BioRoute: A Network-Flow-Based Routing Algorithm for the Synthesis of Digital Microfluidic Biochips

Ping-Hung Yuh, Chia-Lin Yang, *Member, IEEE*, and Yao-Wen Chang, *Member, IEEE*

***Abstract*—Due to recent advances in microfluidics, digital microfluidic biochips are expected to revolutionize laboratory procedures. One critical problem for biochip synthesis is the droplet routing problem. Unlike traditional very large scale integration routing problems, in addition to routing path selection, the biochip routing problem needs to address the issue of scheduling droplets under practical constraints imposed by the fluidic property and timing restriction of synthesis results. In this paper, we present the *first* network-flow-based routing algorithm that can concurrently route a set of noninterfering nets for the droplet routing problem on biochips. We adopt a two-stage technique of global routing followed by detailed routing. In global routing, we first identify a set of noninterfering nets and then adopt the network-flow approach to generate optimal global-routing paths for nets. In detailed routing, we present the *first* polynomial-time algorithm for simultaneous routing and scheduling using the global-routing paths with a negotiation-based routing scheme. Our algorithm targets at both the minimization of cells used for routing for better fault tolerance and minimization of droplet transportation time for better reliability and faster bioassay execution. Experimental results show the robustness and efficiency of our algorithm.**

***Index Terms*—Detailed routing, digital microfluidic biochips, global routing, network-flow-based algorithm.**

I. Introduction

DUE TO the advances in microfabrication and microelectromechanical systems, microfluidic technology has gained much attention recently. The composite microsystems could perform conventional biological laboratory procedures on a small and integrated system by manipulating microliter or nanoliter fluids. Therefore, microfluidic biochips are used in several common procedures in molecular biology, such as clinical diagnostics and deoxyribonucleic acid sequencing analysis.

First-generation (analog) microfluidic biochips are based on manipulating continuous liquid flow by permanently etched microchannels and external pressure sources (e.g., micropumps). Recently, *second-generation* (digital) microfluidic biochips, which are based on the manipulation of discrete microliter or nanoliter liquid particles (the *droplets*), have been proposed [2]. In digital microfluidic biochips, each droplet can be independently controlled by electrohydrodynamic forces generated by an electric field. The field can be generated by an individually accessible electrode. Compared with the first-generation biochips, droplets can move anywhere in a 2-D array to perform desired chemical reactions, and electrodes can be reprogrammed for different bioassays.

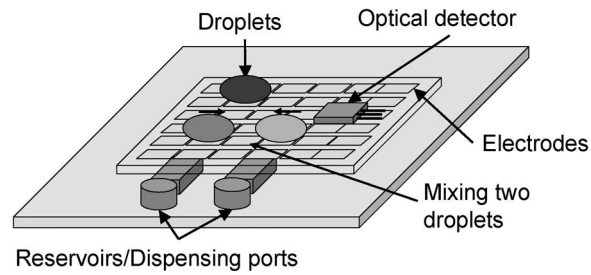


Fig. 1. Schematic view of digital microfluidic biochips.

Fig. 1 shows the schematic view of a digital microfluidic biochip based on the principle of electrowetting on dielectric [3]. There are three major components in a biochip: 2-D microfluidic array, dispensing ports/reservoirs, and optical detectors. The 2-D microfluidic array contains a set of basic cells (i.e., unit cells for droplet movement). A droplet moves one cell in one clock cycle. Each basic cell has identical architecture and is used to perform various fundamental operations, such as mixing of multiple droplets, droplet transportation, droplet dilution, and droplet fission. Note that we can perform these fundamental operations anywhere on the 2-D array. In other words, a portion of the 2-D array can perform different operations at different times. This property is referred to as the *reconfigurability* of biochips. Moreover, we can use these fundamental operations to build a complex bioassay. We refer to this property as the *scalability* of biochips. The dispensing ports/reservoirs are responsible for droplet generation while the optical detectors are used for reaction detection. These three components allow researchers to perform laboratory procedures on a biochip, from sample preparation, reaction, to detection.

Similar to traditional very large scale integrated (VLSI) synthesis methodology, a top–down synthesis approach for digital microfluidic biochips has been proposed [2]. The synthesis step is divided into architectural- and geometry-level syntheses.

Manuscript received March 11, 2008; revised June 27, 2008. Current version published October 22, 2008. This work was supported in part by the National Science Council of Taiwan under Grants NSC 96-2752-E-002-008-PAE, NSC 96-2628-E-002-248-MY3, NSC 96-2628-E-002-249-MY3, and NSC 96-2221-E-002-245 and in part by the Excellent Research Projects of National Taiwan University under Grant 96R0062-AE00-07. An earlier version of this paper was presented at the IEEE/ACM International Conference on Computer-Aided Design 2007 [1]. This paper was recommended by Associate Editor K. Chakrabarty.

P.-H. Yuh and C.-L. Yang are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: r91089@csie.ntu.edu.tw; yangc@csie.ntu.edu.tw).

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TCAD.2008.2006140

0278-0070/$25.00 © 2008 IEEE

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

The architectural-level synthesis performs scheduling with a given bioassay and a set of design specifications [4], while the geometry-level synthesis performs physical placement to determine the physical location of each operation as well as other geometry details [5]. Recently, a unified synthesis and physical placement approach has been proposed [6]–[8]. However, there is not much work that handles the droplet routing problem [9]–[11].

The main challenge of droplet routing is ensuring the correct execution of bioassays; the fluidic property that prevents unexpected mixing among droplets needs to be satisfied. Unlike traditional VLSI routing, in addition to routing path selection, the biochip routing problem needs to address the issue of scheduling droplets under practical constraints imposed by the fluidic property and timing restriction of synthesis results.

There are two main optimization objectives for droplet routing. The first objective is to minimize the number of cells used for routing for better fault tolerance, which is important for safety-critical applications, such as patient health monitoring or biosensors for detecting environmental toxins. As discussed in [5], a biochip contains primary cells for bioassay execution and spare cells for replacing faulty primary cells to ensure the correctness of bioassay execution. Since droplets can only be routed through spare cells, to maximize the number of spare cells for fault tolerance, we need to minimize the number of cells for droplet routing. The second objective is to shorten droplet transportation time, i.e., minimizing the time (in cycles) to route all droplets. Droplet transportation time is critical for applications requiring real-time response for early warnings, such as monitoring environmental toxins. Moreover, shorter droplet transportation time improves the reliability of a biochip. Longer transportation time implies that high actuation voltage (up to 90 V [12]) must be maintained for a long period of time, thereby accelerating dielectric breakdown on some cells. Droplet transportation time is also critical for maintaining the integrity of bioassay execution. Biological samples are sensitive to environmental variations. For example, many biological reactions require very small temperature variation (within 1 °C [13]). Unfortunately, it is hard to maintain an optimal laboratory environment on a biochip. Therefore, it is desirable to shorten the droplet transportation time to maintain the integrity of bioassay execution.

A. Previous Work

In the literature, there are three methods to solve the droplet routing problem. The first one is the prioritized A*-search algorithm [9]. Each droplet is assigned a priority, and the A*-search algorithm is used to coordinate each droplet based on its priority. The drawback of this approach is that they did not consider the practical timing constraint for throughput consideration. Moreover, they only considered two-pin nets. However, for practical bioassays, droplet routing must be modeled as multipin nets, since droplets connect multiple terminals for a mix assay operation.

The second one is based on the open shortest path first routing protocol [10]. They defined layout patterns of a biochip. Each layout pattern has a routing table that is computed by Dijkstra's shortest path algorithm. We can then route each droplet based on this routing table. However, since droplet routing only occurs at these layout patterns, their algorithm did not exploit the dynamic reconfiguration property of digital microfluidic biochips.

The third one is a two-stage algorithm [11]. In the first stage (alternative routing path generation), a set of shortest routing paths for each droplet is generated by maze routing. In the second stage (random selection and scheduling), a random selection approach is used to randomly select a routing path for each droplet. A scheduling approach is used to schedule droplets based on the selected routing paths. The aforementioned procedure (random selection and scheduling) repeats for an adequate number of iterations to find a feasible solution. There are two drawbacks of this approach. First, since droplet routing and scheduling are separated into two stages without considering the interaction between them, this approach may not find a good solution. Second, this approach may not be efficient, since the maze routing algorithm is performed multiple times to generate alternative routing paths for each droplet.

B. Our Contribution

In this paper, we propose the *first* network-flow-based routing algorithm for the droplet routing problem on digital microfluidic biochips. The network-flow routing approach can concurrently route a set of noninterfering nets and obtain optimal routing solutions in polynomial time. To tackle the complexity issue of simultaneously considering routing and scheduling, we adopt a two-stage technique of global routing followed by detailed routing. In global routing, we first identify a set of noninterfering nets and then adopt the network-flow approach to generate optimal global-routing paths for the identified nets. In detailed routing, we present the *first* polynomial-time algorithm for simultaneous routing and scheduling with a negotiation-based routing scheme based on the global-routing paths in the context of biochip routing.

In this paper, we also present how to handle three-pin nets for practical bioassays. As discussed in [11] and [14], for a mix operation, mixing time can be greatly reduced if its two input droplets are mixed during their transportation. Since mix operations are one of the fundamental operations of a bioassay, it is important to induce the mixing of droplets before reaching their destinations. Hence, these two input droplets must be modeled as three-pin nets, instead of two two-pin ones.

