A High-Performance Droplet Routing Algorithm for Digital Microfluidic Biochips

Minsik Cho and David Z. Pan, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a high-performance droplet router for a digital microfluidic biochip (DMFB) design. Due to recent advancements in the biomicroelectromechanical system and its various applications to clinical, environmental, and military operations, the design complexity and the scale of a DMFB are expected to explode in the near future, thus requiring strong support from CAD as in conventional VLSI design. Among the multiple design stages of a DMFB, droplet routing, which schedules the movement of each droplet in a time-multiplexed manner, is one of the most critical design challenges due to high complexity as well as large impacts on performance. Our algorithm first routes a droplet with higher bypassibility which is less likely to block the movement of the others. When multiple droplets form a deadlock, our algorithm resolves it by backing off some droplets for concession. The final compaction step further enhances timing as well as fault tolerance by tuning each droplet movement greedily. The experimental results on hard benchmarks show that our algorithm achieves over 35× and 20× better routability with comparable timing and fault tolerance than the popular prioritized A* search and the state-of-the-art network-flow-based algorithm, respectively.**

***Index Terms*—Biochip, bypassibility, droplet, microfluidics, routing, synthesis.**

I. Introduction

Since 1988, nearly 30 years after Dr. Feynman's celebrated 1959 lecture on future nanotechnology (presented to the American Physical Society) [3], microelectromechanical system (MEMS) has significantly advanced from the early stage of microfabrication/device research to the mature stage of mass production for commercial applications and, now, further opens up a new era for exploring research and applications such as RF/optical communications, microenergy fuel cells, or clinical/biochemical instruments [4]. Among them, bio-MEMS for clinical or biochemical purposes holds great promise due to its cost effectiveness, portability, yet critical applications. For example, a biochip based on bio-MEMS technology becomes popular in analysis of DNA/protein for clinical/medical diagnosis, detection of toxins/pathogens/terror for military/environmental safety, manipulation of biological samples for laboratory experiments, and so on [5], [6]. Moreover, all these critical tasks can be performed in a small space efficiently without involving any human experimenter or expensive equipment due to automated operations at low cost.

One of the most advanced technologies to build a biochip is based on microfluidics where micro/nanoliter droplets are controlled or manipulated to perform intended biochemical operations on a *miniaturized laboratory*, so-called lab-on-a-chip [7]. The old generation of microfluidic biochip consists of several micrometer-scale components including channels, valves, actuators, sensors, pumps, and so on. Even though this generation shows successful applications like DNA probing, it is unsuitable to build a large and complex biochip because it uses *continuous* liquid flows, like continuous voltages in analog VLSI designs (see Section II-A for more details). The new generation of microfluidic biochip has been proposed based on a recent technology breakthrough where the continuous liquid flow is sliced or *digitized* into droplets. Such droplets are manipulated independently by electric signals. This new generation is referred to as a digital microfluidic biochip (DMFB).

Due to such a digital nature of a DMFB, any operation on droplets can be accomplished with a set of library operations like VLSI standard library, controlling a droplet by applying a sequence of preprogrammed electric signals [8]. Therefore, a hierarchical cell-based design methodology can be applied to a DMFB. Under this circumstance, we can easily envision that a large-scale complex DMFB can be designed as done in VLSI, and the market will greatly demand such a DMFB due to economical/portable efficiency as well as safety/health-critical applications. Hence, it is expected that DMFB design needs CAD support as strongly as VLSI design does shortly.

However, CAD research for DMFB design has started very recently. In [9], the first top-down methodology for a DMFB is proposed, which mainly consists of architecture- and geometry-level syntheses. Operation scheduling and resource binding are performed to minimize the maximum chip response time in architecture-level synthesis (i.e., high-level synthesis in VLSI design), while resources are physically placed as modules, and operations are connected by moving droplets in geometry-level synthesis (i.e., physical synthesis in VLSI design). In detail, geometry-level synthesis can be further divided into module placement and droplet routing. During module placement, the location and time interval of each module are determined to minimize area or chip response time. Since different modules can be on the same spot during different time intervals based on reconfigurability (see Section II-A), module placement is

Manuscript received December 26, 2007; revised April 25, 2008. Current version published September 19, 2008. This paper was recommended by Associate Editor K. Chakrabarty.

M. Cho was with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA. He is now with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: thyeros@cerc.utexas.edu).

D. Z. Pan is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA (e-mail: dpan@ece.utexas.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2008.2003282

equivalent to a 3-D packing problem [10], [11]. Meanwhile, in droplet routing, the path of each droplet is found to transport it without any unexpected mixture under design requirements. Similarly to module placement, a spot can be used to transport different droplets during different time intervals (simply in a time-multiplexed manner), which increases the complexity of routing. The most critical goal of droplet routing is routability as in VLSI [1], while satisfying timing constraint and maximizing fault tolerance. More discussion on prior papers to achieve this goal is in Section II-B.

In this paper, we propose a high-performance droplet router for a DMFB. Our approach is mainly based on two ideas, *bypassibility* and *concession*. Bypassibility analysis quantifies how easy it is for unrouted droplets to bypass blockages introduced by a routed droplet (the easier to bypass, the higher bypassibility is). Therefore, we repeat routing one with higher bypassibility to maximize the number of droplets routed, which eventually leaves only the hard-to-route droplets under a deadlock situation. Then we break the deadlock by concession which backs off some droplets to allow the others to pass by. These two ideas provide higher quality solutions than that in [1] and [2]. The major contributions of this paper include the following.

1) We propose a simple yet effective metric bypassibility to estimate the degradation of routability after a droplet is routed. This maximizes the number of routed droplets and narrows down the problem size until multiple droplets under a deadlock are identified.
2) We introduce the concept of a concession zone where some droplet may migrate to break a deadlock between droplets. We route earlier a droplet with longer distance to any of concession zones, as it is harder to be routed in a later stage of routing.
3) We propose 2-D routing for the droplet chosen by bypassibility analysis to reduce runtime. If only one droplet chosen by bypassibility is routed while the others are frozen, this can be solved in a compact 2-D plane rather than in a huge 3-D plane where the third axis represents time.

The rest of this paper is organized as follows. Section II presents preliminaries. In particular, routing problems in a DMFB and a VLSI circuit are compared in Section II-B to help readers with VLSI background. The droplet routing in a DMFB is defined in Section III, and Section IV presents our proposed algorithm for DMFB routing. Experimental results are discussed in Section V, followed by the conclusion in Section VI.

II. Preliminaries

A. Digital Microfluidic Biochips

The first generation of biochips is based on a continuous-flow system where liquid flows through microfabricated channels continuously using electrokinetic-based microactuators. Although a continuous-flow biochip is widely used for simple yet well-defined biochemical operations like DNA probing, it is inherently unsuitable for large-scale complex biochip design due to the following reasons: 1) Permanently microfabricated channels limit the reconfigurability for both applications and fault tolerance, and 2) inevitable shear flow around microactuators and diffusion on channels increase the possibility of sample contamination [10].

To overcome the aforementioned drawbacks, a DMFB is devised where liquid is discretized or *digitized* into independently controllable droplets ($\ll 1\ \mu l$), and each droplet is moved or manipulated on a substrate according to a preprogrammed schedule. Such digitization and programmability enable one to design a large-scale and complex DMFB by allowing a hierarchical and cell-based design methodology as in modern VLSI design. They also provide reconfigurability for various biochemical applications with enhanced fault tolerance.

Although multiple technologies to control droplets, such as chemical [13], [14] or thermal [15] methods, have been proposed, electrical methods such as dielectrophoresis (DEP) [16] and electrowetting-on-dielectric (EWOD) [8], [17] have received more attention due to their high accuracy. Both techniques leverage electrohydrodynamics for faster droplet movement, but DEP suffers from excessive Joule heating [16]. In this paper, we mainly consider an EWOD-based DMFB, but the proposed algorithm itself is generic enough for any type of technology.

