

Placement and Routing for Cross-Referencing Digital Microfluidic Biochips

Zigang Xiao and Evangeline F. Y. Young

Abstract—Computer-aided design problems of digital microfluidic biochips are receiving much attention, and most of the previous works focus on direct-addressing biochips. In this paper, we solve the placement and droplet routing problem in *cross-referencing biochips*. In these biochips, the electrodes are addressed in a row-column manner, which may cause *electrode interference* that prevents simultaneous movements of multiple droplets. We propose a routing algorithm that solves the droplet routing problem directly. A two-coloring graph-theoretic method is used in our router to detect and prevent the electrode interference. In addition, we propose an integer linear programming based method to solve the placement problem. Our method considers the characteristics of cross-referencing biochips and is aware of droplet routing. Real-life benchmarks are used to evaluate the proposed methods. Compared with previous works, our router improves on average 4% in routing time and 58% in runtime. It can route all the benchmarks within the time limits, while the latest work fails in some cases. Moreover, experimental results show that by running our router on the placement result generated by our method and those generated by the latest work, an average improvement of 11%, 29%, 54%, and 46% in the maximum routing time, average routing time, stalling steps, and cell usage can be achieved.

Index Terms—Biochip, cross-referencing, droplet, microfluidics, placement, routing.

I. INTRODUCTION

MICROFLUIDIC-BASED biochips have received much attention today [1]. It shows great advantages in medical, pharmaceutical, and environmental monitoring applications [2]. Instances include immunoassays for point-of-care medical diagnostics, DNA sequencing, and detection of airborne particulate matter [2]–[9]. In contrast to conventional expensive and tedious laboratory procedures, advantages of miniaturized biochips include higher sensitivity, lower cost due to smaller sample and reagent volumes, higher levels of system integration, less human resource, and less likelihood of human error [10]. As a result, markets are opening up for such kind of devices. For example, the worldwide market for *in-vitro*

diagnostics in 2007 was estimated at \$38 billion [11]. The remarkable market indicates a prospective future of biochips.

Traditional biochips utilize continuous-flow microfluidic technologies. In continuous-flow systems, flow of liquid is piped through micro-fabricated channels. Simple and well-defined applications are made available in these systems. However, they are not suitable for highly complicate fluid manipulation because of the inherent complicated nature of continuous liquid flow. The design and analysis of these systems are barely tractable even for a moderate complex application. In addition, the systems have to be custom-made according to specific applications, because of the tight coupling of the system implementation and the functionality.

Digital microfluidic technology is utilized in second generation biochips. In these systems, nano-level biochemical material can be transported and processed in the form of tiny discrete “droplets.” Precise control of nanoliter droplets that contain biochemical reagents, samples, and buffers are made available in these devices. As a result, basic operation units such as mixing, detection, and diluting can be built. Consequently, various kinds of biochemical reaction can be modeled and performed on this platform. The droplets can be imported onto the chip via *dispensing* ports, and can be exported via *reservoir* ports. A droplet can be moved to one of its four neighboring cells by applying a control voltage over the target cell and at the same time deactivating the electrode at the droplet’s current position. As a result, the droplets can be moved to any free cells on the array. Detectors such as photodiodes and LEDs can also be integrated into the biochips to perform optical monitoring tasks. In contrast to previous continuous-flow biochip systems, digital microfluidic biochips (DMFBs) offers an important property, i.e., *dynamic reconfigurability* [13], because fluidic operations can be performed anywhere on the array. Hence, microfluidic modules can be viewed as “virtual devices,” with the corresponding droplet being routed by applying a series of adjacent electrodes. For instance, the mixing operation is accomplished by moving two droplets to the same location first, and then driving them together around some pivot point. Also, the operations can be reconfigured into different sizes [10]. The reconfigurability of DMFBs will be further discussed in Section II-B.