Experimental results demonstrate the robustness and efficiency of our algorithm. Our algorithm can successfully route all benchmarks while previous works cannot. Moreover, our algorithm can achieve better solution quality in reasonable CPU time. For example, for the *in vitro* diagnostics, our algorithm achieves an 11.23% fewer number of cells used for routing (237 versus 267) with less CPU time (0.05 versus 0.15 s) than the two-stage algorithm proposed in [11]. Our algorithm also outperforms previous works in minimizing the number of cycles to route all bioassays. For example, for the same bioassay, our algorithm obtains a routing solution requiring less

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

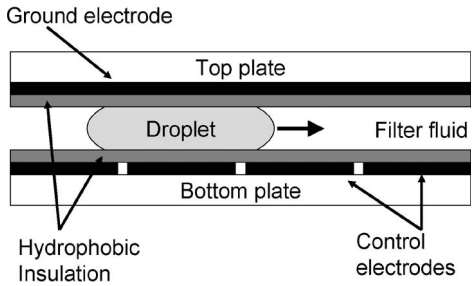


Fig. 2. Side view of a 2-D microfluidic array.

routing time[1] than the two-stage algorithm proposed in [11] (1.16 versus 2.22) with less CPU time (0.05 versus 0.17 s).

The remainder of this paper is organized as follows. Section II describes droplet routing on biochips and formulates the droplet routing problem. Section III details our routing algorithm. Section IV shows the experimental results. Finally, concluding remarks are given in Section V.

II. Routing on Biochips

In this section, we first show droplet routing on biochips. Then, we detail the routing constraints for droplet routing. Finally, we present the problem formulation of the droplet routing problem.

A. Droplet Routing

Fig. 2 shows the side view of a 2-D microfluidic array. A droplet is sandwiched by two plates. The top plate contains one ground electrode, and the bottom plate contains a set of control electrodes. A droplet moves to an adjacent electrode when this electrode is activated. A droplet can stay at a cell for a period of time if we do not activate its neighboring electrodes. Fig. 3 shows a droplet routing example. Fig. 3(a) shows a task graph to represent a bioassay and a 3-D module placement with three modules to represent a synthesis result. In a task graph, nodes represent assay operations, and edges represent data dependences among operations. A 3-D module placement can be divided into a set of 2-D planes at different time steps due to the ability of dynamic reconfiguration [7]. Droplet movement among modules only occurs at these 2-D planes. For example, the 3-D placement shown in Fig. 3(a) can be divided into two 2-D planes, one representing the time $t1$ before the execution of the two dilute operations a and c and the other one representing the time $t2$ when dilute a is finished. Fig. 3(b) shows the corresponding two 2-D planes. Note that each module is wrapped with segregation cells for functional isolation.

The droplet routing problem is to route all droplets from a reservoir/dispensing port to a target pin [such as the solid lines shown in Fig. 3(b)], from a source pin to a target pin [such as the dashed lines shown in Fig. 3(b)], or from a source pin to a waste reservoir [such as the dotted lines shown in Fig. 3(b)]. A *pin* is defined as a fluidic port on the boundary of a module. Since droplets are generated before routing, the source pin of a droplet generated by a reservoir is the cell next to this reservoir. To satisfy the fluidic property for correct droplet movement,[2] a droplet may stay at a basic cell for a period of time. Therefore, in addition to determining the routing path for each droplet, we need to schedule each droplet to satisfy the fluidic property, i.e., to determine the arrival and departure times of each droplet on each basic cell. Only modules (and the surrounding segregation cells) that are active during droplet routing on one 2-D plane are considered as obstacles. For example, dilute c is considered as an obstacle at time $t2$ since this operation is active at $t2$. To obtain a complete routing solution, we can sequentially route each 2-D plane to determine the routing path and schedule of each droplet.

The fluidic route of a droplet can be modeled either as a two- or three-pin net. For a dilute operation, we model each input droplet as a two-pin net with only one droplet. For example, the two input droplets from the reservoirs to dilute a at time $t1$ can be modeled as two two-pin nets. However, for a mix operation, we need to model two input droplets as a three-pin net due to the preference of merging two droplets during their transportation for an efficient mix assay operation [11].[3] With this modeling, the two input droplets will be merged before reaching their sink. For example, in Fig. 3(b), the two droplets of the mix b operation form a three-pin net. A droplet routing algorithm must be capable of handling both two- and three-pin nets. We use d_j^a to denote the jth droplet of net n_a. If n_a is a two-pin net, j is always 1; otherwise, $j = 1$ or 2. For a two-pin net n_a, we also use d^a to denote the droplet of n_a.

B. Routing Constraints

There are two routing constraints in droplet routing: *fluidic and timing constraints*. The fluidic constraints are used to avoid an unexpected mixing between two droplets of different nets during their transportation, while the timing constraint states the maximum allowed transportation time of a droplet.

The fluidic constraints can be further divided into the *static* and *dynamic* fluidic constraints [11]. The static fluidic constraint states that the minimum spacing between two droplets is two cells if the Cartesian coordinate system is used. In other words, if a droplet is located at cell c at time t, then there does not exist any droplet at the neighboring cells of c at time t. The dynamic fluidic constraint is related to two moving droplets: If a droplet d_p moves to cell c at time $t+1$, then there must not be any other droplet d_q that moves to cell c' at time $t+1$ and locates at one of the neighboring cells of c at time t. The reason is that since both cells c and c' are activated, d_q may stay at its original location due to these two opposing electrohydrodynamic forces. As a result, an unexpected droplet mixing may occur.

Besides the fluidic constraints, there exists the timing constraint. The timing constraint specifies the maximum allowed transportation time of a droplet from its source to its target. Since droplet movement is relatively fast compared to assay

[1]Routing time is measured as the maximum droplet transportation time over the maximum Manhattan distance of all nets.

[2]The fluidic property will be formally described in Section II-B.

[3]Droplets of the same net have the same target pin.

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

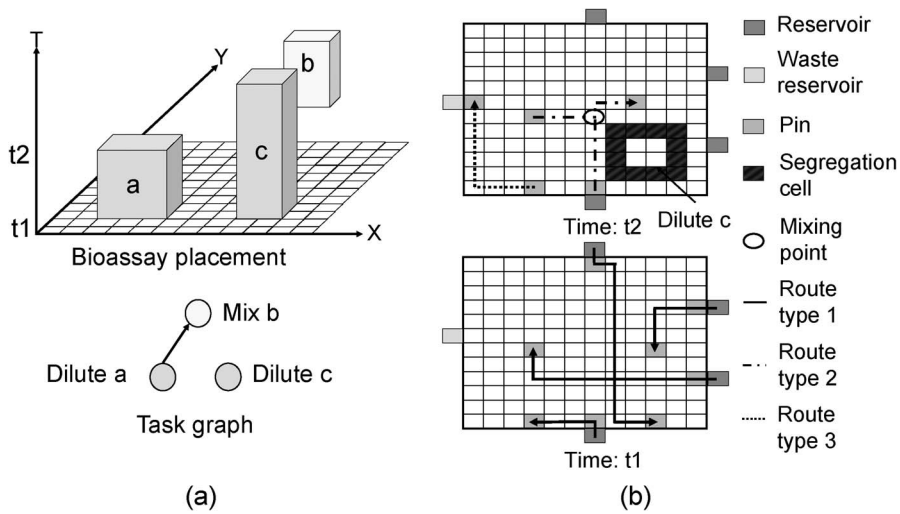


Fig. 3. Example of droplet routing on biochip. (a) A task graph and a 3-D module placement, i.e., a synthesis result. (b) The corresponding two 2-D planes. Droplet routing only occurs at these 2-D planes.

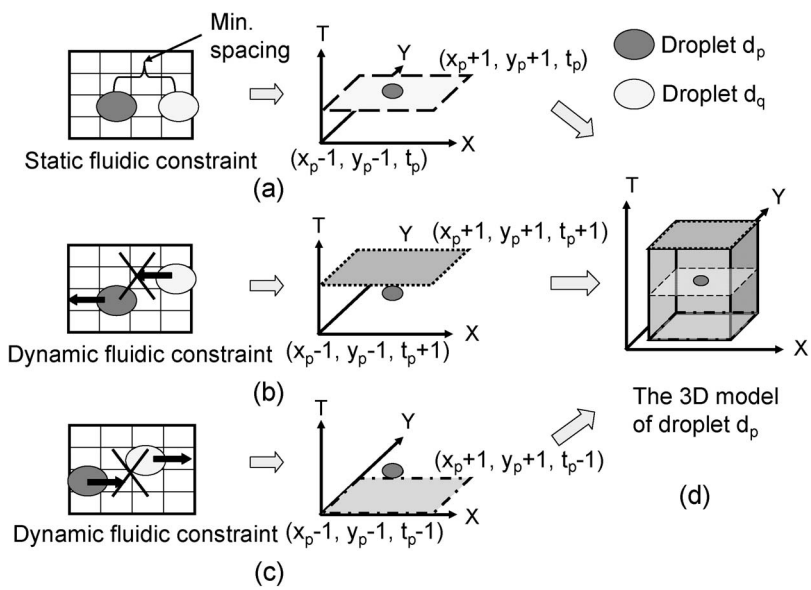


Fig. 4. Example of fluidic constraints. (a) The static fluidic constraint. (b) The dynamic fluidic constraint when a droplet d_q moves to one of the neighboring cells of cell (x_p, y_p) at time t_p. (c) The dynamic fluidic constraint when a droplet d_p moves to cell (x_p, y_p) at time t_p. (d) The 3-D modeling of d_p, where d_p is located at the center of this 3-D cube.

operations, the existing synthesis algorithms of biochips [6], [7] usually ignore droplet transportation time. To ensure that the aforementioned assumption is valid for complex bioassays, the droplet transportation time must be within a maximum value. Note that we need to account for a droplet's idle time when calculating the transportation time of this droplet.

