes until the mixing operation is completed. The movement of the droplet inside the module is described by the mixing pattern, represented by the arrows inside the virtual module.

Table 1 presents the results of the experiments performed in [11], where several mixing times were obtained for various areas, creating a module library. One problem addressed by the experiments is flow reversibility, when complex patterns inside the droplet are unfold into simpler ones when the direction in which the droplet is transported is changed by 180°. This is the case of linear mixers (e.g., Fig. 3b), where the motion of the droplet is bidirectional. One solution to avoid flow reversibility is to transport the droplet in a circular motion, as in the 2×2 virtual module shown in Fig. 3d. However, it has been shown that since the droplet is rotating around the pivot point in the center of the created module, part of the droplet remains unmixed and thus the operation takes longer (9.95 s) to complete. In the 2×3 module shown in Fig. 3c two additional electrodes are introduced in order to eliminate the static pivot point present in the 2×2 module, thus reducing the mixing time to 6.1 s. The mixing time is further improved for the 2×4 mixer in Fig. 3a, leading to a 2.9 s completion time. The experiments show that faster mixing is obtained by moving the droplet linearly for as long as possible, reducing thus the flow reversibility.

Table 1: Module library

Operation	Area (cells)	Time (s)
Mixing/Dilution	2×4	2.9
Mixing/Dilution	1×4	4.6
Mixing/Dilution	2×3	6.1
Mixing/Dilution	2×2	9.95
Dispensing	–	2
Detection	1×1	30

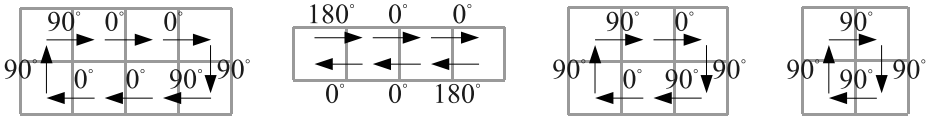
During module-based operation execution, all cells inside the module are considered occupied, although the droplet uses only one cell at a time. Thus, the remaining cells cannot be used for other operations, which is inefficient since it reduces the potential for parallelism. In addition, in order to prevent the accidential merging of a droplet with another droplet in its vicinity, a minimum distance must be kept between operations executing on the microfluidic array. For example, in Fig. 2a these fluidic constraints are enforced by surrounding the module by a 1-cell segregation area (the hashed area), containing cells that can not be used by other operations until mixing finishes. For further details on the fluidic constraints the reader is directed to [19].

An alternative to modules, proposed in this paper, is routing-based operation execution. As mixing is performed by routing, an operation can be executed anywhere on the array, unconstrained by a rectangular shape representing a virtual module. This characteristic of the mixing operation is shown in Fig. 2b, where the droplet is routed freely on a sequence of electrodes, according to the shown route.

2.3 Characterizing Routing-Based Operations

Table 1 gives the operation completion times for modules. For routing-based operation execution, the completion time depends on the actual route. In this section, we propose a safe approximation for the percentage of mixing performed while routing the droplet on a given route. As there is no mathematical model to characterize how the percentage of mixing varies depending on the movement of the droplet, our method provides estimates by decomposing the devices from Table 1.

Let us consider that while mixing a droplet, it reaches the cell c_1 at time t in Fig. 2b. We have five possibilities for the next time moment, $t+1$, as shown in Fig. 2c: routing the droplet to the left, to the right, up, down or keeping the droplet on c_1. Let us denote with p^0 the percentage of mixing obtained while routing the droplet on an electrode in a forward movement (relative to the previous move),



(a) 2 × 4 module (b) 1 × 4 module (c) 2 × 3 module (d) 2 × 2 module

Figure 3: Characterization of droplet mixing

with p^{90} the percentage obtained from a perpendicular movement of the droplet and with p^{180} the percentage of mixing obtained from a backward movement, see Fig. 2c.

Considering Table 1, we can estimate the percentage of mixing over one cell, coresponding to each type of movement (forward, backward, perpendicular). In order to approximate p^0, p^{90} and p^{180} we decompose the mixing patterns from the module library in Table 1 in a sequence of forward, backward and perpendicular motions, as shown in Fig. 3. For example, the 2 × 2 mixer in Fig. 3d can be decomposed in perpendicular movements, because after each move the droplet changes its routing direction by 90°. As shown in Table 1, the operation takes 9.95 s to execute inside the 2 × 2 module, thus we can safely approximate[1] the percentage of mixing p^{90} to 0.1%.

For the 2 × 3 module shown in Fig. 3c, the mixing pattern is composed of forward and perpendicular movements. By considering the mixing time shown in Table 1 and p^{90} = 0.1%, we obtain the percentage of mixing resulted from one forward movement p^0 = 0.29%. Note that by decomposing the 2 × 4 module shown in Fig. 3a, we obtain a different value for p^0: 0.58%. This is because the forward mixing percentage is not constant, but it depends on the number of electrodes used. Therefore we consider that there are two values that estimate the percentage of forward movement: p_1^0, when the forward movement is continued only for one cell as in Fig. 3c, and p_2^0, when the forward movement of the droplet is of at least two cells. This is a safe (pessimistic) approximation, since the value of p^0 will further increase if the droplet continues to move forward.