Fig. 1 shows the schematic view of an EWOD-based DMFB and an example of its 3-D placement. As shown in Fig. 1(a), a unit cell consists of two parallel glass plates which sandwich biochemical droplets. While the top glass plate has a ground electrode only, the bottom has a regularly patterned array of individually controllable electrodes. The EWOD effect to drive the droplet occurs when control voltage is applied to the controllable electrode. Therefore, by controlling voltage to each electrode in the bottom glass plate with VLSI circuitries, we can have fine control over droplet movement. In [6], four essential operations for DMFB, namely, creating, transporting, cutting, and merging droplets, are demonstrated by applying control voltages to the bottom electrodes. Fig. 1(b) shows the overview of a DMFB. Due to individual controllability of each electrode (thus, each droplet), we can manipulate multiple droplets simultaneously and move them parallel to anywhere in the chip to perform preprogrammed biochemical operations. Therefore, any operation on droplets can happen anywhere in the chip, which provides the reconfigurability of a DMFB. For example, when multiple droplets perform operations like mixing, they need some real estate of the chip for fixed amount of time. After the operation time elapses, these droplets can go to somewhere else for their next scheduled operations, after releasing the taken area for the other droplets to perform different operations such as diluting. This requires 3-D placement of operations, as shown in Fig. 1(c), where each 3-D box indicates biochemical operation.

This reconfigurability raises two important physical design challenges: 1) where and when to perform which biochemical operations, and 2) how to move droplets avoiding undesired mixtures and blockages. The first problem is DMFB placement

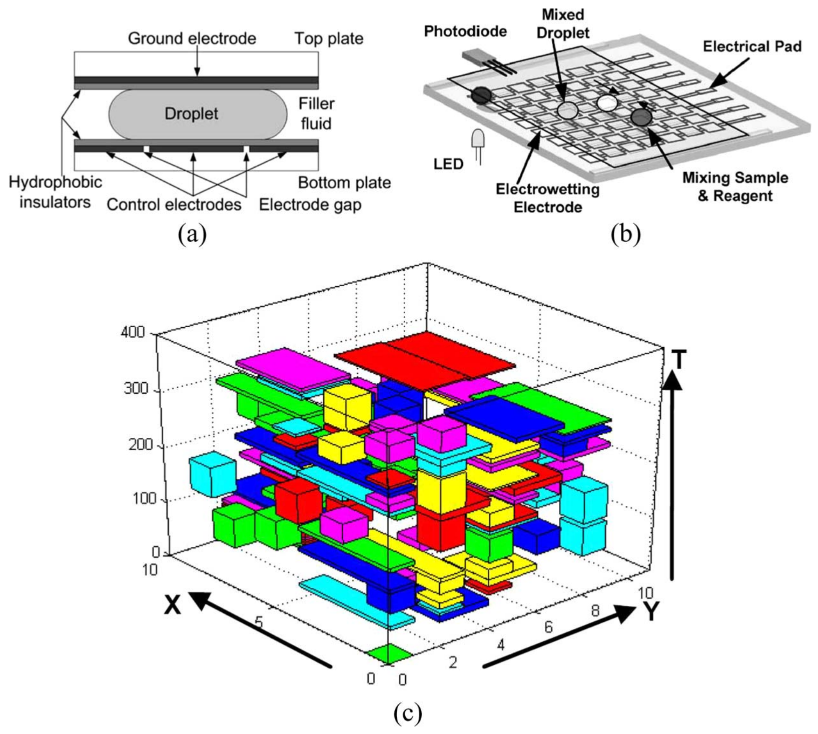


Fig. 1. Schematic view of DMFBs for colorimetric assays [1]. (a) EWOD-based basic unit cell. (b) Top view of microfluidic array. (c) 3-D placement of operations for DMFBs [12].

which is essentially 3-D packing [11], [18], and the second problem is droplet routing [1], [12], [19] which will be further discussed in Section II-B.

B. Routing for DMFB

The goal of droplet routing in a DMFB is to find an efficient schedule for each droplet from its source to target while satisfying design constraints. This sounds similar to VLSI routing where wires need to be connected under design rules, but the reconfigurability of a DMFB makes fundamental differences from VLSI routing in the following aspects.

1) DMFB routing allows multiple droplets to share the same spot during different time intervals [1], [2], [19] like time division multiplexing, while VLSI routing makes one single wire permanently and exclusively occupy the routing area.
2) DMFB routing allows a droplet to stall/stand by at a spot, if needed. For example, when a droplet has to pass busy/congested regions, stalling can be more effective than detouring.
3) VLSI routing requires 2-D spacing by design rules, but DMFB routing needs 3-D spacing by dynamic/static fluidic constraints.
4) In DMFB, there are special spots, called *waste reservoirs*, where all the useless or dreg droplets are discarded/dumped. Hence, differently from VLSI routing, some droplets can dynamically disappear.

A highly equivalent problem to DMFB droplet routing has been extensively studied in robotics as mobile robot motion planning and solved by prioritized A^* search [1]. In [20] and [21], the mobile robot motion planning is shown to be NP-hard, and an integer linear programming approach is proposed. Recent research efforts in DMFB design from the VLSI the community attack the problem using various heuristics such as Internet routing protocol (open shortest path first) or pattern selection [19], [22]. However, these approaches suffer from initialization overhead either to build routing tables or to discover a set of feasible routing patterns. Moreover, as a DMFB keeps reconfiguring, this overhead occurs repeatedly, involving large storage overhead. In [2], a novel network-flow-based algorithm with negotiation is proposed for DMFB droplet routing, showing better performance than that in [1] and [19]. However, the network-flow formulation is significantly bottlenecked by the distribution of blockages. To conservatively guarantee the fluidic constraint (see Section III), a channel with at least three unit cells is considered in the network-flow formulation. Hence, if the width of the channel between blockages is less than three unit cells (even though a droplet can use it), the channel will not be utilized in the network-flow formulation, resulting in suboptimal solutions in terms of routability.

Once a routing solution is found during design time or offline, then the solution will be stored in memory logic (e.g., ROM) to activate electrodes accordingly in order to move droplets during runtime or online. How to dynamically change routing paths under dynamic defects and variations is still under heavy research. The amount of parallelism depends on a problem instance or a routing algorithm. For example, if there

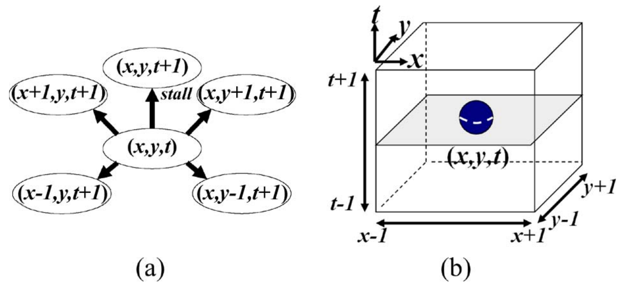


Fig. 2. Graph model and fluidic constraints for DMFB design. (a) Our graph for droplet routing models geometric paths as well as temporal schedules simultaneously. (b) Dynamic and static fluidic constraints are to prevent unexpected mixtures of droplets during movement.

TABLE I
NOTATIONS IN THIS PAPER

d_i	droplet i
S_i	source location of $d_i = (x_i^s, y_i^s)$
T_i	target location of $d_i = (x_i^t, y_i^t)$
AT_i	arrival time of d_i at T_i
R_i^t	shadowed region of d_i at (x_i, y_i) at time t_i $= \{(x,y,t) \mid \lvert x_i^{t_i} - x\rvert \le 1, \lvert y_i^{t_i} - y\rvert \le 1, \lvert t_i - t\rvert \le 1\}$
C_i	a set of cells used to route a d_i

are too many blockages, there will not be large parallelism, as only a few droplets can be transported concurrently.