C. Modeling the Routing Constraints

We first detail how to model the fluidic constraints. The fluidic constraints can be illustrated in three scenarios as shown in Fig. 4. The $X(Y)$ dimension represents the width (height) of a biochip, and the T dimension represents the droplet transportation time. Let (x_p, y_p, t_p) be the coordinate of droplet d_p in this 3-D space to represent the location of d_p at time t_p. To satisfy the static fluidic constraint, there exist no other droplets in the 2-D rectangle defined by the two coordinates $(x_p - 1, y_p - 1, t_p)$ and $(x_p + 1, y_p + 1, t_p)$ in the 3-D space, as shown in Fig. 4(a). For the dynamic fluidic constraint, we need to consider two cases. First, when d_q moves to one of the neighboring cells of (x_p, y_p) at time t_p, to satisfy the dynamic fluidic constraint, no other droplets can be in the 2-D rectangle defined by the two coordinates $(x_p - 1, y_p - 1, t_p + 1)$ and $(x_p + 1, y_p + 1, t_p + 1)$ in the 3-D space, as shown in Fig. 4(b). Second, when d_p moves to cell (x_p, y_p) at time t_p, to satisfy the dynamic fluidic constraint, there exist no other droplets in the 2-D rectangle defined by the two coordinates $(x_p - 1, y_p - 1, t_p - 1)$ and $(x_p + 1, y_p + 1, t_p - 1)$ in the 3-D space, as shown in Fig. 4(c). Therefore, the three rectangles identified in the aforementioned three scenarios form a $3 \times 3 \times 3$ 3-D cube in the 3-D space as shown in Fig. 4(d), where d_p is located at the center of this 3-D cube. Given a routing solution, the fluidic constraints are

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

satisfied if, for each droplet d_p located at cell c at time t, there exist no other droplets in the 3-D cube defined by d_p.

Now, we present how the timing constraint is modeled. The notations and definitions used in modeling the timing constraint will be used in the droplet routing algorithm described in Section III. Given the timing constraint $T_{\max}$, we define $T_s^m(c, d_j^i)(T_s^M(c, d_j^i))$ as the earliest (latest) time that the droplet d_j^i of net n_i can reach (stay at) a cell c without violating the timing constraint, where $T_s^m(c, d_j^i) = m_d(c, s_j^i)(T_s^M(c, d_j^i) = T_{\max} - m_d(c, \hat{t}^i))$, s_j^i represents the source cell of the droplet d_j^i, $\hat{t}^i$ represents the target cell of net n_i, and $m_d(c_1, c_2)$ represents the Manhattan distance between two cells c_1 and c_2. We say that a cell c is *available* to a droplet d_j^i if $T_s^M(c, d_j^i) \geq T_s^m(c, d_j^i)$ and no obstacle is located at c. Moreover, c is available to d_j^i at time t if $T_s^M(c, d_j^i) \geq t \geq T_s^m(c, d_j^i)$. Similarly, c is available to a net n_i if c is available to at least one droplet of n_i, and c is available to n_i at time t if c is available to at least one droplet of n_i at time t. The time interval that a droplet can stay at a cell without violating the timing constraint is referred to as the *idle interval*. We use di_{ij}^c to denote the idle interval of droplet d_j^i at cell c, where di_{ij}^c is defined as $[T_s^m(c, d_j^i), T_s^M(c, d_j^i)]$ if c is available to d_j^i. We also define the *violation interval* vi_{ij}^c of d_j^i at cell c as the time interval $[T_s^m(c, d_j^i) - 1, T_s^M(c, d_j^i) + 1]$. If another droplet is scheduled in c or c's neighboring cells during the violation interval of c and d_j^i is scheduled at c in its idle interval, then the fluidic constraints may be violated. We say that a cell c is used for routing if at least one droplet uses c for routing.

D. Problem Formulation

Since we can sequentially route each 2-D plane to form a complete droplet routing solution, we only show the problem formulation of one 2-D plane. Other 2-D planes can be handled similarly. In this paper, we consider two problems. The first one is to minimize the number of cells used for routing for better fault tolerance. The problem formulation is given as follows.

Input) A netlist of m nets $N = \{n_1, n_2, \ldots, n_m\}$, where each net n_a is a two- (one droplet) or three-pin net (two droplets), the locations of pins and obstacles, and the timing constraint $T_{\max}$.

Objective) Minimize the number of cells used for routing for better fault tolerance.

Constraint) Both fluidic and timing constraints must be satisfied. The second problem is to minimize the maximum droplet transportation time for fast bioassay execution or better reliability. The problem formulation is given as follows.

Input) A netlist of m nets $N = \{n_1, n_2, \ldots, n_m\}$, where each net n_a is a two- (one droplet) or three-pin net (two droplets), and the locations of pins and obstacles.

Objective) Minimize the maximum droplet transportation time for better reliability.

Constraint) The fluidic constraints must be satisfied.

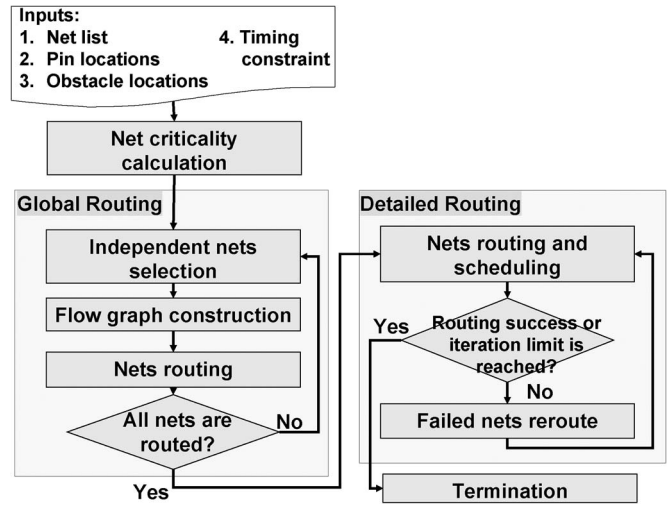


Fig. 5. Droplet routing algorithm overview.

III. Biochip Routing Algorithm

In this section, we present our biochip routing algorithm. We first give the overview of the proposed routing algorithm. Then, we detail each phase of our algorithm in subsequent sections with the optimization objective of minimizing the number of cells used for routing. Finally, we show how to extend our algorithm to handle the timing-aware routing problem.

A. Routing Algorithm Overview

Fig. 5 shows the overview of the proposed routing algorithm. There are three phases in our routing algorithm: 1) net criticality calculation; 2) global routing based on the min-cost max-flow (MCMF) algorithm [15]; and 3) detailed routing based on a negotiation-based routing algorithm.

In net criticality calculation, we determine the criticality of each net. A net is said to be critical if it is difficult to route this net, due to the severe interferences with other nets or a tight timing constraint. This criticality information will be used in both global and detailed routing.

In global routing, the goal is to determine a rough routing path of each droplet. We divide a biochip into a set of global cells. We first select a set of *independent nets*[4] that do not interfere with each other. Based on these global cells, we construct the flow network. We then apply the MCMF algorithm to route the selected nets with the constructed flow network.

In detailed routing, the goal is to simultaneously perform routing and scheduling based on the result of global routing. Scheduling a droplet is equivalent to determining the arrival and departure times of this droplet on each cell. We propose a negotiation-based routing algorithm to handle the detailed routing. The negotiation-based routing algorithm terminates when a feasible solution is found or a specified maximum number of iterations is reached.

[4]The formal definition of independent nets will be given in Section III-C.

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

TABLE I
NOTATIONS USED IN GLOBAL ROUTING

N	set of nets
N'	set of independent nets
a, b	index of nets
p, q	index of cells
d_j^a	the j-th droplet of net n_a
C_a	a set of available cells in the bounding box of net n_a
C'_p	set of cell c_p and c_p's neighbors
$crit(a)$	criticality of the net n_a
$\hat{c}$	a global cell
$\hat{U}_p$	capacity of the global cell $\hat{c}_p$
U	the largest capacity of all edges in the flow graph
O_p	number of droplets that use $\hat{c}_p$ for routing
O_p^n	number of nets in N' that can use $\hat{c}_p$ for routing
$\phi_g(v_p^a)$	cost of the node v_p^a
ϕ_{max}	the largest cost of all nodes
L_p	union of all idle intervals of cell c in the global cell $\hat{c}_p$
W_c/H_c	width/height of a biochip
Q	a priority queue

B. Net Criticality Calculation

A net n_a is said to be critical if 1) n_a has fewer possible solutions (routing paths and schedules) due to the timing constraint or 2) there are more nets whose solutions affect the solution of n_a. We use crit(a) to denote the criticality value of n_a. crit(a) is defined by the following:

$$\text{crit}(a) = \frac{\sum_{k\in N}\sum_{c\in C_a}\sum_{t\in vi_{a1}^c\cup vi_{a2}^c} u(c,k,t)}{\sum_{c\in C_a}\sum_{t\in di_{a1}^c\cup di_{a2}^c} u(c,a,t)} \quad (1)$$

where C_a is the set of available cells in the bounding box of n_a and $u(c,a,t)$ is one if c is available to net n_a at time t; otherwise, $u(c,a,t)$ is zero. The larger the crit(a) is, the more critical the n_a is. The reason is that, with a tighter timing constraint, there are fewer possible routing solutions for n_a, and thus, the denominator is decreased. If there are more nets that might use cells in C_a for routing, the value of the numerator is increased, meaning that it is more difficult to route n_a without violating the fluidic constraints.

C. Global Routing

Global routing is to determine a rough routing path for each droplet. We decompose the global routing problem into a set of subproblems by selecting a set of independent nets that do not interfere with each other. We first explain how to select a set of independent nets. Then, we present the MCMF algorithm to solve the global routing problem with the selected nets and the approach to estimate the capacity of a global cell. Finally, we handle three-pin nets with the MCMF algorithm. Table I shows the notations used in global routing.

1) Net Selection: We first give the following two definitions.

Definition 1: A cell c is said to be a *violation-free cell* for two nets n_a and n_b if it is guaranteed to satisfy the fluidic constraints when the droplet d_i^a uses c and the droplet d_j^b uses one of c's neighboring cells or c for routing.

Definition 2: Two nets are said to be *independent nets* if 1) their bounding boxes are not overlapped or adjacent, or 2) all cells in the overlapping area (if they are overlapped) or all boundary cells (if they are adjacent) are violation-free cells.