Considering the percentage of forward movement p_2^0 in the decomposition of the 1 × 4 module in Fig. 3b, we obtain the (pessimistic) percentage of mixing performed during a backward motion: p^{180} = −0.5%. The negative mixing is explained by the unfolding of patterns inside the droplet, i.e., the two droplets tend to separate when moved backward.

Using these percentages, we can determine the operation completion time for any given route. For example, in Fig. 2b we have 3.19% of the mixing completed in 0.13 s. We assume that before routing-based synthesis is performed, the set of percentages $\mu = \{p_1^0, p_2^0, p^{90}, p^{180}\}$ is determined through experiments such as the ones in [11] which have produced Table 1. The method presented in this subsection can be applied to any such experimental data.

2.4 Biochip Application Model

We model a biochemical application using an abstract model consisting of a sequencing graph [3]. The graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is directed, acyclic and polar (i.e., there is a *source node*, which is a node that has no predecessors and a *sink node* that has no successors). Each node $O_i \in \mathcal{V}$ represents one operation. For non-reconfigurable devices (e.g., dispensing, detection), the binding of operations to modules is captured by the function $\mathcal{B} : \mathcal{V} \rightarrow \mathcal{A}$, where $\mathcal{A}$ is the list of allocated modules from the given library $\mathcal{L}$. Each reconfigurable operation O_i (e.g., mixing, dilution) is allocated and bound to route $R_i \in \mathcal{R}$ on the array C.

An edge $e_{i,j} \in \mathcal{E}$ from O_i to O_j indicates that the output of operation O_i is the input of O_j. An operation can be activated after all its inputs have arrived and it issues its outputs when it terminates. We assume that, for each operation O_i, we know the execution time C_i on the non-reconfigurable module $M_k = \mathcal{B}(O_i)$ or route $R(O_i)$, where it is assigned for execution. In Fig. 1c we have an example of an application graph with thirteen operations, O_1 to O_{13}. The application consists of four mixing operations (O_7, O_8, O_{10} and O_{12}), one diluting operation (O_9), six input operations (O_1, O_2, O_3, O_4, O_5, O_6 and O_{11}) and one output operation (O_{13}).

O_9 is a dilution operation during which two unit droplets of different concentrations are mixed, resulting in a droplet of intermediate concentration having twice the unit volume. The mixing is followed by a split operation, during which two droplets of unit volume and intermediate concentration are obtained. Considering Fig. 1a, a droplet is split by turning on the left and right electrodes and turning off the middle electrode [13]. The result of the dilution operation O_9 is two droplets of intermediate concentration, one used as an input for the mixing operation O_{12} (denoted by $e_{9,12}$), and the other one, corresponding to O_{13}, discarded to the output reservoir. We assume that the biochemical application has been correctly designed, such that all the operations will have the required input droplet volumes.

3. PROBLEM FORMULATION

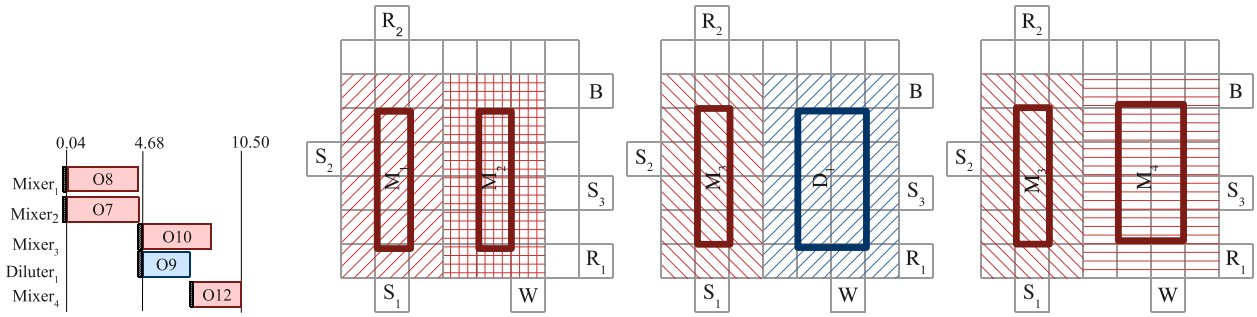
The problem we are addressing in this paper can be formulated as follows. Given (1) a biochemical application modeled as a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; (2) a biochip consisting of a two-dimensional $m \times n$ array C of cells ; and (3) a library $\mathcal{L}$ characterizing the completion time of the operations, we are interested to synthesize that implementation Ψ, which minimizes the completion time of the application $\delta_{\mathcal{G}}$. Let us first illustrate the module-based synthesis, followed by our proposed routing-based synthesis approach, presented in Section 3.2.