III. Problem Formulation

In this section, we first show a routing model and constraints, and then propose a problem formulation. Since the problem can be abstracted as transporting each droplet from its source to target, we cast droplet routing into a graph search as done in VLSI routing. As resource sharing in a time-multiplexed fashion is allowed in a DMFB, we can model it as a 3-D graph where z-axis is for time, which enables one to optimize geometric paths and temporal schedules simultaneously. Fig. 2(a) shows the concept of our graph where a droplet at (x, y, t) can move to one of five nodes at $t+1$. This graph is not only directed but also acyclic due to the causality of time multiplexing differently from the graph in VLSI routing [23].

Since all the droplets are moving in parallel, there can be unwanted mixtures if keep-off distance/spacing is not observed. Let d_i at (x_i^t, y_i^t) and d_j at (x_j^t, y_j^t) denote two independent droplets at time t. Then, the following constraints should be satisfied for any t during routing:

1) Static constraint: $|x_i^t - x_j^t| > 1$ or $|y_i^t - y_j^t| > 1$.
2) Dynamic constraint: $|x_i^{t+1} - x_j^t| > 1$ or $|y_i^{t+1} - y_j^t| > 1$ or $|x_i^t - x_j^{t+1}| > 1$ or $|y_i^t - y_j^{t+1}| > 1$.

Dynamic constraint requires that the activated cell for d_i cannot be adjacent to d_j. Otherwise, there can be more than one activated neighboring cell for d_j, which may lead to errant fluidic operations. Such static and dynamic fluidic constraints can be visually illustrated, as shown in Fig. 2(b), where there should not be any other droplets in a cube centered by one droplet. In addition, defective or reserved unit cells can be blockages for routing [10].

Sometimes, droplets may have a required arrival time to prevent spoilage, which becomes a timing constraint. Finally, it is desirable to minimize the number of unit cells that are used at least once by droplets. Since a unit cell of a DMFB can be defective due to manufacturing or environmental issues, using a smaller number of nodes (each node corresponds to one unit cell) can be beneficial for robustness. Considering all the aforementioned constraints, we can define the problem as follows using the notations in Table I.

Let $G = (V, E)$, $D = \{d_1, d_2, \ldots, d_n\}$, **and** RT **denote an acyclic graph model for a** DMFB, **a set of droplets to be routed, and a required arrival time**, **respectively**. **Droplet routing problem is to transport each droplet** $d_i \in D$ **from** S_i **to** T_i **through** G **such that** d_i **is the only one in** R_i^t $(t \ge 0)$ **and** $\mathrm{AT}_i \le \mathrm{RT}$ **while minimizing** $|\bigcup_{i=1,\ldots,n} C_i|$.

As an efficient solution to this NP-hard problem, we propose a strategy inspired by Chaitin's algorithm [23] to solve *k-coloring* [24], [25], where all the nodes in a graph should be colored differently from their connected nodes using k colors. According to [23], they first take off a node with less than k edges from the graph, as it is guaranteed to be colored differently from its neighbors (at most $k-1$ colors will be used for the neighbor nodes). By removing such nodes repeatedly, some node will have less than k edges (which had more than k edges previously), and eventually, the graph is reduced to the level where no node can be removed, which implies that a hard part of the problem is identified. Then, a complex approach can be applied to attack the hard part which is significantly smaller than the original graph. We use bypassibility analysis to reduce the problem size, and concession to solve a hard part of the problem as to be explained in Section IV.

```
Algorithm 1 Overall Algorithm
Require: A set of all droplets $D$, a routing graph $G$, a timing
    constraint RT
Ensure: $D_u \leftarrow D, T_b \leftarrow 0, T_c \leftarrow 0$
 1: repeat
 2:   $T_b$ = Routing-Bypassibility$(D_u, G, max(T_b, T_c))$
 3:   if $T_b$ is not increased then
 4:     $T_c = max($Routing-Concession$(D_u, G, T_b), T_c)$
 5:   end if
 6: until No droplet routed
 7: Routing-Compaction$(D_u, D, G, \mathrm{RT})$
```

IV. Algorithm

In this section, we propose our algorithm for droplet routing in a DMFB. The key ideas behind our approach are as follows.

1) If T_i happens to be in a highly sparse region, it may not be hard for the unrouted droplets to bypass the blockages induced by routing d_i, implying high bypassibility of d_i. This motivates us to route d_i first.
2) In case more than two droplets are in a deadlock, we need to back some droplets off to provide other droplets with free paths. This is done based on the distances to

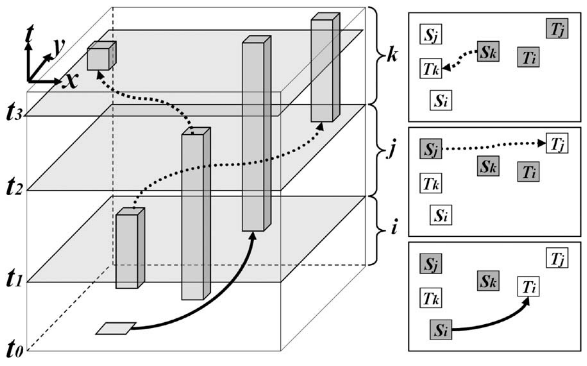


Fig. 3. Each droplet is routed during different time intervals to reduce A* search complexity.

concession zones, which will be explained in Section IV-B in detail.

3) We route each droplet chosen by bypassibility during different time intervals to improve runtime, which effectively converts 3-D routing into 2-D routing. As a result, this approach reduces runtime overhead.

Our overall algorithm is presented in Algorithm 1. First, we repeat picking a routable droplet with the maximum bypassibility and making it routed in line 2, which continuously narrows down the problem size as in Section IV-A. When no droplet can be routed as in line 3, it means that there is a deadlock between droplets and we encounter the hard part of the problem. Hence, we apply an algorithm with concession to resolve the deadlock in line 4, which is in Section IV-B. Then, we continue to route based on bypassibility in line 2. As a final step in line 7, we compact the routing solution greedily to enhance multiple design objectives as in Section IV-C.

The intuition behind our routing algorithm is similar to traffic control, as each droplet can be regarded as a car. If a car is parked in a busy areas it will block traffic and make flow worse, which leads to the bypassibility concept. If two cars drive toward each other on the narrow local load, one car should back off first, which leads to the concession concept.

While routing is based on bypassibility, we move only one droplet while freezing the others, which can be done in a 2-D plane rather than in a 3-D plane. Fig. 3 shows an example of routing three droplets d_i, d_j, and d_k. Until routing d_i is completed (until t_1), d_j and d_k are frozen at S_j and S_k, respectively, and from t_1, T_i becomes a blockage for d_j and d_k. In the same fashion, d_j is routed while d_k is frozen. In this way, we can find a path in a 2-D plane and then map the path to a 3-D plane as shown in Fig. 3. For this, we need to keep track of the last time when a droplet routing is completed such as t_1, t_2, and t_3 in Fig. 3 using T_b and T_c in Algorithm 1.

A. Routing by Bypassibility

Once a droplet d_i is routed (moved to T_i), it stays at T_i, permanently blocking shadowed regions $\{R_i^t | t \geq \mathrm{AT}_i\}$. Therefore, if T_i happens to be in a highly congested region, the unrouted droplets may not find feasible paths to their target locations, particularly in case they have to pass around T_i. For such a case, it is clearly better to route d_i as late as possible.