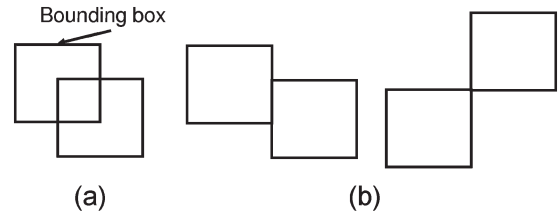


Fig. 6. Illustration of Definition 2. (a) Two boundaries are overlapped. (b) Two boundaries are (diagonally) adjacent.

Fig. 6 shows Definition 2. Based on the aforementioned two definitions, the goal of the net selection process is to select a set of independent nets with the maximum sum of criticality, since we should route critical nets first. First, we define C'_p as the set of c_p and c_p's neighboring cells. The cell c_p is not violation free for two nets n_a and n_b if there exists a cell $c_q \in C'_p$ such that the violation interval $vi_{ai}^{c_p}$ of droplet d_i^a overlaps with the idle interval $di_{bj}^{c'_q}$ of droplet d_j^b. If these two intervals are overlapped, it means that it is possible to violate the fluidic constraints if d_i^a uses c_p and d_j^b uses c_q for routing. If the bounding boxes of n_a and n_b are overlapped and at least one cell in the overlapping area is not violation free, then n_a and n_b are not independent nets. Similarly, if the bounding boxes of n_a and n_b are adjacent and at least one boundary cell is not violation free, then n_a and n_b are not independent nets.

Now, we present how to select a set of independent nets for routing. We construct an undirected *conflict graph* G_c for net selection. For each unrouted net n_a, we create a corresponding node v_a in G_c. The weight of v_a is its criticality. Two nodes v_a and v_b are connected if n_a and n_b are not independent. Under this formulation, the net selection problem is equivalent to the maximum weighted independent set (MWIS) problem, which is NP-complete on general graphs [16]. Therefore, we resort to an efficient heuristic to find the MWIS of G_c.

We use a priority queue Q to find the MWIS of G_c. Q contains all candidate nets, and the priority is the criticality of each net. We iteratively select the most critical net n_a in Q and delete all nodes v_b that are connected by v_a and v_a in G_c. We also remove n_b and n_a from Q. When Q becomes empty, we find the set of independent nets, denoted by N'. N' will be used as the input to the MCMF algorithm. The time complexity of the net selection process is $O(|N|^3)$, since we need to visit at most $O(|N|^2)$ edges per iteration, and there are at most $|N|$ iterations. Finally, Fig. 7 shows our net selection algorithm.

2) Network-Flow-Based Routing Algorithm: We divide a biochip into a set of global cells $\hat{c}$. Each global cell contains 3×3 basic cells. With the global cells and the selected set N' of independent nets to represent a subproblem, we use the MCMF algorithm to solve each subproblem. We first present the basic network formulation for routing all two-pin nets. Finally, we explain how to handle three-pin nets with the MCMF algorithm.

Basic network formulation: We create a directed graph $G_f = (V_g \cup \{s_f, \hat{t}_f\}, E_g)$, where $s_f(\hat{t}_f)$ is the source (target) of G_f, V_g is the set of routing nodes, and E_g is the set of edges. For each droplet d^a, we create a node v_p^a for each global cell $\hat{c}_p$ if at least one cell c in $\hat{c}_p$ is available to d^a and c is in the bounding box of n_a. Note that, under this construction,

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

Algorithm: Net Selection (N)
N: set of un-selected nets;
begin
1 $N' = \emptyset$;
2 Construct the conflict graph G_c;
3 Add all nets in N into the priority queue Q;
4 **while** Q is not empty
5 Let n_a be the most critical net in Q;
6 Delete n_a from Q;
7 Add n_a into N';
8 Delete all nodes v_b in G_c if v_b and v_a are connected and all n_b in Q;
9 $N = N - N'$;
10 **return** N';
end

Fig. 7. Summary of the net selection algorithm.

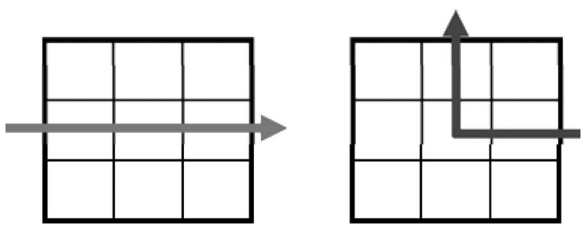


Fig. 8. Illustration of the simple routing path assumption when a droplet passes through a global cell.

multiple nodes may correspond to the same global cell since c may be available to multiple droplets. Each node v_p^a has a capacity $\tilde{U}_p - O_p$ and its cost $\phi_g(v_p^a)$.

To calculate $\tilde{U}_p$, we first define the set of $L_p = \{l_1^p, l_2^p, \ldots, l_k^p\}$ as the union of all idle intervals of each droplet d_j^a at each cell c in $\hat{c}_p$. L_p represents all possible time instances that a droplet may use $\hat{c}_p$ for routing. To determine the capacity of $\hat{c}_p$, we first assume that each droplet passes through a global cell with a straight line or an L-shaped routing path, as shown in Fig. 8. Under this assumption, if a droplet arrives at $\hat{c}_p$ at time t, then this droplet leaves $\hat{c}_p$ at time $t+2$. It also means that to satisfy the fluidic constraints, there exist no other droplets scheduled in the time interval $[t-1, t+3]$ based on the 3-D modeling of the fluidic constraints presented in Section II. In other words, we need five time units for a droplet to pass through a global cell without violating the fluidic constraints. Then, $\tilde{U}_p$ is calculated by the following:

$$\tilde{U}_p = \sum_{l_k^p \in L_p} \left\lfloor \frac{|l_k^p| + 2}{5} \right\rfloor \tag{2}$$

where $|l_k^p|$ is the range of l_k^p and is defined as the difference of the two endpoints of l_k^p plus one. Moreover, to consider the case when a droplet arrives or leaves at one of the two endpoints of l_k^p, the numerator is increased by two.

For two adjacent global cells, $\hat{c}_p$ and $\hat{c}_q$, we create two directed edges between v_p^a and v_q^a for all nets n_a. The costs and capacities of these two edges are both zero and one, respectively. For all droplets d^a, we create an edge from s_f to v_p^a with capacity one and cost zero if the source of d^a is in $\hat{c}_p$. Similarly, we create an edge from v_p^a to $\hat{t}_f$ if the target of d^a is in $\hat{c}_p$ with capacity one and cost zero. We will detail how to transfer the node cost/capacity to the edge cost/capacity later. Fig. 9 shows an example of our network formulation. Fig. 9(a) shows the 2-D plane at time $t1$ of the 3-D module placement shown in Fig. 3(a). The whole chip is divided into 16 global cells. The bottom-left global cell is labeled as 0 while the upper-right global cell is labeled as 15. Fig. 9(b) shows the corresponding network flow formulation. For simplicity, we only show nets n_1 and n_2. As shown in this figure, there are two nodes that represent the same global cell $\hat{c}_6$. Similarly, there are two nodes that represent the same global cell $\hat{c}_7$.

Cost assignment and node construction: The cost $\phi_g(v_p^a)$ of the global cell $\hat{c}_p$ for net n_a is defined by the following:

$$\phi_g\left(v_p^a\right) = \begin{cases} |N'| - O_p^n, & O_p \neq 0 \\ 1 + |N'| - O_p^n, & \text{otherwise} \end{cases} \tag{3}$$

where $|N'|$ is the size of the input nets. Since our goal is to minimize the number of cells used for routing, we encourage multiple droplets to share the same global cell by assigning a smaller cost to v_p^a if O_p is not zero. Moreover, to encourage two nets in N' to share the same global cell, we add the difference of $|N'|$ and O_p^n into the cost function.

Now, we present how to transfer the node cost/capacity to the edge cost/capacity so that the MCMF algorithm can be applied. Since each node v_p^a has capacity $\tilde{U}_p - O_p$, it means that the number of outgoing flows cannot exceed $\tilde{U}_p - O_p$. The same condition holds for all incoming flows of v_p^a. We use the node split technique [15] to decompose each node v_p into two intermediate nodes v_p' and v_p'', and an edge is connected from v_p' to v_p''. Fig. 10 shows this technique. All outgoing edges of v_p are now connected from v_p'', and all incoming ones are now connected to v_p'. The cost and capacity of the edge from v_p' to v_p'' are $\phi_g(v_p^a)$ and $\tilde{U}_p - O_p$, respectively.

Under our flow-network construction, one special situation occurs when the size of N' is larger than the capacity of a node v_p. In this situation, it is possible that, after applying the MCMF algorithm, O_p is larger than the capacity of $\hat{c}_p$, since in our formulation, multiple nodes represent the same global cell and these nodes may be used by different droplets for routing. Assume that nets n_a and n_b [crit$(b) >$ crit(a)] both use $\hat{c}_p$ for routing and $O_p > \tilde{U}_p$. We then keep the flow of n_b and rip up and reroute n_a without using $\hat{c}_p$ for routing. If n_a cannot reach its sink without using $\hat{c}_p$, then the flow of n_a is restored, and we rip up and reroute n_b. If both n_a and n_b fail for routing, then we keep the flow of n_b (since it is more critical) and treat n_a as a failed net. Then, n_a will be handled in the detailed routing stage. The aforementioned process repeats until $O_p \leq \tilde{U}_p$ for all global cells $\hat{c}_p$.

Based on the aforementioned network formulation and the cost/capacity assignment of edges, we have the following theorem.

Theorem 1: Given a set N' of independent two-pin nets with its size not larger than the minimum capacity of all nodes, we can apply the MCMF algorithm to find the minimum number of cells for nondetour routing.