3.1 Module-Based Synthesis

The module-based synthesis problem consists in determining an implementation $\Psi = < \mathcal{A}, \mathcal{B}, \mathcal{S}, \mathcal{P}, \mathcal{R} >$, which means deciding on: (1) the allocation $\mathcal{A}$, which determines what modules from the library $\mathcal{L}$ should be used; (2) the binding $\mathcal{B}$ of each operation $O_i \in \mathcal{V}$ to a module $M_k \in \mathcal{A}$; (3) the schedule $\mathcal{S}$ of the operations, which contains the start time t_i^{start} of each operation O_i on its corresponding module; (4) the placement $\mathcal{P}$ containing the locations at which operations will be executed on the $m \times n$ array; and (5) the routes $\mathcal{R}$ taken by the droplets between modules and between modules and input/output ports.

Let us consider the graph shown in Fig. 1c. We would like to implement the operations on the 7 × 7 biochip from Fig. 1b. We consider the current moment of time as being $t = 0$. For simplicity, in this example, we consider that the input operations are already

[1]In this paper we consider the data from [12], where the time required to route the droplet one cell is 0.01 s.



(a) Schedule (b) Placement for $t \in [0.04, 4.64]$ (c) $t \in (4.68, 7.58]$ (d) $t \in (7.60, 10.50]$

Figure 4: Module-based synthesis example

assigned to the corresponding input ports. Thus, O_1 is assigned to the input port S_1, O_2 to R_1, O_3 to S_2, O_4 to R_2, O_5 to S_3, O_6 to B and O_{11} to R_1. For the other operations in Fig. 1c, the mixing operations (O_7, O_8, O_{10} and O_{12}) and the dilution operation (O_9) the module-based synthesis will have to allocate the appropriate virtual modules, bind operations to them and perform the placement and scheduling.

Let us assume that the available module library is the one captured by Table 1. We have to select modules from the library while trying to minimize the application completion time and place them on the 7×7 chip. A solution to the problem is presented in Fig. 4, where the following modules are used: three 1×4 mixers ($Mixer_1$, $Mixer_2$, $Mixer_3$), one 2×4 mixer ($Mixer_4$) and one 2×4 diluter ($Diluter_1$).

Considering this allocation, Fig. 4a presents the binding of operation to modules and the optimal schedule. The schedule is depicted as a Gantt chart, where, for each module, we represent the operations as rectangles with their length corresponding to the duration of that operation on the module. The routing times needed for merging the inputs of the operations are represented as hashed rectangles in the schedule. For example, operation O_{12} is bound to module $Mixer_4$, starts after the dilution operation O_9 is completed ($t_9^{finish} = 7.58$) and after its inputs, $e_{9,12}$ and O_{11}, are merged on the microfluidic array, thus $t_{12}^{start} = 7.60$. The operation takes 2.9 s, finishing at time $t_{12}^{finish} = 10.50$ s.

The placement for the solution is as indicated in Fig. 4b–d. Note that only two virtual devices can be placed on the biochip due to space constraints, thus only two operations can execute in parallel. In our case O_7, O_8 and O_9 could potentially be executed in parallel. If we decide to select smaller areas to increase parallelism, such as a 2×2, the execution time is much larger, e.g., 9.95 s for a 2×2, which eliminates the potential gain obtained through parallelism.

3.2 Routing-Based Synthesis

Let us consider the same synthesis problem for DMBs in the case when we remove the concept of "virtual device" (also called "module") and allow operations to execute by routing the droplets on any sequence of electrodes on the array. Similar to the problem formulation for module-based synthesis, we want to synthesize an implementation $\Psi = < \mathcal{A}, \mathcal{B}, \mathcal{S}, \mathcal{P}, \mathcal{R} >$, deciding the allocation, binding, scheduling, placement and routing. However, there are differences when performing routing-based synthesis. The allocation, binding and placement need to be performed only for non-reconfigurable operations, such as input and detection operations. For reconfigurable operations, such as mixing and dilution, the synthesis is determined by the routes $\mathcal{R}$. For each reconfigurable operation O_i we have to determine a time-ordered list containing electrodes on which O_i is executed (i.e., a route). Thus, for reconfigurable operations, the synthesis problem is transformed into a routing problem.

Let us consider the same example presented in Section 3.1. Fig. 5 shows the synthesis of the application on the 7×7 array. The allocation and binding of physical modules to *non-reconfigurable* operations is the same as the one presented in 3.1. We consider the characterization of droplet mixing as discussed in Section 2.3. We have to find the routes $\mathcal{R}$ for all the reconfigurable operations such that the application completion time $\delta_{\mathcal{G}}$ is minimized.