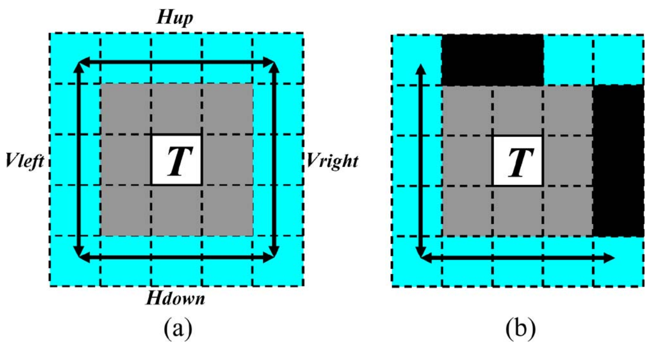


Fig. 4. Bypassibility is based on whether there exist bypasses for the unrouted droplets. (a) 5 × 5 window is considered to evaluate the bypassibility. Four bypasses are shown right out of the shadowed regions. (b) This example has full bypassibility, as there exist at least one vertical and one horizontal bypasses.

TABLE II
BYPASSIBILITY ANALYSIS TABLE

Direction		Ideal	Full									Half						No
H	H_{up}	d[a]	o[b]	x[c]	o	o	o	o	o	x	x	o	x	o	x	x	x	x
	H_{down}	d	o	o	x	o	o	x	x	o	o	o	x	x	o	x	x	x
V	V_{left}	d	o	o	o	x	o	x	o	x	o	x	o	x	x	o	x	x
	V_{right}	d	o	o	o	o	x	o	x	o	x	x	o	x	x	x	o	x

[a] don't-care.
[b] blocked.
[c] unblocked.

In this section, we propose a way to capture the congestion around a target location quantitatively with a concept of bypassibility. The bypassibility of a droplet d_i depends on whether there will be any bypass for the unrouted droplets after d_i is routed. Fig. 4(a) shows four possible bypasses right out of the shadowed region (which is to keep fluidic constraints), namely, H_{up}, H_{down}, V_{left}, and V_{right}, within a 5 × 5 window centered by the target location T. One exceptional case is when T is one of the waste reservoirs where one or more useless droplets can be dumped during operations [6], [8], [17]. Unlike a typical droplet, a droplet transported to a waste reservoir does not create any *new* blockage, thus incurring no impact on overall routability. Then, depending on whether these bypasses are blocked or not, we can divide all the possibilities into the following four classes based on Table II.

1) Ideal bypassibility: This is only when a target is a waste reservoir.
2) Full bypassibility: This allows both horizontal and vertical bypasses.
3) Half bypassibility: This allows only either horizontal or vertical bypass.
4) No bypassibility: This does not allow any bypass.

Note that it is *not* required to have both H_{up} and H_{down} unblocked to have horizontal bypassibility, as either bypass can be shared by multiple droplets in a time-multiplexed manner (also the same for the vertical case). The example in Fig. 4(b) has full bypassibility as Fig. 4(a), in spite of blocked or shadowed regions (H_{up} and V_{right} are blocked), as it still has one vertical and one horizontal bypass. Therefore, if a droplet with ideal or full bypassibility is routed first, it will not affect the overall

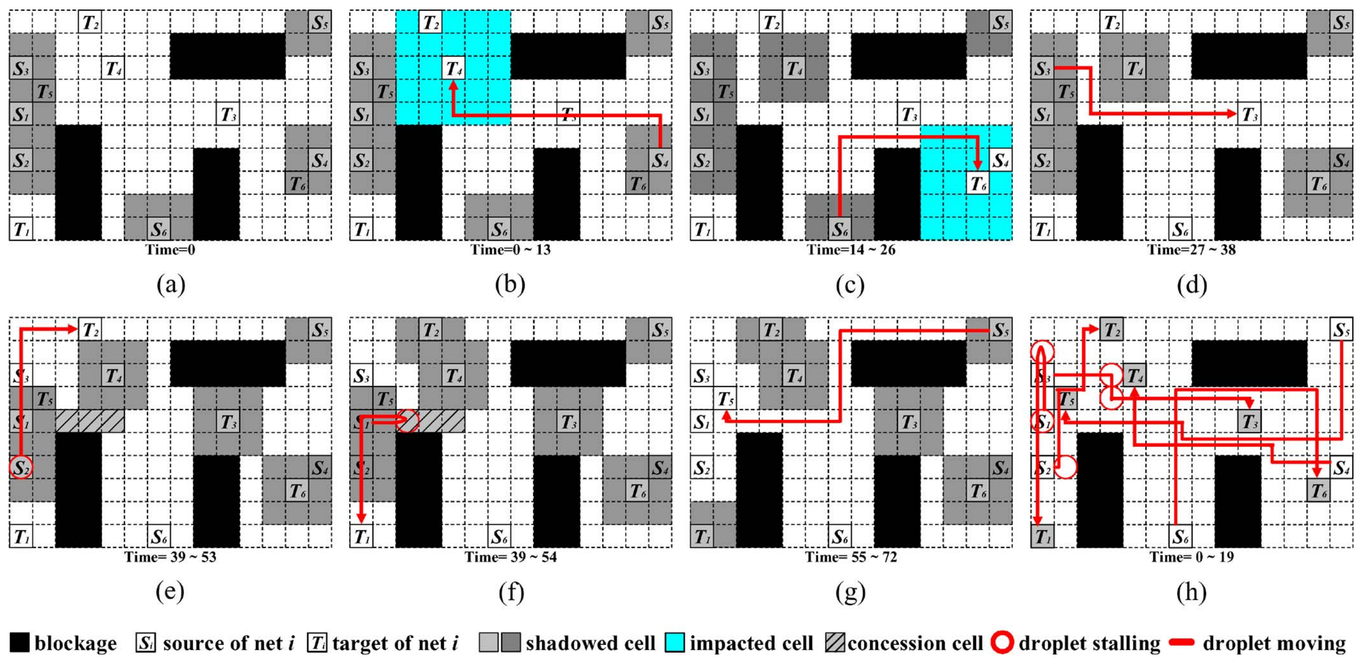


Fig. 5. This example describes the proposed droplet routing algorithm. After the first three routings, (b)–(d) are done by Algorithm 2 (Routing-Bypassibility) then no droplet can be routed in a 2-D plane due to a deadlock between d_1 and d_2. Thus, as in Algorithm 1, (e) and (f) are done in a 3-D plane by Algorithm 3 (Routing-Concession) to resolve the deadlock. After the resolution, (g) is done in 2-D again by Algorithm 2, followed by the compaction in (h) using Algorithm 4. (a) An example routing problem with $d_1 - d_6$ with blockages. (b) d_4 is routed due to full bypassibility. (c) After T_6 is freed up, d_6 has the most bypassibility. (d) d_3 is the only routable one, despite no bypassibility. (e) d_2 is routed due to the longest distance to the concession zone. (f) d_1 migrates to the concession zone first to avoid d_2. (g) d_5 is the only unrouted droplet with half routability. (h) The timing requirement (20) is met after compaction.

chip routability, because the other droplets can bypass vertically or horizontally in a time-multiplexed manner, which leads to Theorems 1 and 2.

Theorem 1: Routing a droplet with ideal bypassibility does neither affect overall chip routability nor increase the Manhattan routing length in a 2-D plane of unrouted droplets.