With its flexible design, many tasks can be performed in DMFBs without using expensive equipment and human resource. However, there is still not much existing design automation techniques for DMFBs. In paper [14], a novel top-down design methodology for DMFBs is proposed. In this

Manuscript received July 6, 2010; revised October 9, 2010 and January 7, 2011; accepted January 11, 2011. Date of current version June 17, 2011. This paper was recommended by Associate Editor K. Chakrabarty.

Z. Xiao is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61820 USA (e-mail: zxiao2@illinois.edu).

E. F. Y. Young is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: fyyoung@cse.cuhk.edu.hk).

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.2011.2113730

paper, the design of biochips is divided into *architectural-level* and *geometry-level* synthesis. The behavioral model for a biochemical assay is first acquired from the laboratory protocol and modeled as a sequencing graph. Then, architectural-level synthesis follows to generate the macroscopic structure of the biochip. *Task scheduling* and *resource binding* are then performed to schedule assay functions and bind them to existing resources so as to minimize assay completion time. Next, geometry-level synthesis that includes the module *placement* and *droplet routing* is performed to generate the detailed layout of the biochip. Analogous to traditional VLSI design, this methodology decouples the complex design flow into several tractable sub-problems. Designers can thus solve the whole problem in several manageable pieces.

Another important design consideration of biochips is the electrode manipulation method. It refers to the manner that the electrodes are controlled and activated by input pins. In a DMFB, the movements of droplets are controlled by the electro-hydrodynamic forces generated by the electrodes. Early designs of biochips used direct-addressing manipulation scheme, in which every electrode is associated with an input pin. Each cell can thus be addressed independently to bring about the droplets' movement. Although this scheme is straightforward and simple, it is not suitable for large array applications because the need for a large number of control pins will increase the production cost significantly. Pin-constrained addressing schemes are proposed to scale down the number of control pins. These schemes map the electrodes to a smaller number of control pins. In other words, each pin controls more than one electrode. Droplet-trace-based array partitioning method [15] and broadcast-addressing method [16] are two commonly seen methods. However, both of them lead to mappings of pins that are inevitably specific to some target application. A more promising design is *cross-referencing* electrode manipulation scheme, which addresses the electrodes in a row-column manner. Biochips that use this scheme are called *cross-referencing* biochips, in which high and low (or low and high) voltages are applied to a row electrode x and a column electrode y , respectively, so as to *activate* the cell at the intersection point (x, y) . However, because the control of the activation of a cell is in a row-column manner, *extra cells* may be activated when several droplets are moving together. Thus, the droplets may be affected unintentionally. This unwanted effect is referred to as *electrode interference*, and *electrode constraints* should be imposed to avoid such erroneous cases during droplet routing. Cross-referencing imposes tighter constraints to simultaneous manipulation of droplets' movements than direct-addressing. Nevertheless, one can maximize the parallelism by carefully arranging voltages. This will be further discussed in the following sections.

In this paper, we present automated design methods for the placement and droplet routing problem in cross-referencing DMFBs. Our placement method considers the properties of cross-referencing biochips and generates a routing input that is more suitable for cross-referencing biochip routers. In our router, a graph two-coloring method is used to address the electrode interference issue. It is shown that by combining both

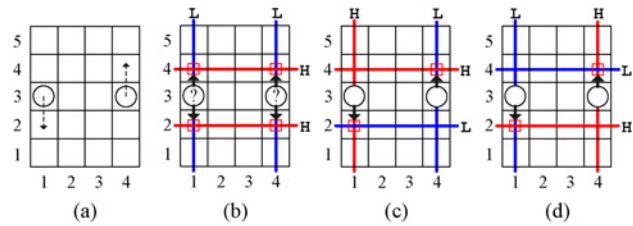


Fig. 1. (a) Two droplets are moving from current cell to the arrow-pointed cell. (b) Electrode interference happens. Droplets' movements are intervened. (c) Solution: apply voltage appropriately. (d) Alternative solution.

methods, high-throughput routing solution with high defect-tolerance for cross-referencing biochips can be achieved.