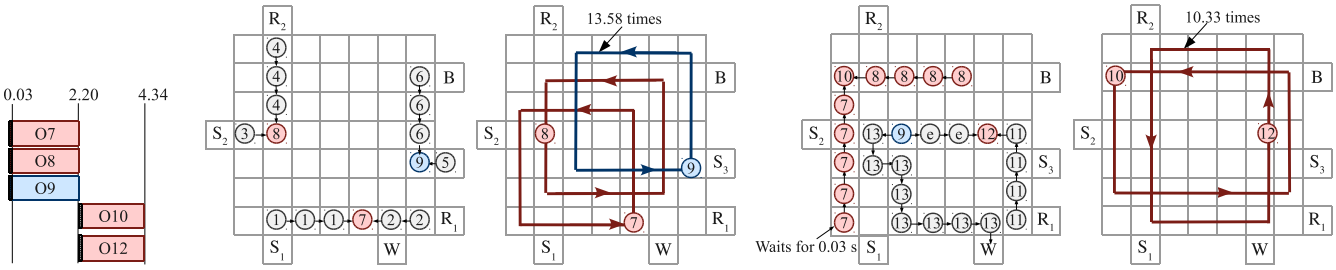
Fig. 5 shows a complete solution for synthetising the application $\mathcal{G}$ in Fig. 1c to the 7×7 chip. Before the reconfigurable operations O_7, O_8 and O_9 can start, we route their inputs to the locations depicted in Fig. 5b. In order to simplify the visual representation of the solution, we assume a repetitive route for the operations: the droplets corresponding to O_7, O_8 and O_9 in Fig. 5c are repeatedly routed on the shown paths 13.58 times, until the mixing is completed.

After completion, the droplets resulted from the mixing operations O_7 and O_8 are routed to a common location on the chip, where they merge, forming the droplet corresponding to operation O_{10} (Fig. 5d). The dilution operation O_9 continues by splitting the mixed droplet into two droplets of intermediate concentration and equal volume, corresponding to $e_{9,12}$ and the output operation O_{13}.

Because of simplicity reasons, in this example, the paths on which the droplets are routed while operations are executed are of rectangular shape. However, in routing-based synthesis any sequence of electrodes can be used as a path, as shown in Fig. 2b.

The schedule of the operations is presented in Fig. 5a, where we notice that the completion time of the application is significantly reduced compared to the module-based schedule presented in Fig. 4a, 4.34 s compared to 10.50 s.

There are several reasons for this reduction. Compared to the solution in Fig. 4, operation O_9 can be executed in parallel with O_7 and O_8 in Fig. 5c. Routing-based synthesis leads to an increase in paralellism due to a more efficient use of the microfluidic array. In module-based synthesis the entire module area, including the segregation borders, is considered occupied by the operation. In routing-based operation execution we know the actual position of the droplets, therefore all the other cells can be used, as long as the droplets are not too close to each other (i.e., the microfluidic constraints from [19] are enforced). For example, in Fig. 5d the droplet corresponding to O_7 must be kept on the initial position shown from time 2.20 s until time 2.23 s, in order to prevent the accidential merging with the droplet discarded to the output reservoir (corresponding to the operation O_{13}).



(a) Schedule (b) Placement at $t = 0.03$ (c) $t \in (0.03, 2.20]$ (d) $t = 2.28$ (e) $t \in (2.28, 4.34]$

Figure 5: Routing-based synthesis example

Another reason for the reduction of $\delta_{\mathcal{G}}$ is the increase in the number of electrodes used for forward movement. As discussed in Section 2.3, forward movement reduces flow reversibility inside the droplet, leading to a faster completion of the reconfigurable operations, such as mixing and dilution.

4. ROUTING-BASED SYNTHESIS

The problem presented in the previous section is NP-complete [4]. Our strategy is derived from GRASP [7] and decides the routes $\mathcal{R}$ taken by droplets during the execution of reconfigurable operations. The allocation, binding and scheduling for non-reconfigurable operations are decided using a greedy approach when these operations are needed by the synthesis of reconfigurable operations.

The proposed algorithm is presented in Fig. 6 and takes as input the application graph $\mathcal{G}$, the biochip array C and the percentages of mixing during droplet movement $\mu = \{p_1^0, p_2^0, p^{90}, p^{180}\}$, and produces an implementation $\Psi = <\mathcal{A}, \mathcal{B}, \mathcal{S}, \mathcal{P}, \mathcal{R}>$, which minimizes the schedule length $\delta_{\mathcal{G}}$.

Let us first discuss the synthesis of routes $\mathcal{R}$ for the reconfigurable operations. At each time t, a set of droplets corresponding to currently executing reconfigurable operations are present on the microfluidic array. A droplet can be in one of the two states: (1) *merge* — when it needs to come into contact with another droplet; and (2) *mix* — when it performs a mixing or dilution operation. For example, the droplets corresponding to operations O_3 and O_4 in Fig. 5b are in the *merge* state, as they need to be routed to a common location on the array in order to form the droplet corresponding to the operation O_8. Once it is formed, the O_8 droplet is routed on a sequence of electrodes until the mixing operation is completed. Thus, we say that in Fig. 5c the droplet corresponding to operation O_8 is in the *mix* state.

We use two lists, L_{merge} and L_{mix}, to capture the operations that are performed on the microfluidic array at time t and are in the *merge* and *mix* states, respectively. L_{merge} is initialized by considering the operations in the graph that are ready to be scheduled (line 4). The list L_{mix} is initially empty (line 5).