Proof: The MCMF algorithm obtains flows with minimum edge costs. Since we encourage a droplet to use a global cell that has been used by other droplets and two to-be-routed

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

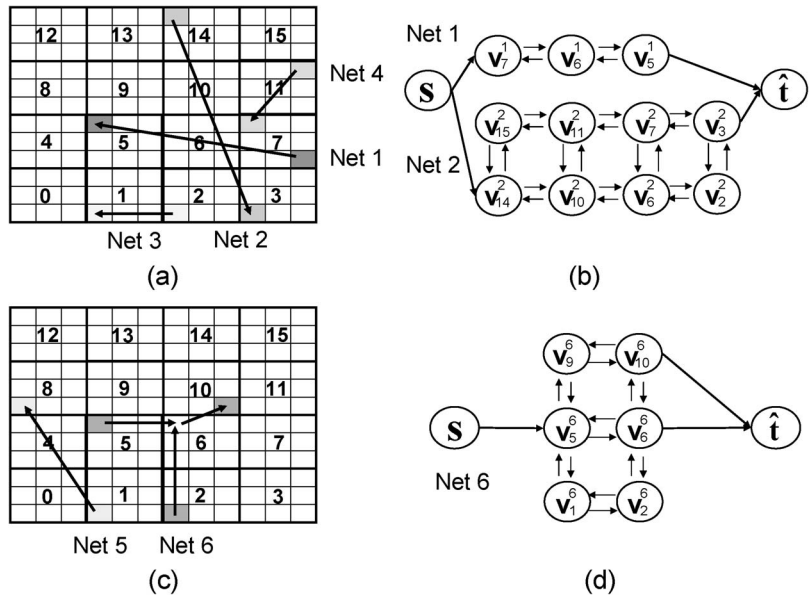


Fig. 9. Example of the MCMF formulation of the 3-D module placement shown in Fig. 3(a) before node split. (a) The 2-D plane at time $t1$. The whole chip is divided into 16 global cells. (b) The network flow formulation for two nets, n_1 and n_2. (c) The 2-D plane at time $t2$. (d) The network flow formulation to route the second droplet of the three-pin net n_6.

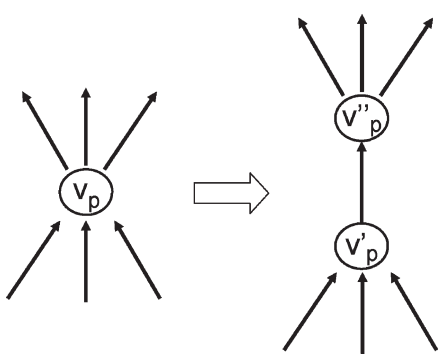


Fig. 10. Node split in global routing.

droplets to use the same global cell, the MCMF algorithm obtains the global routing paths with the minimum number of global cells used for routing, if there are no routing detours. Also, since the size of N' is less than the minimum capacity of all nodes in the flow graph, the flows on the flow graph are the exact global routing paths of all droplets. Therefore, our MCMF algorithm can find the minimum number of cells for nondetour routing. ■

Handling three-pin nets: We decompose a three-pin net n_a into two two-pin nets. We first route the droplet d_1^a with a longer Manhattan distance between its source and sink and then route the other droplet d_2^a. The main idea is to route d_2^a to one of the global cells that are used by d_1^a so that they can be mixed during transportation. We say a node v_p^a is a *mixing node* if at least one cell c in the global cell $\hat{c}_p$ can be used to mix d_1^a and d_2^a, i.e., the two idle intervals di_{a1}^c and di_{a2}^c are overlapped. We refer to the mixing nodes used by d_1^a for routing as the *common mixing nodes*. When routing d_2^a, our goal is to route d_2^a to one of the common mixing nodes. When routing d_2^a, we first remove all edges connecting to $\hat{t}_f$ and connected from s_f. Next, for all droplets d_2^a, we create edges from its common mixing nodes to $\hat{t}_f$ with capacity one and cost zero. Similarly, we create an edge from s_f to v_p^a if the source pin of d_2^a is in $\hat{c}_p$ with capacity one and cost zero. Fig. 9(c) shows the 2-D plane at time $t2$ of the 3-D module placement shown in Fig. 3(a). Fig. 9(d) shows the flow network when we route the second droplet, from dilute a to mix b, of net n_6. Note that the source node is connected to node v_5, and the target node is now connected from nodes v_6 and v_{10}, since these two nodes are common mixing nodes.

Note that, to mix d_1^a and d_2^a, the number of common mixing nodes cannot be zero. When routing d_1^a, we additionally add the mix cost $M_g(v_p^a, d_1^a)$ for each v_p^a to ensure that d_1^a visits at least one mixing node. $M_g(v_p^a, d_1^a)$ is one if v_p^a is not a mixing node; otherwise, $M_g(v_p^a, d_1^a)$ is zero. With the mix cost, the MCMF algorithm will route d_1^a to one of its mixing nodes, and therefore, the number of the common mixing node is not zero when we route d_2^a.

3) Time Complexity Analysis: The global routing problem consists of the net selection and the routing of all droplets by the MCMF algorithm. Given $|N|$ nets, the heuristic algorithm for solving the MWIS problem takes $O(|N|^3)$ time. Therefore, in the worst case, the net selection process can be solved in $O(|N|^4)$ time, where exactly one net is selected for routing at one iteration. The MCMF algorithm can be solved in $O(|V_g||E_g| \log U \log(|V_g|\phi_{\max}))$ time, where U is the largest edge capacity and $\phi_{\max}$ is the largest cost. The size of V_g and E_g are both $O(W_c H_c)$. Note that, in our formulation, multiple routing nodes correspond to the same global cell. Hence, the number of nodes and edges in a network flow graph is proportional to the number of to-be-routed droplets. In the worst case, all nets are three-pin nets, and each time, we determine the routing path of one net and route the remaining nets. Therefore, we need $O(|N|^2)$ times in total. Based on the previous discussion, the global routing problem can be solved in $O(|N|^4 + |N|^4 (W_c H_c)^2 \log U \log(|N|(W_c H_c)\phi_{\max}))$ time.

Theorem 2: Given a set N of nets and a biochip of the width (height) W_c (H_c), the global routing problem can be solved in

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

TABLE II
NOTATIONS USED IN DETAILED ROUTING

$arr(v_p, d^a)/dep(v_p, d^a)$	arrival and departure times of droplet d^a on cell c_p
$\phi_d(v_p, v_q, d^a, t)$	cost of droplet d^a from node v_q to v_p at time t
$U_d(v_p)$	usage cost of cell c_p
$U_d^t(v_p)$	timing cost for timing-aware routing
$F(v_p, v_q, t)$	fluidic penalty for the fluidic constraints
$H_f(v_p, t)$	historic fluidic penalty
$H_f^c(v_p, t)$	fluidic penalty related to current routing iteration
$H_f^p(v_p, t)$	historic fluidic penalty related to previous routing iterations
$M_d(v_p, d_1^a, t)$	mix cost used to mix two droplets of a 3-pin net
$\hat{D}(v_p, d_1^a)$	distance cost
R_a	routing tree of net n_a
$s(v_a, t)$	number of droplets that use v_p for routing at time t

$O(|N|^4 + |N|^4(W_cH_c)^2 \log U \log(|N|(W_cH_c)\phi_{\max}))$ time, where U is the largest edge capacity and $\phi_{\max}$ is the largest cost.

D. Detailed Routing

In this section, we present the proposed detailed routing scheme. The negotiation-based detailed routing algorithm is inspired by [17]. The proposed routing algorithm iteratively routes and schedules each droplet in the decreasing order of their criticality. To schedule each droplet, we determine the arrival and departure times of each droplet on each cell. We also perform rip-up and reroute on failed nets. A failed net is a net that cannot find a routing solution satisfying the fluidic constraints or a three-pin net whose two droplets cannot be mixed during their transportation. The proposed routing algorithm terminates if a feasible routing solution is found or a specified maximum number of iterations is reached. We first present how to route two-pin nets. Then, we detail how to handle three-pin nets. Finally, we present the time complexity analysis. Table II lists the notations used in detailed routing.

1) Routing Graph Construction: We construct a directed routing graph $G_d = (V_d, E_d)$, where V_d is the set of all routing nodes and E_d represents the set of edges. Unlike global routing, we create a unique node v_p for each cell c_p. Two nodes v_p and v_q are connected via a directed edge if droplets can move from c_p to c_q. Each node v_p is associated with two variables, arr(v_p, d^a) and dep(v_p, d^a), to denote the arrival and departure times of the droplet d^a on v_p, respectively. v_p is also associated with its cost $\phi_d(v_p, v_q, d^a, t)$ to represent the cost when a droplet d^a moves from v_q to v_p at time t. The goal of detailed routing is to find the minimum cost routing tree R_a for each droplet embedded in G_d and to determine the arrival and departure times of each node in R_a, provided that the timing and fluidic constraints are both satisfied. A routing tree's cost is the sum of the cost of all tree nodes.