Proof: Consider two unrouted droplets d_i and d_j, and assume that both are on feasible routing paths P_i^t and P_j^t, respectively, at time t. Furthermore, assume that d_i has ideal bypassibility. Since routing d_i does not create any new blockages, d_j still has some feasible routing path $P_j^{\mathrm{AT}_i+1}$ at time $\mathrm{AT}_i + 1$. Also, if $P_j^{\mathrm{AT}_i+1}$ is found by a shortest path algorithm, the Manhattan routing length of $P_j^{\mathrm{AT}_i+1}$ is equal to that of P_j^t in a 2-D plane. ■

Theorem 2: Routing a droplet with full bypassibility does not affect the overall chip routability but may increase the Manhattan routing length in a 2-D plane of unrouted droplets.

Proof: Consider two unrouted droplets d_i and d_j, and assume that both are on feasible routing paths P_i^t and P_j^t, respectively, at time t. Furthermore, assume that d_i has full bypassibility. After d_i is routed, new blockages B's around T_i from time $\mathrm{AT}_i - 1$ are introduced due to fluidic constraints. However, as B's are fully bypassible, d_j still has some feasible routing path $P_j^{\mathrm{AT}_i+1}$ at time $\mathrm{AT}_i + 1$. If $P_j^{\mathrm{AT}_i+1}$ is found by a shortest path algorithm, the Manhattan routing length of $P_j^{\mathrm{AT}_i+1}$ should be greater than or equal to P_j^t due to B's in a 2-D plane. ■

As shown in Algorithm 2, we first find a routable droplet d_i with the best bypassibility in line 1, and then route it in line 5. Accordingly, we need to update the routing base time (T_b) by returning $\mathrm{AT}_i + 1$ as in line 7. The next droplet will stall until T_b to accomplish fast 2-D routing. If there is a tie in terms of bypassibility, we route a shorter one first. After d_i is routed, we need to dynamically update the bypassibilities of all the unrouted droplets, as the shadowed region (which works as blockages) around S_i disappears, but new blockages appear around T_i. Note that bypassibility update can be done incrementally using a bucket list.

```
**Algorithm 2** Routing-Bypassibility
**Require**: A set of unrouted droplets $D_u$, a routing graph $G$,
    a routing base time $T_b$
 1: $S \leftarrow$ sort $D_u$ in desc. order of bypassibility
 2: for each $d_i \in S$ do
 3:   A path $P \leftarrow$ 2D min-cost path for $d_i$ after $T_b$ stalling
 4:   if $P \neq \emptyset$ then
 5:     Make $d_i$ routed with $P$
 6:     $D_u \leftarrow D_u \setminus \{d_i\}$
 7:     return $\mathrm{AT}_i + 1$
 8:   end if
 9: end for
10: return $T_b$
```

Consider the example in Fig. 5 where $D = \{d_1, d_2, \ldots, d_6\}$ are to be routed. While T_1, T_5, and T_6 are inaccessible due blockages or shadows by droplets, T_2, T_3, and T_4 are accessible. To decide the droplet to be routed first, we measure bypassibilities as in Fig. 6 which indicates that T_4 has full bypassibility. After d_4 is routed from S_4 to T_4 as in Fig. 5(b), we need

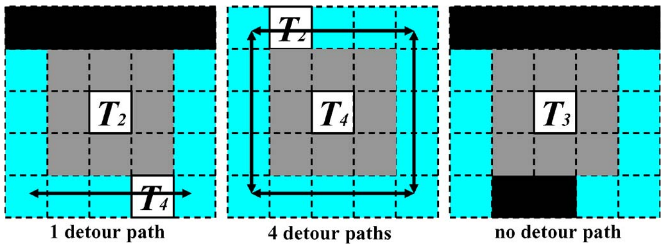


Fig. 6. This example shows bypassibility analysis of Fig. 4(a) where d_4, d_2, and d_3 have half (horizontal), full, and no bypassibility, respectively.

to update bypassiblities of all the unrouted droplets. Then, T_6 becomes accessible, as S_4 is released, and d_6 turns out to have full bypassibility. Thus, d_6 is routed after waiting at S_6 until $t = 14$. In the same fashion, routing d_3 follows, as shown in Fig. 5(d).

B. Routing With Concession

For a complex DMFB, a naive sequential routing of droplets can cause failure due to a deadlock between droplets. Consider the situation in Fig. 5(e) where d_1, d_2, and d_5 remain unrouted. Since d_1 and d_2 block the ways to T_2 and T_1, respectively, they form a deadlock. For such complex cases, 2-D routing by Algorithm 2 or A* search [1] is ended up with failure, and 3-D routing may fail too. According to our experiments in Fig. 5(e), routing either d_1 or d_2 in a 2-D or a 3-D plane without special consideration (which will be our concession) will cause failure eventually. Therefore, it would be desirable to move d_1 and d_2 simultaneously, but any parallel routing approach will increase computational complexity significantly.

```
Algorithm 3 Routing-Concession
Require: A set of unrouted droplets $D_n$, a routing graph $G$,
    a routing base time $T_b$
 1: $S \leftarrow$ sort $D_u$ in desc. order of dist. to concession zone
 2: for each $d_i \in S$ do
 3:    A path $P \leftarrow$ 3D min-cost path for $d_i$ after $T_b + \alpha_i$
       stalling
 4:    if $P \neq \emptyset$ then
 5:       Make $d_i$ routed with $P$
 6:       $D_u \leftarrow D_u \setminus \{d_i\}$
 7:       return $\mathrm{AT}_i + 1$
 8:    end if
 9: end for
10: return $T_b$
```

The only a sequential solution for Fig. 5(e) is to make d_1 back off and wait in some empty space, so-called *concession zone*, for sufficient amount of time until d_2 passes by. The concession zone is defined by any unoccupied continuous space in the chip which is larger than a 3×1 window. Hence, we first identify all the concession zones, and compute the shortest distances from all the unrouted droplets to any nearby concession zones. Then, we route a droplet with the longest distance before the others, as it is harder for such a droplet to migrate and wait in a concession zone, which is performed in line 1 of Algorithm 3. Regarding the example in Fig. 5(e) and (f), we route d_2 before d_1, as d_1 can migrate to a concession zone easily and wait there until the path taken by d_2 becomes available. To make such interaction between two droplets feasible, we stall the departure of a droplet like d_2 by some additional amount of time, α_i in Algorithm 3, which can be computed as follows:

$$\alpha_i = \sum_{j \in B_i \cap D_u} \left|x_j^s - x_j^t\right| + \left|y_j^s - y_j^t\right|$$

where B_i is a set of droplets whose source locations are inside the bounding box of d_i. Assuming $\alpha_2 = 0$ for Fig. 5(e) and (f), then at $t = 41$, d_2 is one grid above S_2 toward T_2, and d_1 is one grid right of S_1, which violates fluidic constraints. If we set $\alpha_2 = 5$ due to $B_2 \bigcap D_u = \{d_1\}$, d_2 first stalls for five clock cycles, which is enough for d_1 to escape from the shadowed region by d_2 and reach the concession zone safely. After d_1 waits until d_2 passes by, it returns to S_1 to head for T_1. Note that this is the only available path for d_1 to go to T_1 at this moment; thus, any min-cost path algorithm should be able to find this path including stalling in the concession zone. As in Algorithm 1, d_1 and d_2 start moving at $t = 39$ when the last successful routing based on bypassibility analysis (Routing-Bypassibility) occurred. As soon as d_1 is routed, the path from S_5 to T_5 becomes available. Thus, d_5 can be routed by Routing-Bypassibility from $\max(\mathrm{AT}_1 + 1, \mathrm{AT}_2 + 1) = 56$.