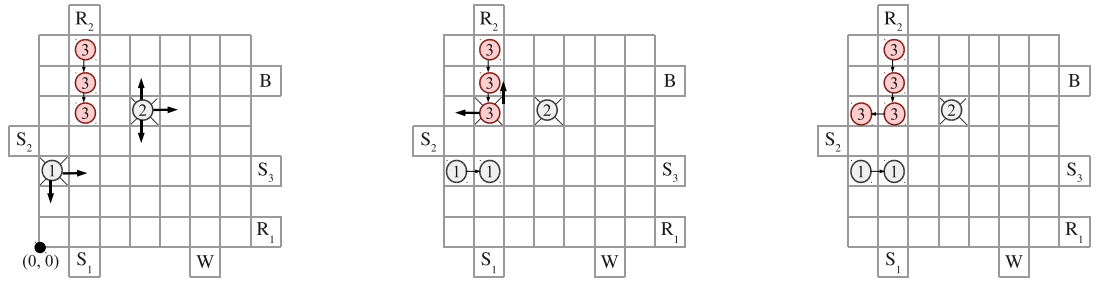
The main part of the algorithm is the while loop, lines 6–32, which terminates when all operations have finished. In each iteration, we increment the current time $t_{current}$ (line 31) and perform the following three steps: (1) We decide the new positions of the droplets present on the chip at $t_{current}$, i.e., $O_i \in L_{merge} \cup L_{mix}$ (lines 7–10); (2) In the second step, we introduce droplets on the array in the *mix* state, in case their predecessor droplets have merged on the chip (lines 11–19); (3) Finally, when the reconfigurable operations have finished executing (the droplets are mixed or diluted), we remember the finishing time (line 22) and put the resulting droplets in the *merge* state (line 29).

RoutingBasedSynthesis($\mathcal{G}$, C, μ)

```
1  t_current = 0
2  t_Oi^start = 0, ∀O_i ∈ G
3  t_Oi^finish = 0, ∀O_i ∈ G
4  L_merge = ConstructMergeList(G)
5  L_mix = ∅
6  while ∃O_i ∈ G ∧ t_Oi^finish = 0 do
7    // Step 1: move droplets present on the array
8    for all O_i ∈ L_merge ∪ L_mix do
9      PerformMove(O_i, C, R)
10   end for
11   // Step 2: if droplet finished merging
12   for all O_i ∈ L_merge ∧ O_i is merged do
13     // update L_merge
14     Remove(O_i, L_merge)
15     // schedule successors
16     ScheduleSuccessors(O_i)
17     // update L_mix
18     Add(O_i, L_mix)
19   end for
20   // Step 3: if droplet finished mixing
21   for all O_i ∈ L_mix ∧ O_i is mixed do
22     t_Oi^finish = t_current
23     // update L_mix
24     Remove(O_i, L_mix)
25     if O_i is a dilution operation then
26       ScheduleSuccessors(O_i)
27     end if
28     // update L_merge
29     Add(O_i, L_merge)
30   end for
31   t_current = t_current + 1
32 end while
33 return Ψ
```

Figure 6: Routing-based synthesis for DMBs

Let us present each step in more detail. In step 1, for each droplet present on the microfluidic array, we have to decide the next position (line 9). There is a large number of position combinations that has to be considered. We take the decision individually for each droplet, using the PerformMove function which takes as input the reconfigurable operation O_i, the biochip array C and the current routes $\mathcal{R}$. We use a randomized greedy approach similar to GRASP: for each droplet we construct a Restricted Candidate List (RCL), containing the three best feasible moves to be performed. Then, a move from the RCL is randomly selected and the



(a) Choosing moves for L_{merge} (b) Choosing a move for L_{mix} (c) Placement at $t_{current} + 1$

Figure 7: Performing droplet moves

droplet is transported in the corresponding direction. We use probabilities to favour the candidates from RCL which have a greater cost function. Thus, there is a probablity of 50% of choosing the best candidate from the RCL, 33.3% of choosing the second best candidate and 16.6% for the third best feasible move. Two cost functions are considered for determining the quality of the moves, depending on the state of the droplets. For a droplet in the *mix* state, the quality of a move is given by the percentage of mixing performed while transporting the droplet in the given direction calculated based on the set μ, see Section 2.3. For a droplet in the *merge* state, the quality of a move is determined by the distance between the two droplets that need to be merged, measured by the Manhattan distance.