2) Cost of Routing Nodes: The cost $\phi_d(v_p, v_q, d^a, t)$ when droplet d^a moves from v_q to v_p at time t is defined by the following:

$$\phi_d(v_p, v_q, d^a, t) = U_d(v_p) + F(v_p, v_q,) \quad (4)$$

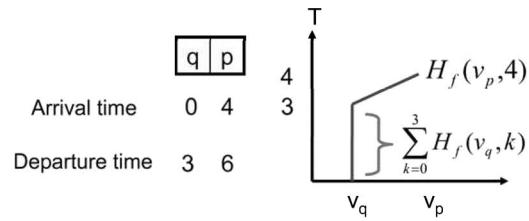


Fig. 11. Illustration of the fluidic penalty.

where $U_d(v_p)$ is the usage cost of v_p and $F(v_p, v_q, t)$ is the fluidic penalty for the fluidic constraints. $U_d(v_p)$ is zero if c_p is used by at least one droplet; otherwise, $U_d(v_p)$ is one. The fluidic penalty $F(v_p, v_q, t)$ is used to guide the detailed router to satisfy the fluidic constraints and is defined as follows:

$$F(v_p, v_q, t) = \sum_{t'+1 \le k \le t-1} \frac{s(v_q, k)}{N_f} \times H_f(v_q, k) + \frac{s(v_p, t)}{N_f} \times H_f(v_p, t) \quad (5)$$

where t' is the arrival time of droplet d^a at v_q, $s(v_p, t)$ is the number of droplets that use v_p for routing at time t, and $H_f(v_p, t)$ is the historic fluidic penalty. N_f is used for normalization and is set to 27 in this paper. It will be explained later when we present our detailed routing algorithm. In the aforementioned equation, the first term accounts for the fluidic penalty if d^a stays at v_q from time $t'+1$ to $t-1$, and the second term represents the fluidic penalty when d^a arrives at v_p at time t. Fig. 11 shows the fluidic penalty computation. Fig. 11 shows a simple biochip with two cells v_q and v_p and the arrival/departure times of a droplet for the two cells. Since this droplet stays at cell v_q from time 0 to 3, the fluidic penalty is $\sum_{k=0}^{3} H_f(v_q, k)$. Also, since this droplet moves to v_p at time 4, we need to add $H_f(v_p, 4)$ to the fluidic penalty. The historic fluidic penalty $H_f(v_p, t)$ is defined by the following:

$$H_f(v_p, t) = H_f^c(v_p, t) \times H_f^p(v_p, t) \quad (6)$$

where $H_f^c(v_p, t)$ is the fluidic penalty related to the current iteration and $H_f^p(c_p, t)$ is the fluidic penalty related to previous iterations. We update the value of H_f^c after routing a net to avoid violating the fluidic constraints when routing other nets and the value of H_f^p at the end of one routing iteration. The initial values of $H_f^c(v_p, t)$ and $H_f^p(v_p, t)$ are both one. Both $H_f^c(v_p, t)$ and $H_f^p(v_p, t)$ are increased when $s(v_p, t)$ is larger than one. If the fluidic constraints are violated in many previous iterations, the value of $H_f(v_p, t)$ will be large. Therefore, the detailed router tends not to route a droplet to a cell at the time t with a large historic fluidic penalty.

3) Routing Algorithm: Fig. 12 shows the flow of our detailed routing algorithm. We route all successfully routed nets in global routing in the decreasing order of their criticality value. At first, we erase the routing tree R_a from the previous iteration and add the source s^a of d^a with arr$(s^a, d^a) = 0$ into a priority Q. The cost of s^a is zero. Let v_q be the node with the lowest cost in Q. We remove v_q from Q and examine the fan-out v_p of v_q to evaluate the cost of v_p if c_p is available to d^a and v_p is

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

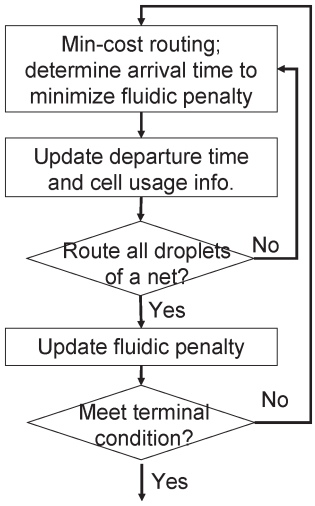


Fig. 12. Detailed routing flow.

used to route d^a in global routing. As described in the previous section, the cost of v_p consists of both usage cost and fluidic penalty which is related to the time when d^a arrives at v_p. We need to determine the arrival time of each droplet on each cell to minimize the fluidic penalty during routing. If there is a tie, we choose the smallest arrival time. We then add v_p into Q with the cost $\phi_d(v_p, v_q, d^a, t) + P_{sq}$, where P_{sq} is the path cost from s^a to v_q. The aforementioned process repeats until d^a reaches its target. Then, we perform back trace to update the departure time and usage cost of each cell and to form the routing tree R_a of d^a. We also update the value of $s(v_p, t)$ if d^a uses cell c_p at time t for routing. Note that, based on the 3-D model presented in Section II-B, if d^a stays at c_p at time t, d^a is considered to use all cells in C'_p from time $t-1$ to $t+1$. Therefore, we update all $s(v_j, k)$ and $H^c_f(v_j, k)$, $c_j \in C'_p, t-1 \le k \le t+1$ after routing d^a. Therefore, the maximum number of droplets that can use c_p at time t for routing is 27. This is the reason why the normalization factor N_f in (5) is set to be 27. If a net is a failed net, we rip up and reroute this net in the next iteration. Note that we do not honor the global routing result after the first iteration. Moreover, if a net n_a fails in global routing, we treat n_a as a failed net and route n_a after the first iteration.

4) Handling Three-Pin Nets: We also handle three-pin nets in detailed routing. Similar to global routing, we first route d^a_1 with a longer Manhattan distance from its source to its sink and then route d^a_2 to a cell that is also used by d^a_1, and the idle intervals of the two droplets in this cell are overlapped. The mixing nodes defined in global routing now refer to the routing nodes v_p if c_p can be used to mix d^a_1 and d^a_2, i.e., the two idle intervals of the two droplets are overlapped in c_p. The common mixing nodes now refer to the mixing nodes used by d^a_1, and the arrival time of d^a_1 is in the idle interval of d^a_2. Since to mix two droplets, these two droplets must be in the same cell at the same time. Our goal is to route d^a_2 to one of its common mixing nodes for mixing.

To handle three-pin nets more efficiently and effectively, we add additional mix and distance costs when routing d^a_1. The mix cost is used to encourage d^a_1 to use a mixing node for routing. The distance cost is used to encourage the detailed router to route d^a_1 as close to d^a_2 as possible. The new routing cost $\phi'_d(v_p, v_q, d^a_1, t)$ to route d^a_1 is defined by the following:

$$\phi'_d\left(v_p, v_q, d^a_1, t\right) = U_d(v_p) + F(v_p, v_q, t) + M_d\left(v_p, d^a_1, t\right) + \hat{D}\left(v_p, d^a_1\right) \quad (7)$$

where $M_d(v_p, d^a_1, t)$ is the mix cost and $\hat{D}(v_p, d^a_1)$ is the distance cost. The mix cost is defined as follows:

$$M_d\left(v_p, d^a_1, t\right) = \begin{cases} 0, & v_p \text{ is a mixing node and } t \in di^i_{a1} \cap di^i_{a2} \\ H_m(v_p, d^a), & \text{otherwise} \end{cases} \quad (8)$$

where $H_m(v_p, d^a_1)$ is the historic mix cost of v_p related to previous iterations. Initially, $H_m(v_p, d^a_1)$ is zero and is increased by one every time we cannot mix a three-pin net n_a. There are two reasons. First, since three-pin nets are handled in global routing, it is very likely that d^a_1 uses at least one mixing node for routing. Second, our goal is to minimize the number of cells used for routing. Therefore, we first focus on the cell number minimization problem. If these two droplets fail to be mixed, then we increase the cost of nonmixing nodes. By doing so, the detailed router will be forced to use mixing nodes for routing due to the mix cost at next iteration.

We observe that using the mix cost alone may not be efficient and effective for droplet mixing. When two droplets fail to be mixed, we need another iteration and, therefore, more CPU time. Therefore, we also define the distance cost when routing d^a_1. The distance cost of v_p is the sum of the Manhattan distance between cell c_p and the source of d^a_2 and between c_p and the target of d^a_2 divided by $T_{\max}$ and is defined by the following:

$$\hat{D}\left(v_p, d^a_1\right) = \frac{m_d\left(s^a_2, c_p\right) + m_d\left(c_p, \hat{t}^a\right)}{T_{\max}}. \quad (9)$$

With the distance cost, the detailed router will route d^a_1 as close to d^a_2 as possible; as a result, it is more likely to use a mixing node when routing d^a_1.

Note that we do not update the fluidic penalty after routing the first droplet of a three-pin net. The fluidic penalty is referenced when considering the fluidic constraints. Since we intentionally mix two droplets of a three-pin net n_a during their transportation, it could be treated as no fluidic constraints between d^a_1 and d^a_2. Instead, we update the usage cost so that d^a_1 and d^a_2 can use the same cells for routing. After the mixing of d^a_1 and d^a_2, we treat it as one droplet. Finally, Fig. 13 shows the proposed detailed routing algorithm.

5) Time Complexity Analysis: We route at most $O(|N|)$ droplets per iteration. The while loop in line 10 of Fig. 13 iterates at most $O(|E_d|)$ times. By using a priority queue, the time complexity of queue insertion, deletion, and adjustment is $O(\log(|V_d|))$. The size of V_d and E_d are both $O(W_c H_c)$, where W_c (H_c) is the width (height) of a biochip. The violation detection of the fluidic constraints takes $O(|N|^2)$ time. Therefore, the time complexity of detailed routing is $O(|N|^2 + |N|(W_c H_c)\log(W_c H_c))$ if the maximum number of iterations is given as a constant.

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

```
Algorithm: Detailed Routing (N_g)
N_g: set of nets that are successfully routed in global routing
     in the decreasing order of its criticality;
begin
E = N_g; // E is the set of failed nets
iter = 1;
while E is not empty or the maximum iteration limit is reached
  while E is not empty
      Let n_a be the most critical net in E;
      E = E − n_a;
      Rip up the routing tree R_a;
      for all droplets d_k^a, k = 1, 2
          Add s_k^a with arr(s_k^a, d_k^a) = 0 and cost zero into a priority queue Q;
          while (t̂^a is not reached and k = 1) or
                (a common mixing node v_m is not reached and k = 2)
              Remove node v_q with the lowest cost from Q;
              for all fanouts v_p of v_q and c_p is available to d_k^a
                  if iter = 1 and v_p is not used in global routing
                      continue;
                  Choose t such that φ'_d(v_p, v_q, d_k^a, t) (if n_a is a 3-pin net and k = 1)
                  or φ_d(v_p, v_q, d_k^a, t) (otherwise) is minimized;
                  Set arr(v_p, d_k^a) = t;
                  Add v_p to Q with its cost plus P_sq;
          loop nodes v_p from v_l to s_k^a // v_l is the last node
              Update U_d(v_p) and dep(v_p, d_k^a);
      for all v_p used by d_k^a for routing
          Update s(v_j, k) and H_f^c(v_j, k), c_j ∈ C'_p, arr(v_p, d_k^a) − 1 ≤ k ≤ dep(v_p, d_k^a) + 1;
  for all cells c_p
      Update H_f^p(v_p, k), 0 ≤ k ≤ T_max − 1
  Add all failed nets into E;
  iter = iter + 1;
  if iter = 2
      Add net n_b that fails in global routing in E;
end
```

Fig. 13. Summary of the detailed routing algorithm.