C. Solution Compaction

Algorithm 2 in Section IV-A allows only one droplet routing during a certain time interval, and the one in Section IV-B intentionally stalls the departure of a droplet to enhance routability. As a result, the routing resources are under low utilization, creating a large number of timing violations. Therefore, all the droplets, including any unrouted one, are rerouted greedily to compact the solution vertically or along the time axis. By rerouting each droplet in a greedy manner, we can increase the resource utilization and satisfy timing constraints without hurting routability. We can improve fault tolerance during compaction as well. According to previous works [2], [10], [12], using a smaller number of cells would improve fault tolerance, as the chance of getting defects can be reduced (assuming that each cell has the same probability of being defective). Therefore, during compaction, we try to minimize the number of cells at least used by any droplet in order to improve faulty tolerance.

Fig. 5(h) shows that the routing solution after the compaction is completed with timing constraint 20. The latest arrival time is reduced from 72 to 19, as the routing path for each droplet is optimized to meet timing. During this compaction, a droplet d_i with larger AT_i is rerouted first. Moreover, compare the path of d_5 in Fig. 5(g) with the one in Fig. 5(h). In Fig. 5(h), d_5 passes by the center of the design (around T_3) to minimize the number of unit cells in use to increase fault tolerance at a cost of larger AT_5 (which is still ≤ 20). This compaction is repeated until there is no improvement or maximum iteration is reached as in Algorithm 4.

TABLE III
COMPARISON BETWEEN THE PRIORITIZED A* SEARCH, THE TWO-STAGE ROUTING ALGORITHM, THE NETWORK-FLOW-BASED ALGORITHM, AND OUR ALGORITHM ON BENCHMARK SUITE I

test designs			Prioritized A* [1]		Two-stage [19]		Network-flow [2]		Our algorithm	
name	max droplets	size	u.cell[a]	CPU[b]	u.cell[a]	CPU[b]	u.cell[a]	CPU[b]	u.cell[a]	CPU
in-vitro_1	5	16x16	269	3.36	263	0.15	237	0.05	258	0.06
in-vitro_2	6	14x14	failed	n/a	failed	n/a	236	0.04	246	0.14
protein_1	6	21x21	failed	n/a	1735	1.33	1618	0.22	1688	0.47
protein_2	6	13x13	failed	n/a	failed	n/a	939	0.12	963	0.32

[a] total number of unit cells used for routing.
[b] the cpu time is from 1.2GHz Sun Blade-2000 machine with 8GB RAM.

Algorithm 4 Routing-Compaction

```
Require: A set of unrouted droplets $D_u$, a set of all droplets
    $D$, a routing graph $G$, a timing constraint RT
1: for each $d_i \in D_n$ do
2:     $\mathrm{AT}_i \leftarrow \infty$
3: end for
4: repeat
5:     $S \leftarrow$ sort $D$ in desc. order of $\mathrm{AT}_*$
6:     for each $d_i \in S$ do
7:         if RT $< \max\{\mathrm{AT}_i | \forall i\}$ then
8:             A path $P \leftarrow$ 3D min-cost path for $d_i$ for timing
9:             if $P \neq \emptyset$ and $\mathrm{AT}_i$ will improve then
10:                Make $d_i$ routed with $P$
11:            end if
12:        else
13:            A path $P \leftarrow$ 3D min-cost path for $d_i$ for fault
               tolerance
14:            if $\mathrm{AT}_i$ will be $\leq$ RT then
15:                Make $d_i$ routed with $P$
16:            end if
17:        end if
18:    end for
19: until no improvement or maximum iteration
```

In detail, Algorithm 4 shows two different phases, the first for timing (from lines 7–11) and the second for fault tolerance (from lines 13–16). Until a timing constraint is satisfied, we find a min-cost path where a cost is purely the distance. Once the timing constraint is met, we utilize the slack of each droplet to enhance fault tolerance by finding a different min-cost path where passing a unit cell already in use by others is encouraged. Therefore, fault tolerance will be pursued only if the timing constraint is satisfied.

D. Three-Droplet Routing Handling

In DMFB design, there can be a three-droplet routing case where either two droplets departing from different source locations get to the same target location after mixture or one droplet from a source location gets split into two for different target locations. We decompose such a three-droplet routing case into two typical two-droplet routing cases, and route them sequentially. In detail, we route one with longer Manhattan distance between its source and target first. Then, while routing the other one, we encourage this to share the path taken by the first one to improve routability as well as fault tolerance.

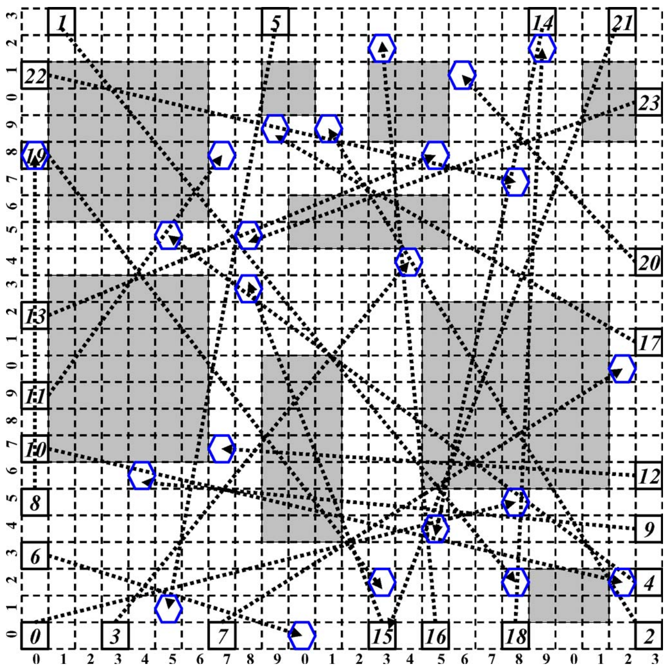


Fig. 7. Test16 in Table IV has over 20% blockages area and 24 droplets.

E. Runtime Complexity Analysis

From Algorithm 1, it is clear that Routing-Compaction in Algorithm 4 is the runtime bottleneck, because it repeats rerouting for all droplets to improve timing and fault tolerance using A* search. Let D denote a set of droplets and $G = (V, E)$ as a graph which models droplet routing problems. Rerouting a single droplet requires $O(|V|^2)$, when a min-cost path algorithm is adopted. Therefore, one iteration to reroute all droplets requires $O(|D||V|^2)$, where $|D|$ denote the number of droplets in the set D. Therefore, if we set the maximum number of iterations as M, the final runtime complexity of Algorithm 1 is $O(M|D||V|^2)$.

V. EXPERIMENTAL RESULTS

We implement the proposed droplet routing algorithm for DMFBs in C++, and perform all the experiments on an Intel 2.6-GHz 32-b Linux machine with 4-GB RAM. We compare our algorithm with various other known droplet routing algorithms [1], [2], [19] on two benchmark suites, Benchmark Suite I and Benchmark Suite II. Benchmark Suite I consists of widely used bioassays from [2] and [19], and Benchmark Suite II is a set of 30 hard test cases from ourselves. We make the same assumptions as in [2] and [19] for fair comparison.