Let us use Fig. 7a to illustrate how we determine the directions in which the droplets are moved. We consider that at time $t_{current}$ there are three operations executing on the array: the operations O_1 and O_2, that need to be merged, and the mixing operation O_3, thus $L_{merge} = \{O_1, O_2\}$ and $L_{mix} = \{O_3\}$. As discussed in Section 2.3, for an operation executing by routing, the amount of mixing performed during one move depends on the previous path on which the droplet was transported. The previous two moves for mixing operation O_3 are as indicated in Fig. 7a, by the position of the O_3 droplet, and the corresponding connecting arrows. For each of the droplets on the array we have a number of feasible moves that can be performed at the current time step. In Fig. 7a the feasible moves are depicted with thick arrows. The set of feasible moves includes also the decision of keeping the droplet on the same position, illustrated with an "X" under the droplet. When considering the feasible moves set we enforce the microfluidic constraints, which prevent the droplets from getting too close to each other and accidentialy merge. For example the move of droplet O_1 up is not permitted, since doing so would cause it to merge with droplet O_3.

The operations in L_{merge} are considered first. For the droplet corresponding to operation O_1 we have three possible moves: to the right, down or mantaining the droplet at the current location. We evaluate each of the possible directions, by computing the Manhattan distance between the new feasible position of O_1 and the position of the droplet that O_1 has to merge with, O_2. The current positions of the O_1 and O_2 droplets are (0,2) and (3,4) respectively, thus the initial Manhattan distance is 5, as shown in Fig. 7a. By moving O_1 to the right the new location of the droplet is (1,2), therefore the Manhattan distance between O_1 and O_2 is reduced to 4. Similarly, the Manhattan distance obtained by moving the droplet O_1 down and maintaining it at the current location are 6 and 5, respectively. Thus, moving O_1 to the right is the best decision, as it brings the droplets O_1 and O_2 closer to each other. The RCL is constructed by considering only the three best moves, thus $RCL_{O_1} = \{right, mantain, down\}$. A move is randomly chosen from the RCL and the placement of the droplet on the chip is updated. Let us consider for example that the droplet is moved to the right. Similarly we construct $RCL_{O_2} = \{down, mantain, right\}$ and randomly choose to mantain the droplet corresponding to O_2 at the current location. Fig. 7b shows the updated placement on the microfluidic array after the two moves are performed.

Next, the mixing operation O_3 is considered. The feasible directions in which the droplet can be routed are to the left, up or maintaining the droplet on the current position. Moving the droplet in a forward direction is not possible, as this could lead to an accidential merge with the droplet corresponding to O_1 (see Fig. 7b). As moving the droplet to the left would result in a perpendicular move compared to the previous one, the percentage of obtained mixing according to Section 2.3 is $p^{90} = 0.10\%$, while moving it backwards (up) would result in a negative mixing, $p^{180} = -0.5\%$. We consider mixing by pure diffusion negligible, thus no mixing is performed while the droplet remains at the same location. Therefore, $RCL_{O_3} = \{left, mantain, up\}$. We assume the droplet is randomly moved to the left, resulting in the placement at $t_{current} + 1$ shown in Fig. 7c.

In Step 2, for all the droplets in L_{merge} that have been brought to a common location at time $t_{current}$, their successors are activated and inserted into the corresponding lists. Their t_{start} is set to $t_{current}$(line 16) and their positions are at the same location where the droplets have met. For example, when O_1 and O_2 in Fig. 5b are merged at time $t = 0.03$, the mixing operation O_7 is placed on the array and starts executing, thus $t^{start}_{O_7} = 0.03$.

In Step 3, all the mixing operations completed at time $t_{current}$ and having sucessors are promoted to the *merge* state (lines 20–30). For example, at time $t = 2.20$, the state of the operation O_7 in Fig. 5d is changed from *mix* to *merge*, as O_7 needs to be merged with O_8 in order to form the droplet corresponding to the operation O_{10}. If the completed operation is of type dilution, then the droplet is split into two droplets of equal volumes, see the dilution operation O_9 in Fig. 5d. The droplets resulting from the split operation are scheduled (line 26) and their locations on the array are determined by their predecessor's final position.

Regarding non-reconfigurable operations, such as dispensing from input reservoirs and detection using optical devices, we consider that their allocation $\mathcal{A}$ and placement $\mathcal{P}$ are fixed and given as part of the biochip architecture model, see Fig. 1b for an example. However, we decide the binding $\mathcal{B} \in \Psi$ and scheduling $\mathcal{S} \in \Psi$ of non-reconfigurable devices as part of the synthesis process. Thus, if a droplet corresponding to an input operation is needed on the microfluidic array at $t_{current}$, we schedule the dispensing of the droplet such that it finishes at time $t_{current}$, and not earlier. This is in order to avoid storing the dispensed droplets on the array, until they are needed by other operations, as they will otherwise occupy space

Table 2: Results for the real-life applications

Application	Area	Best		Average		Standard dev.	
		RBS	MBS	RBS	MBS	RBS	MBS
In-vitro (28 operations)	8×9	68.43	72.94	68.77	77.81	0.16	2.12
	8×8	68.87	82.12	69.13	102.37	0.14	13.58
	7×8	69.12	87.33	69.46	111.18	0.17	12.26
Proteins (103 operations)	11×11	113.63	184.06	117.51	205.30	4.65	8.38
	11×10	114.33	185.91	119.62	202.14	6.63	8.84
	10×10	115.65	208.90	120.65	219.17	7.73	7.89