Theorem 3: Given a set of N nets and a biochip of the width (height) W_c (H_c), the time complexity of the detailed routing algorithm is $O(|N|^2 + |N|(W_c H_c)\log(W_c H_c))$ if the maximum number of iterations is given as a constant.

E. Timing-Aware Routing

In this section, we detail how to extend the aforementioned algorithm for the timing-aware routing problem. The goal of timing-aware droplet routing is to minimize the maximum transportation time for higher reliability and faster bioassay execution. To achieve this goal, we replace the usage cost with the timing cost in both global and detailed routing. In this way, we could minimize the droplet transportation time while minimizing the routing cost.

To minimize droplet transportation time, different from the optimization objective of minimizing the cells used for routing, a droplet needs to use as few numbers of global cells for routing as possible. To minimize the cells used for routing, we would like to route a droplet through the cells that have already been used by other droplets as much as possible. Even with the objective of shortening routing time, we need to use as few numbers of cells as possible to route a droplet, no matter if these cells are used by other droplets or not. Therefore, the node cost of all nodes in the flow graph is one plus the mix cost when routing a three-pin net. Note that the MCMF algorithm can also find the minimum droplet transportation time for a set of independent nets for nondetour routing. With the timing cost, the MCMF algorithm finds the flow that visits the minimum number of nodes. This flow corresponds to a shortest path and, hence, the shortest time to route a droplet from its source to its sink.

In detailed routing, the usage cost $U_d(v_p)$ in (4) is replaced with the timing cost $U_d^t(v_p)$. $U_d^t(v_p)$ is defined by the following:

$$U_d^t(v_p) = \alpha \times \frac{\text{arr}\,(v_p, d_k^a)}{T_l} \tag{10}$$

where T_l is used for normalization and α is a user-specified constant. In this paper, we set T_l as the maximum bounding box of all nets and $\alpha = 10$. With the timing cost, the detailed router will minimize the time that a droplet arrives at its sink. Therefore, the maximum droplet transportation time is minimized.

IV. Experimental Results

Our algorithm was implemented in the C++ language and ran on a 1.2-GHz SUN Blade-2000 machine with 8-GB memory. For the MCMF algorithm, we used the LEDA package [18]. We also implemented the two-stage routing algorithm [11] and the prioritized A*-search algorithm [9] on the same machine. For both our algorithm and the two-stage algorithm, the maximum number of iterations of routing one 2-D plane is 30. The prioritized A*-search algorithm originally targets at minimizing the droplet transportation time. For fair comparison, we performed

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

four modifications on the prioritized A*-search algorithm. First, we used the criticality information calculated in Section III-B as the priority. Second, for a three-pin net, we simultaneously routed the two droplets to ensure that these two droplets will be mixed during their transportation and to obtain the optimal solution. Third, to minimize the number of cells used when routing a net n_a, the total cost of a node in the search graph is the sum of the number of cells that are only visited by these droplets d_j^a, $j = 1, 2$, and the half perimeter of the bounding box defined by the coordinates of d_j^a and the target pin. Finally, to satisfy the timing constraint, we also used the idle interval defined in Section II-C to restrict the possible droplet routing paths and schedules. We also modified the basic integer linear programming (ILP) formulation proposed in [19] to find the optimal solution. The major constraints include the fluidic constraints and the droplet movement constraint that constrains a droplet to move only to one of its four adjacent cells or to stay at where it is at the next time step. We also handled three-pin nets in the basic ILP formulation. The GNU Linear Programming Kit [20] is used as our ILP solver. For the details of the basic ILP formulation, please refer to [19].

We evaluated our routing algorithm on two bioassays: the *in vitro* diagnostics [11] and the colorimetric protein assay [6]. The *in vitro* diagnostics involves the measurement of glucose and lactate in human physiological fluids, which is very important in the clinical diagnostics of metabolic disorder. A colorimetric enzyme-based method (Trinder's reaction) is used to measure the concentrations of both glucose and lactate. The diagnostics consists of three steps: the dispensing step to generate droplets containing samples/reagents, the mixing step for biological reactions, and the detection step for the detection of the reaction result by an optical detector [21]. The protocol of protein assay is to first dilute samples containing protein with buffers such as 1-M NaOH solution. Then, reagents are mixed with samples for reaction. Finally, an optical detector (e.g., an LED-photodiode setup) is used to detect protein concentration. The readers can refer to [22] for more details.

The diagnostics_1 is the benchmark used in [11]. To compare the routability of each router on harder routing cases, we used the placer proposed in [7] to place the two bioassays with the same design specification specified in [11] and [6] for the *in vitro* diagnostic and the protein assay, respectively. The new benchmarks are diagnostics_2, protein_1, and protein_2. We also performed pin assignment. Table III shows the statistics of each benchmark. Column 2 shows the chip dimension. Column 3 lists the total number of 2-D planes. Column 4 lists the total number of nets of all 2-D planes, and column 5 shows the timing constraint. For all benchmarks, we followed [11] to assume that the electrodes are controlled by a 100-Hz clock, and the maximum delay constraint is 0.2 s. Therefore, one time unit in routing is 10 ms, and the timing constraint is 20 time units.

For the comparative studies, we first observed that the basic ILP formulation needs at least five days to route a benchmark. This result reveals that the ILP formulation is not practical for the droplet routing problem. Therefore, we omit the comparison with the basic ILP formulation. Instead, we compared our proposed routing algorithms with the two-stage and prioritized A*-search algorithms in terms of solution quality and CPU time. Table IV shows the experimental results. We report the total number of cells used for routing on all 2-D planes (#Tcell) and the CPU time to route all 2-D planes. As shown in this table, our routing algorithm can route all benchmarks while previous works cannot. For example, for the diagnostics_2 benchmark, neither of the two-stage routing and the prioritized A*-search algorithms is able to generate a routing solution, while our schemes have successfully routed this benchmark with reasonable CPU time. Furthermore, for those benchmarks where previous approaches can generate a feasible solution, e.g., diagnostics_1, our routing algorithm provides solutions with fewer cells used for routing in less CPU time compared with the two-stage routing algorithm and the prioritized A*-search algorithm. This result demonstrates the robustness and efficiency of our routing algorithm. Fig. 14 shows the routing result of one 2-D plane of the diagnostics_1 benchmark. This 2-D plane has 5 nets and 45 cells used for routing. Arrows represent the droplets' moving directions.

Here, we discuss why our approach is more robust and effective than the other two routing algorithms. For the two-stage algorithm, droplet routing and scheduling are performed in separate stages without considering the interaction among them. Moreover, the alternative routing path generation stage only finds the shortest path without explicitly minimizing the number of cells used for routing. In contrast, our algorithm can

TABLE III
STATISTICS OF THE ROUTING BENCHMARKS

Circuit	Chip dimension	#2D planes	#Tnets	T_{max}
Diagnostics_1	16 × 16	11	28	20
Diagnostics_2	14 × 14	15	35	20
Protein_1	21 × 21	64	181	20
Protein_2	13 × 13	78	178	20

TABLE IV
ROUTING RESULT OF THE TWO REAL-WORLD BIOASSAYS. N/A DENOTES THAT SOME 2-D PLANES ARE FAILED FOR ROUTING

Circuit	[11]		[9]		Ours	
	#Tcells	CPU time (sec)	#Tcells	CPU time (sec)	#Tcells	CPU time (sec)
Diagnostics_1	267	0.15	269	3.36	237	0.05
Diagnostics_2	N/A	N/A	N/A	N/A	236	0.04
Protein_1	1735	1.33	N/A	N/A	1618	0.22
Protein_2	N/A	N/A	N/A	N/A	939	0.12

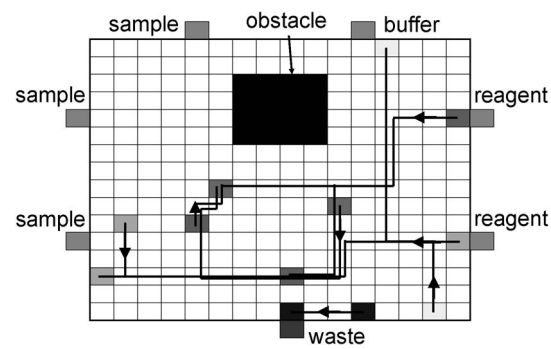


Fig. 14. Routing result of the 2-D plane of the diagnostics_1 benchmark obtained. There are a total of five nets in this 2-D plane. The arrows represent droplet movement directions.

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

TABLE V
TIMING-AWARE ROUTING RESULT OF THE TWO REAL-WORLD BIOASSAYS. N/A DENOTES THAT SOME 2-D PLANES ARE FAILED FOR ROUTING

Circuit	[11]		[9]		Ours	
	R_d^t	CPU time (sec)	R_d^t	CPU time (sec)	R_d^t	CPU time (sec)
Diagnostics_1	2.22	0.17	1.17	45.26	1.16	0.05
Diagnostics_2	N/A	N/A	N/A	N/A	1.33	0.05
Protein_1	2.85	1.35	N/A	N/A	1.44	0.24
Protein_2	N/A	N/A	N/A	N/A	1.12	0.17

concurrently route a set of independent nets in global routing and simultaneously perform droplet routing and scheduling in detailed routing. For the prioritized A*-search algorithm, the possible routing path and schedules of low-priority droplets are not considered while routing high-priority droplets. Therefore, droplets with lower priorities may be blocked by droplets with higher priorities, making this approach harder to find a feasible solution. In contrast, our algorithm adopts a negotiation-based routing scheme. We iteratively rip up and reroute a set of nets to modify the routing solution. Therefore, our algorithm is more robust for various bioassays.