TABLE IV
COMPARISON BETWEEN THE PRIORITIZED A* SEARCH, THE NETWORK-FLOW-BASED ALGORITHM, AND OUR ALGORITHM ON BENCHMARK SUITE II

test designs				Prioritized A* [1]			Network-flow [2]			Our algorithm			
name	max droplets	size	blockage area	failure[a]	la.time[b]	u.cell[c]	failure[a]	la.time[b]	u.cell[c]	failure	la.time[b]	u.celll[c]	d.bypass[d]
test 1	12	12x12	8 (5.6%)	0	37	66	2	n/a	n/a	0	100	67	7
test 2	12	12x12	9 (6.2%)	4	n/a	n/a	7	n/a	n/a	1	n/a	n/a	**8**
test 3	12	12x12	11 (7.6%)	4	n/a	n/a	6	n/a	n/a	1	n/a	n/a	3
test 4	12	12x12	11 (7.6%)	3	n/a	n/a	5	n/a	n/a	0	70	64	2
test 5	16	16x16	17 (6.6%)	0	28	108	2	n/a	n/a	0	78	118	**14**
test 6	16	16x16	14 (5.5%)	0	42	116	0	44	132	0	55	119	14
test 7	16	16x16	27 (10.5%)	0	33	104	3	n/a	n/a	0	89	113	9
test 8	16	16x16	26 (10.2%)	2	n/a	n/a	0	47	129	0	41	94	15
test 9	16	16x16	39 (15.2%)	4	n/a	n/a	3	n/a	n/a	1	n/a	n/a	9
test10	16	16x16	39 (15.2%)	4	n/a	n/a	2	n/a	n/a	0	77	110	9
test11	24	24x24	64 (11.1%)	0	62	252	0	100	264	0	47	249	**24**
test12	24	24x24	58 (10.1%)	3	n/a	n/a	0	80	242	0	52	219	22
test13	24	24x24	89 (15.5%)	0	60	241	2	n/a	n/a	0	52	247	19
test14	24	24x24	91 (15.8%)	3	n/a	n/a	2	n/a	n/a	0	57	234	19
test15	24	24x24	119 (20.7%)	0	63	246	0	74	233	0	83	230	17
test16	24	24x24	117 (20.3%)	4	n/a	n/a	3	n/a	n/a	0	63	223	19
test17	32	32x32	205 (20.0%)	9	n/a	n/a	2	n/a	n/a	0	68	394	**31**
test18	32	32x32	205 (20.0%)	4	n/a	n/a	0	88	408	0	91	403	**32**
test19	32	32x32	260 (25.4%)	0	70	402	2	n/a	n/a	0	90	371	**32**
test20	32	32x32	259 (25.3%)	3	n/a	n/a	0	91	382	0	99	393	24
test21	32	32x32	257 (25.1%)	8	n/a	n/a	2	n/a	n/a	0	76	389	22
test22	32	32x32	269 (26.3%)	5	n/a	n/a	4	n/a	n/a	0	85	393	27
test23	48	48x48	499 (21.7%)	6	n/a	n/a	0	100	681	0	78	738	**48**
test24	48	48x48	492 (21.4%)	8	n/a	n/a	0	99	737	0	94	807	**48**
test25	48	48x48	601 (26.1%)	5	n/a	n/a	0	100	729	0	91	792	**48**
test26	48	48x48	604 (26.2%)	3	n/a	n/a	0	99	709	0	88	798	**48**
test27	48	48x48	698 (30.3%)	4	n/a	n/a	0	100	770	0	99	762	47
test28	48	48x48	692 (30.0%)	5	n/a	n/a	4	n/a	n/a	0	99	808	**48**
test29	48	48x48	816 (35.4%)	7	n/a	n/a	6	n/a	n/a	0	98	733	**46**
test30	48	48x48	824 (35.8%)	8	n/a	n/a	4	n/a	n/a	0	88	751	41
total	864			106			61			3			752

[a] the number of failed droplets (unable to find a valid routing path or satisfy timing constraint).
[b] latest arrival time of droplets.
[c] total number of unit cells used for routing.
[d] the number of droplet routed based bypassibility and compaction using Algorithms 2 and 4 only.

A. Results on Benchmark Suite I

Table III compares the results from the widely used prioritized A* search [1], the two-stage routing algorithm [19], the state-of-the-art network-flow-based algorithm [2], and ours. The results of all the competitors are from [2]. Overall, it shows that our algorithm completes all the test designs in less than 1 s without any timing violation, as the network-flow-based algorithm does. Also, we achieve similar fault tolerance with the best known results (4% worse than that in [2]). Since Benchmark Suite I has only four fairly small/easy cases, we create a significantly harder test design to demonstrate the performance of our algorithm, which becomes Benchmark Suite II in the next section.

B. Results on Benchmark Suite II

We randomly generate 30 hard test designs with various potions of blockages to demonstrate the performance of our algorithm, which becomes Benchmark Suite II. In detail, for a given design size, the number of droplets is the same as the length of the longer side of the design. Then, multiple blockages are randomly generated and placed until the total area of blockages exceeds the given threshold. A source of each droplet is randomly placed on the boundary, while its target is randomly located at any place in the design. To prevent any trivially short case, the Manhattan distance in a 2-D plane between the source and target is forced to be longer than 50% of the length of the longer side of the design. We set a timing constraint of all the test designs as 100 time unit. Fig. 7 shows one test design at moderate difficulty, which is 24×24 with a 20.3% blockage area and has 24 droplets. For comparison, note that the hardest case of *in-vitro* in [19] is 16×16 with 6.3% blockage area and has only five droplets. We plan to release the benchmark circuits for the follow-up researches.

For comparison purpose, we implement the widely used prioritized A* search [1]. We also obtain the simulation results on our test designs from the author of the network-flow-based algorithm [2] which is shown to be superior to the prioritized A* search and the two-stage algorithm [19] as in Table III.

Table IV shows the overall comparison results. First, our approach shows significantly better routability by completing 27 test cases out of 30 (90.0%), while the priority A* search and the network-flow approach complete 8 (26.7%) and 12 (40%), respectively. In terms of the number of failures, our approach shows $35\times$ and $20\times$ better routability. This result is consistent with that in [2] in a sense that the network-flow-based algorithm is superior to the prioritized A* search. Overall, our algorithm yields stronger routability on harder/larger test designs.

Table IV also reveals the effectiveness of the proposed bypassibility analysis. We find that 752 out of 864 droplets (87%) can be routed by compaction and bypassibility analysis only (no concession), which is shown to be as powerful as the sophisticated network-flow-based algorithm for some cases. Regarding test17, the number of droplets routed by simply

TABLE V
COMPARISON BETWEEN THE PRIORITIZED A* SEARCH AND OUR ALGORITHM

test designs	Prioritized A* [1]			Our algorithm		
name	la.time	u.cell	cpu (sec)	la.time	u.cell	cpu (sec)
test 1	37	66	0.16	100	67	0.10
test 5	28	108	1.02	78	118	0.39
test 6	42	116	0.29	55	119	0.22
test 7	33	104	0.87	89	113	0.47
test11	62	252	1.26	47	249	0.67
test13	60	241	2.07	52	247	1.31
test15	63	246	2.72	83	230	1.15
test19	70	402	3.58	90	371	1.52
total		1535	11.97		1514	5.83

TABLE VI
COMPARISON BETWEEN THE NETWORK-FLOW-BASED ALGORITHM AND OUR ALGORITHM

test designs	Network-flow [2]			Our algorithm		
name	la.time	u.cell	cpu (sec)[a]	la.time	u.cell	cpu (sec)
test 6	44	132	0.24	55	119	0.22
test 8	47	129	0.16	41	94	0.29
test11	100	264	1.37	47	249	0.67
test12	80	242	0.35	52	219	1.28
test15	74	233	0.53	83	230	1.15
test18	88	408	1.14	91	403	1.55
test20	91	382	1.24	99	393	4,27
test23	100	681	2.51	78	738	4.17
test24	99	737	2.06	94	807	3.84
test25	100	729	3.48	91	792	4.42
test26	99	709	2.49	88	798	3.98
test27	100	770	3.44	99	762	5.40
total		5416			5604	

[a] the cpu time is from 1.2GHz Sun Blade-2000 machine with 8GB RAM.

bypassibility analysis is more than that by the network-flow-based algorithm. Our bypassibility-only based routing works as well as the network-flow-based algorithm for about 40% of test designs (these test designs are in bold).