Table 3: Results for synthetic benchmarks

Operations	$Area_1$	$Average_1$		$Best_1$		$Area_2$	$Average_2$		$Best_2$		$Area_3$	$Average_3$		$Best_3$	
		RBS	MBS	RBS	MBS		RBS	MBS	RBS	MBS		RBS	MBS	RBS	MBS
10	6×6	39.92	42.61	39.12	42.61	5×7	39.95	76.1	39.55	76.1	5×6	40.97	102.9	40.46	102.9
20	8×8	50.18	52.71	49.73	52.71	7×8	50.95	53.62	50.5	49.01	7×7	51.74	60.06	51.19	49.81
30	8×8	65.96	72.84	64.73	67	7×8	67.79	84.08	66.92	76.4	7×7	69.68	95.54	68.42	82.49
40	8×8	61.93	102.69	61.18	91.97	7×8	63.74	111.47	63.01	98.25	7×7	65.85	131.63	64.75	99.29
50	9×10	83.89	86.99	83.27	82.4	9×9	84.76	93.5	84.02	87.21	8×9	86.34	101.59	85.37	87.03
60	9×9	94.98	100.44	93.82	89.90	8×10	95.15	104.80	94.34	95.70	8×9	95.85	122.42	94.39	106.7
70	10×10	179.97	194.91	140.4	153.8	9×11	197.05	182.99	155.93	164.01	9×10	186.02	233.57	147.39	162.41
80	10×10	112.98	124.98	112.38	113.4	9×10	113.48	139.26	112.43	124.75	9×9	114.23	147.86	113.6	133.87
90	11×11	139.33	180.64	128.08	127.41	10×10	144.23	215.76	131.32	149.68	9×10	148.59	227.02	136.94	156.31
100	11×11	172.15	325.57	153.06	285.05	10×10	172.46	321.87	154.09	255.97	9×11	170.17	325.66	153.08	278.63

required for performing other operations. Because of the constraint on the number of available reservoirs on a given chip, creating a dispensed droplet at $t_{current}$ is not always possible. In this case, the input operation is bound using a greedy approach to the reservoir that will be available at the earliest time. We use the same approach for determining the binding of detection operations to optical devices.

Due to its randomized nature, the algorithm in Fig. 6 might produce different results for different runs, with the same inputs. The algorithm terminates when all operations have been synthetized, and returns the solution Ψ (line 32). Our route-based synthesis approach is given a time limit, and runs repeatedly RoutingBasedSynthesis from Fig. 6 until the time limit is reached, collecting the best solution Ψ in terms of the application completion time $\delta_{\mathcal{G}}$.

5. EXPERIMENTAL EVALUATION

In order to evaluate our proposed approach, we have used two-real life examples and ten synthetic benchmarks. The GRASP-derived algorithm was implemented in Java (JDK 1.6), running on SunFire v440 computers with UltraSPARC IIIi CPUs at 1,062 GHz and 8 GB of RAM. The module library used for all the experiments is shown in Table 1.

In our experiments we were interested to determine the improvement that can be obtained by using Routing-Based Synthesis (RBS) compared to Module-Based Synthesis (MBS). For MBS, we have used the Tabu Search-based synthesis approach we have proposed in [9].

Table 2 presents the results obtained by using RBS and MBS for two real-life applications: (1) *In-vitro* diagnosis on human physiological fluids (IVD) [15], which has 28 operations and (2) the *colorimetric protein assay* (103 operations) [14], utilized for measuring the concentration of a protein in a solution. Table 2 presents the best solution (in terms of the application completion time $\delta_{\mathcal{G}}$), in columns 3 and 4. The comparison is made for three progressively smaller areas for both approaches, using a time limit of 10 minutes for both synthesis approaches.

As we can see, eliminating the concept of "virtual modules" and allowing the operations to perform on any route on the microfluidic array can lead to significant improvements in terms of application completion time, allowing us to use smaller areas and thus reduce costs. Using routing-based synthesis is particularly important for more constrained synthesis problems, when knowing the exact location of all droplets on the array, leads to more efficient space usage. For example, in the most constrained case for the colorimetric protein assay, the 10×10 array, we have obtained an improvement of 44.95% in the schedule length.

Moreover, the routing-based approach determines a complete solution for the problem, while for the module-based synthesis a post-synthesis step is necessary to determine the routing, which means additional delays.

Both RBS and MBS implementations are stochastic; random decisions during the exploration process can lead to slightly different results. To determine the quality of the RBS implementation, we have run RBS and MBS 50 times. The best results for RBS and MBS, presented in columns 3 and 4 in Table 2, respectively, are collected after 50 runs. The average and standard deviation over the 50 runs compared to the best application completion time $\delta_{\mathcal{G}}$ are also reported in Table 2. As we can see, the difference between RBS and MBS is larger in the average case, and the standard deviation with RBS is very small, which means that RBS consistently finds solutions which are very close to the best solution found over the 50 runs.