Next, we show the timing-aware droplet routing results in Table V. Note that, in this experiment, for fair comparison, we adopted the original prioritized A*-search algorithm without any modifications. Let R_d^t be the ratio of the maximum droplet transportation time (in cycles) over the maximum Manhattan distance of all nets in one 2-D plane. The maximum Manhattan distance is the minimum time to route all droplets from their sources to sinks. Therefore, a smaller R_d^t indicates shorter time to route all droplets. We report the maximum R_d^t of all 2-D planes and CPU time. Compared with the two-stage routing algorithm, our algorithm obtains a better solution, i.e., smaller R_d^t (1.16 versus 2.22), in less CPU time (0.05 versus 0.17 s) for the diagnostics_1 benchmark. The experimental results also show that our routing algorithm outperforms the prioritized A*-search algorithm. For the same benchmark, our algorithm obtains a routing solution with shorter routing time (1.16 versus 1.17) in much less CPU time (0.05 versus 45.26 s).

V. CONCLUSION

In this paper, we have proposed an efficient and robust routing algorithm for the droplet routing problem on digital microfluidic biochips. We adopted a two-stage routing methodology. In global routing, we proposed the first network-flow-based routing algorithm to optimally route a set of independent nets. In detailed routing, we proposed the first polynomial-time routing algorithm to simultaneously route and schedule all droplets. The proposed routing algorithm can handle two different routing objectives: minimizing the number of cells used for routing or shortening routing time. The experimental results demonstrated the robustness and efficiency of our routing algorithm.

Future work includes the consideration of cross-contamination among different samples while minimizing the number of cells used for routing. The avoidance of cross-contamination is important since, once proteins are absorbed on surface, they may trigger further protein absorption [23]. There are two possible ways to handle this problem. The first one is to update the absorption level of biological samples on each cell after routing a droplet. When the absorption level of a cell exceeds a threshold, this cell is treated as an obstacle. No other droplets can further use this cell for routing, and therefore, the risk of cross-contamination is minimized. The other one is to incorporate the possibility of absorption of biological samples on each cell into the routing cost function. Research along this direction is ongoing.

REFERENCES

[1] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "BioRoute: A network-flow based routing algorithm for digital microfluidic biochips," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2007, pp. 752–757.

[2] F. Su, K. Chakrabarty, and R. B. Fair, "Microfluidics-based biochips: Technology issues, implementation platforms, and design-automation challenges," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 2, pp. 211–223, Feb. 2006.

[3] R. B. Fair, V. Srinivasan, H. Ren, P. Paik, V. Pamula, and M. Pollack, "Electrowetting-based on-chip sample processing for integrated microfluidics," in *IEDM Tech. Dig.*, Dec. 2003, pp. 32.5.1–32.5.4.

[4] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 223–228.

[5] F. Su and K. Chakrabarty, "Design of fault-tolerant and dynamically-reconfigurable microfluidic biochips," in *Proc. Des. Autom. Test Eur.*, Mar. 2005, pp. 1202–1207.

[6] F. Su and K. Chakrabarty, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. Des. Autom. Conf.*, Jun. 2005, pp. 825–830.

[7] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Placement of digital microfluidic biochips using the T-tree formulation," in *Proc. Des. Autom. Conf.*, Jul. 2006, pp. 931–934.

[8] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Placement of defect-tolerant digital microfluidic biochips using the T-tree formulation," *ACM J. Emerging Technol. Comput. Syst.*, vol. 3, no. 3, pp. 1–31, Nov. 2007.

[9] K. F. Böhringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 2, pp. 334–344, Feb. 2006.

[10] E. J. Griffith, S. Akella, and M. K. Goldberg, "Performance characterization of a reconfigurable planar-array digital microfluidic system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 2, pp. 345–357, Feb. 2006.

[11] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proc. Des. Autom. Test Eur.*, Mar. 2006, pp. 323–328.

[12] M. G. Pallock, A. D. Shenderov, and R. B. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab Chip*, vol. 2, no. 2, pp. 96–101, May 2002.

[13] G. N. Somero, "Proteins and temperature," *Annu. Rev. Physiol.*, vol. 57, pp. 43–68, 1995.

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

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[16] R. D. Mohring, *Graphs and Orders: The Role of Graphs in the Theory of Ordered Sets and Its Application*. Amsterdam, The Netherlands: Reidel, 1984.

[17] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. Int. Symp. Field-Programmable Gate Array*, Feb. 1995, pp. 111–117.

[18] K. Mehlhorn and S. Näher, *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[19] P.-H. Yuh, S. S. Sapatnekar, C.-L. Yang, and Y.-W. Chang, "A progressive-ILP based routing algorithm for cross-referencing biochips," in *Proc. Des. Autom. Conf.*, Jun. 2008, pp. 284–289.

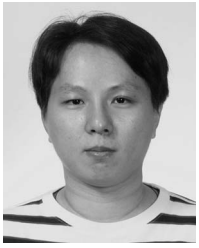
[20] [Online]. Available: http://www.gnu.org/software/glpk/

[21] V. Srinivasan, V. Pamula, and R. Fair, "A droplet-based microfluidic lab-on-a-chip for glucose detection," *Anal. Chim. Acta*, vol. 507, no. 1, pp. 145–150, 2004.

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.

[22] V. Srinivasan, V. Pamula, P. Paik, and R. Fair, "Protein stamping for maldi mass spectrometry using an electrowetting-based microfluidic platform," in *Proc. Int. Soc. Opt. Eng.*, 2004, pp. 26–32.

[23] J.-Y. Yoon and R. L. Garrell, "Preventing biomolecular adsorption in electrowetting-based biofluidic chips," *Anal. Chem.*, vol. 75, no. 19, pp. 5097–5102, Oct. 2003.



Ping-Hung Yuh received the B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2002 and the Ph.D. degree in computer science from National Taiwan University (NTU), Taipei, Taiwan, in 2008.

He was a Visiting Scholar at the University of Minnesota, Minneapolis, in 2007–2008. He is currently with the Department of Computer Science and Information Engineering, NTU. His current research interests include temporal floorplanning for reconfigurable computing and biochip design automation, including placement and routing.

Chia-Lin Yang (M'02) received the B.S. degree from the National Taiwan Normal University, Taipei, Taiwan, in 1989, the M.S. degree from the University of Texas, Austin, in 1992, and the Ph.D. degree from the Department of Computer Science, Duke University, Durham, NC, in 2001.

In 1993, she was with VLSI Technology Inc. (now Philips Semiconductors) as a Software Engineer. She is currently an Associate Professor with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei. Her research interests include energy-efficient microarchitectures, memory hierarchy design, and multimedia workload characterization.

Dr. Yang is the recipient of a 2000–2001 Intel Foundation Graduate Fellowship Award and the 2005 IBM Faculty Award.

Yao-Wen Chang (S'94–A'96–M'96) received the B.S. degree in computer science from National Taiwan University (NTU), Taipei, Taiwan, in 1988 and the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, in 1993 and 1996, respectively.

He is a Professor with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, NTU. He is currently also a Visiting Professor at Waseda University, Kitakyushu, Japan. He was with National Chiao Tung University (NCTU), Hsinchu, Taiwan, from 1996 to 2001 and with IBM T. J. Watson Research Center, Yorktown Heights, NY, in the summer of 1994. He has been working closely with industry in these areas. He has coedited one textbook on electronic design automation and coauthored one book on routing and over 140 Association for Computing Machinery (ACM)/IEEE conference/journal papers in these areas. He is the Editor of the *Journal of Information Science and Engineering*. His current research interests lie in very large scale integration physical design, design for manufacturability/reliability, and design automation for biochips.

Dr. Chang was the recipient of the 2008 ACM International Symposium on Physical Design (ISPD) Global Routing Contest and the 2006 ACM ISPD Placement Contest. He was a recipient of the Best Paper Award at the International Conference on Computer Design (ICCD)-95 and 12 Best Paper Award Nominations from the Design Automation Conference (DAC) (four times), the International Conference on Computer-Aided Design (ICCAD) (twice), ISPD (three times), ACM Transactions on Design Automation of Electronic Systems, Asia and South Pacific Design Automation Conference and ICCD in the past eight years. He has received many research awards, such as the 2007 Outstanding Research Award, the inaugural 2005 First-Class Principal Investigator Award, and the 2004 Dr. Wu Ta You Memorial Award, all from National Science Council of Taiwan, and the 2004 MXIC Young Chair Professorship from the MXIC Corporation, and excellent teaching awards from NTU (four times) and NCTU. He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He has served on the ICCAD Executive Committee, the ACM/Special Interest Group on Design Automation (SIGDA) Physical Design Technical Committee, the ACM ISPD and IEEE International Conference on Field-Programmable Technology Organizing Committees, and the Technical Program Committees of ASP-DAC, DAC, Design Automation and Test in Europe, International Conference on Field Programmable Logic and Applications, FPT, Great Lakes Symposium on Very Large Scale Integration (VLSI), ICCAD, ICCD, Industrial Electronics Conference, ISPD, SoC Conference, IEEE Region 10 Conference, and VLSI-DAT. He is currently an independent board director of Genesys Logic, Inc., a member of the board of governors of Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.

Authorized licensed use limited to: IEEE Xplore. Downloaded on May 1, 2009 at 07:08 from IEEE Xplore. Restrictions apply.