Since the number of failed designs is so different, it is hard to compare runtime, timing, and fault tolerance. Therefore, we focus on the test cases which are completed by both our approach and another approach as in Tables V and VI. Table V shows that the prioritized A* search and our algorithm use a similar number of unit cells for routing, which implies similar fault tolerance, but our algorithm runs over 2$\times$ faster. Table VI compares our algorithm with the network-flow-based algorithm and shows that both achieve a comparable level of fault tolerance (ours is 3.3% worse). Unfortunately, we cannot directly compare the runtime, as Yuh *et al.* [2] have performed experiments on a completely different computing platform from ours (see the note below Table VI), but all the test designs listed in Table VI are completed in less than 6 s by our algorithm.

VI. CONCLUSION

The DMFB design is expected to be in a larger scale with higher complexity shortly due to its various applications and high efficiency. In order to cope with droplet routing automation, one of the key steps in DMFB design, we propose a high-performance droplet router with timing and fault tolerance taken into account. Experiments demonstrate that our algorithm works significantly better than the widely used prioritized A* search, the two-stage algorithm, and the state-of-the-art network-flow-based algorithm.

ACKNOWLEDGMENT

The authors would like to thank P.-H. Yuh, Prof. C.-L. Yang, and Prof. Y.-W. Chang from the National Taiwan University for providing experimental results of the network-flow-based algorithm on the test designs.

REFERENCES

[1] 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.
[2] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "BioRoute: A network-flow based routing algorithm for digital microfluidic biochips," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2007, pp. 752–757.
[3] R. Feynman, "There's plenty of room at the bottom: An invitation to enter a new field of physics," *Eng. Sci.*, vol. 23, no. 5, pp. 23–36, Feb. 1960.
[4] W. H. Ko, "Trends and frontiers of MEMS," *Sens. Actuators A, Phys.*, vol. 136, no. 1, pp. 62–67, May 2007.
[5] V. Srinivasan, V. K. Pamula, and R. B. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab Chip*, vol. 4, no. 4, pp. 310–315, Aug. 2004.
[6] S. K. Cho, H. Moon, and C.-J. Kim, "Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits," *J. Microelectromech. Syst.*, vol. 12, no. 1, pp. 70–80, Feb. 2003.
[7] T. Mukherjee, "Design automation issues for biofluidic microchips," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 463–470.
[8] S. K. Cho, S.-K. Fan, H. Moon, and C.-J. Kim, "Towards digital microfluidic circuits: Creating, transporting, cutting and merging liquid droplets by electrowetting-based actuation," in *Proc. MEMS Conf.*, Jan. 2002, pp. 32–35.
[9] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 223–228.
[10] 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.
[11] 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.
[12] T. Xu and K. Chakrabarty, "Integrated droplet routing in the synthesis of microfluidic biochips," in *Proc. Des. Autom. Conf.*, Jun. 2007, pp. 948–953.
[13] B. S. Gallardo, V. K. Gupta, F. D. Eagerton, L. I. Jong, V. S. Craig, R. R. Shah, and N. L. Abbott, "Electrochemical principles for active control of liquids on submillimeter scales," *Science*, vol. 283, no. 5398, pp. 57–60, Jan. 1999.
[14] K. Ichimura, S.-K. Oh, and M. Nakagawa, "Light-driven motion of liquids on a photoresponsive surface," *Science*, vol. 288, no. 5471, pp. 1624–1626, Jun. 2000.
[15] T. S. Sammarco and M. A. Burns, "Thermocapillary pumping of discrete drops in microfabricated analysis devices," *AIChe J.*, vol. 45, no. 2, pp. 350–366, Feb. 1999.
[16] T. B. Jones, M. Gunji, M. Washizu, and M. J. Feldman, "Dielectrophoretic liquid actuation and nanodroplet formation," *J. Appl. Phys.*, vol. 89, no. 2, pp. 1441–1448, Jan. 2001.
[17] M. G. Pollack, 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.
[18] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 11, no. 3, pp. 682–710, Jul. 2006.
[19] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proc. Des. Autom. Test Eur.*, 2006, pp. 1–6.
[20] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *Int. J. Rob. Res.*, vol. 24, no. 4, pp. 295–310, Apr. 2005.

[21] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 624–631.

[22] 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.

[23] G. Chaitin, "Register allocation and spilling via graph coloring," *ACM SIGPLAN Not.*, vol. 39, no. 4, pp. 66–74, Apr. 2004.

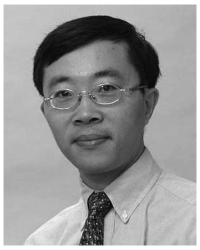
[24] P. Vitanyi, "How well can a graph be n-colored?" *Discrete Math.*, vol. 34, no. 1, pp. 69–80, 1981.

[25] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

Minsik Cho received the B.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 1999, the M.S. degree in electrical and computer engineering from the University of Wisconsin, Madison, in 2004, and the Ph.D. degree in electrical and computer engineering from The University of Texas at Austin in 2008.

He was with Intel during the summer of 2005 and with IBM T. J. Watson Research Center during the summers of 2006 and 2007. He is currently a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY. His research interests include nanometer VLSI physical synthesis and design automation for emerging technologies.

Dr. Cho received the Korean Information Technology Scholarship in 2002, Best Paper Award Nominations from ASPDAC 2006 and DAC 2006, Routing Contest Awards from ISPD 2007, and an IBM Ph.D. Scholarship in 2007, and the SRC Inventor Recognition Award in 2008.



David Z. Pan (S'97–M'00–SM'06) received the Ph.D. degree in computer science from the University of California at Los Angeles in 2000.

From 2000 to 2003, he was a Research Staff Member with IBM T. J. Watson Research Center. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, The University of Texas at Austin. He has published over 90 technical papers and holds five U.S. patents. His research interests include nanometer physical design, design for manufacturing, low-power vertical integration design and technology, and CAD for emerging technologies.

He has served or is serving as Associate Editor IEEE TRANSACTIONS ON CAD (TCAD), IEEE TRANSACTIONS ON VLSI SYSTEMS (TVLSI), IEEE TRANSACTIONS ON CAS-I (TCAS-I), IEEE TRANSACTIONS ON CAS-II (TCAS-II), and IEEE CAS Society Newsletter. He is also a Guest Editor of TCAD Special Section on "International Symposium on Physical Design in 2007 and 2008. He is in the Design Technology Working Group of the International Technology Roadmap for Semiconductor (ITRS). He has served in the Technical Program Committees of major VLSI/CAD conferences, including ASPDAC (Topic Chair), DATE, ICCAD, ISPD (Program Chair), ISQED (Topic Chair), ISCAS (CAD Track Chair), SLIP, GLSVLSI, ACISC (Program Co-Chair), ICICDT, and VLSI-DAT. He is the General Chair of ISPD 2008 and the Steering Committee Chair of ISPD 2009. He is an officer in the IEEE CANDE Committee (Workshop Chair in 2007 and Secretary in 2008). He is a member of the ACM/SIGDA Technical Committee on Physical Design and a member of the Technical Advisory Board of Pyxis Technology Inc. He has received a number of awards for his research contributions and professional services, including the ACM/SIGDA Outstanding New Faculty Award (2005), NSF CAREER Award (2007), SRC Inventor Recognition Award (2000 and 2008), IBM Faculty Award (2004–2006), IBM Research Bravo Award (2003), SRC Techcon Best Paper in Session Award (1998 and 2007), Dimitris Chorafas Foundation Research Award (2000), ISPD Routing Contest Awards, several Best Paper Award Nominations at DAC/ICCAD/ASPDAC, and ACM Recognition of Service Award. He is a Cadence Distinguished Speaker in 2007 and an IEEE CAS Society Distinguished Lecturer for 2008–2009.