In a second set of experiments we have compared RBS with MBS on ten synthetic applications. We have generated a set of synthetic graphs using Task Graphs For Free (TGFF) [5]. We have manually modified the graphs in order to capture the characteristics of biochemical applications. The graphs are composed of 10 up to

100 operations and the results in Table 3 show the best and the average completion time obtained out of 50 runs for RBS and MBS, using a time limit of 10 minutes.

For each synthetic application we have considered three progressively smaller areas. The results in Table 3 confirm the conclusion from Table 2: as the area decreases, performing routing-based synthesis becomes more important, and leads to significant improvements. For example, for the synthetic application with 100 operations, in the case of the 9×11 array, we have obtained an improvement of 47.74% in the average completion time compared with module-based synthesis.

6. CONCLUSIONS

In this paper we have presented a routing based-approach for the synthesis of digital microfluidic biochips. We have shown that by eliminating the concept of "virtual modules" and allowing droplets to move on any route during operation execution significant improvements can be gained, allowing us to use smaller area biochips and thus reduce costs. We have proposed a method for determining the percentages of operation completion for any given route. Using this metric, we have devised an algorithm for routing-based synthesis such that the application completion times is minimized. Two real life examples and ten synthetic applications have been used for evaluating the effectiveness of the proposed algorithm, compared to module-based synthesis.

7. REFERENCES

[1] K. F. Bohringer. Towards optimal strategies for moving droplets in digital microfluidic systems *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1468–1474, 2004.

[2] K. Chakrabarty and J. Zeng. *Design automation methods and tools for microfluidic-based biochips*. Springer, 2006.

[3] K. Chakrabarty and J. Zeng. Design automation for microfluidics-based biochips. *ACM Journal on Emerging Technologies in Computing Systems*, 1(3):186–223, 2005.

[4] M. Cho and D. Z. Pan. A high-performance droplet router for digital microfluidic biochips. In *Proceedings of International Symposium on Physical Design*, pages 200–206, 2008.

[5] R. P. Dick, D. L. Rhodes, and W. Wolf. TGFF: task graphs for free. In *Proceedings of the Sixth International Workshop on Hardware/Software Codesign*, pages 97–101, 1998.

[6] R. B. Fair. Digital microfluidics: is a true lab-on-a-chip possible? *Microfluidics and Nanofluidics*, 3(3):245–281, 2007.

[7] T. A. Feo and M. G. C. Resende. Greedy Randomized Adaptive Search Procedure. *Journal of Global Optimization*, 6:109-133, 1995.

[8] E. J. Griffith, S. Akella, and M. K. Goldberg. Performance characterization of a reconfigurable plannar array digital microfluidic system. *TCAD*, 25:340–352, 2006.

[9] E. Maftei, P. Paul, and J. Madsen. Tabu Search-Based Synthesis of Dynamically Reconfigurable Digital Microfluidic Biochips. In *Proceedings of the Compilers, Architecture, and Synthesis for Embedded Systems Conference*, pages 195–203, 2009.

[10] E. Maftei, P. Paul, J. Madsen, and T. Stidsen. Placement-aware architectural synthesis of digital microfluidic biochips using ILP. In *Proceedings of the International Conference on Very Large Scale Integration of System on Chip*, pages 425–430, 2008.

[11] P. Paik, V. K. Pamula, and R. B. Fair. Rapid droplet mixers for digital microfluidic systems. In *Lab on a Chip*, 3:253–259, 2003.

[12] M. G. Pollack, A. D. Shenderov, and R. B. Fair. Electrowetting-based actuation of droplets for integrated microfluidics. *Lab Chip Journal*, 2:96–101, 2002.

[13] H. Ren, V. Srinivasan, and R. B. Fair. Design and testing of an interpolating mixing architecture for electrowetting-based droplet-on-chip chemical dilution. In *Proceedings of the International Conference on Transducers, Solid-State Sensors, Actuators and Microsystems*, pages 619–622, 2003.

[14] F. Su and K. Chakrabarty. Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips. In *Proceedings of the 42nd annual Conference on Design Automation*, pages 825–830, 2005.

[15] F. Su, W. Hwang, and K. Chakrabarty. Droplet routing in the synthesis of digital microfluidic biochips. In *Proceedings of Design, Automation and Test in Europe*, volume 1, pages 73–78, 2006.

[16] T. Thorsen, S. Maerkl, and S. Quake. Microfluidic largescale integration. *Sci.*, pages 580–584, 2002.

[17] T. Xu and K. Chakrabarty. Integrated droplet routing and defect tolerance in the synthesis of digital microfluidic biochips. In *Proceedings of Design Automation Conference*, pages 948–953, 2007.

[18] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang. Placement of digital microfluidic biochips using the T-tree formulation. In *Proceedings of Design Automation Conference*, pages 931–934, 2006.

[19] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang. BioRoute: A Network-Flow-Based Routing Algorithm for the Synthesis of Digital Microfluidic Biochips. In *Proceedings of International Conference on Computer-Aided Design*, pages 752–757, 2007.