The rest of this paper is organized as follows. Section II gives more details about the cross-referencing biochips and the electrode interference issue, as well as the placement and droplet routing problem on it. In Section III, we discuss the related prior works on both problems and present our contributions. Sections IV and V describe our methods for the droplet routing problem and the placement problem. Note that in our approach to solve the placement problem, we will consider the properties of cross-referencing biochips and some details about droplet routing. Hence, in order to better explain our method, the routing sub-problem and our proposed router will be introduced in Section IV first. The placement problem and our approach will be introduced in Section V. Experimental results will be given in Section VI. Finally, conclusions will be drawn in Section VII.

II. PLACEMENT AND ROUTING IN CROSS-REFERENCING DIGITAL MICROFLUIDIC BIOCHIPS

A. Electrode Interference

In cross-referencing biochips, the control of the activation of a cell is in a row-column manner. *Extra cells* may be activated when more than one droplet are manipulated at the same time. Thus, some droplets may be affected by the *extra-activated cells* unintentionally. Hence, the cross-referencing electrode manipulation scheme imposes tighter constraints on simultaneous manipulation of droplets' movements than the direct-addressing scheme. An example is given in Fig. 1. Detail explanation of the example can be found in [32]. Note that those extra-activated cells do not necessarily imply electrode interference. If no droplet is around the extra-activated cell, no electrode interference will ever happen. The main challenge of routing droplet in cross-referencing DMFBs is to avoid electrode interference. In this sense, cross-referencing DMFBs have much tighter constraints than direct-addressing ones, which means the throughput may be severely decreased. However, as the example suggests, with the help of a well-designed router, cross-referencing biochips can still achieve high-throughput that is as good as that in direct-addressing biochips.

B. Placement in DMFB

Given scheduling and resource binding results generated from architectural-level synthesis, the placement problem involves placing the microfluidic modules and optimizing some

design metrics while satisfying various constraints. The objective is to place all the modules in the biochip and maximize the routability for later routing problems. Meanwhile, we want to maintain high fault-tolerance. We will discuss fault-tolerance of DMFBs later. Note that the placement problem of DMFBs is different from the traditional placement problem in electronic design [18] because of the reconfigurability. Since the biochips are reconfigurable, the modules are “virtual devices” that can be placed anywhere in the biochips as long as they do not overlap with each other at a specific time. However, the modules cannot be placed adjacent to the dispensing port; otherwise, it will block the importing of droplets.

C. Droplet Routing in DMFB

After the placement phase, routing will be done. The output of placement consists of locations of operations, i.e., modules. Before an operation can take place, the droplets have to be transported to the corresponding locations. Hence, the objective of droplet routing is to find valid routes for all the droplets without causing unexpected mixture, and the number of cells used should be minimized for better fault-tolerance. Note that between two successive operations, a certain amount of time interval is assumed to be reserved for routing the droplets. This forms the *timing constraint* of the droplet routing problem [12]. Although this time interval is relatively small compared to the duration of each sub-operation (e.g., 0.2 s versus 2 s), it is desirable to be minimized to prevent spoilage and ensure the correctness of subsequent operations. During droplet routing, at least one cell should be kept between droplets to prevent unintended mixing, except when droplets are to be mixed intentionally. Hence, *static fluidic constraint* and *dynamic fluidic constraint* are introduced between two droplets [12] to restrict the locations of droplets. Furthermore, recall that high and low (or low and high) voltage needs to be assigned to the row and column electrodes without causing electrode interference. This can be modeled as *electrode constraint* to ensure the correctness of droplet movements. Finally, the number of cells used by droplets during routing is preferred to be minimized to achieve better fault tolerance and robustness. As discussed in paper [19], minimization of the droplet routes leads to the minimization of the total number of cells used during routing. More spare cells are freed for fault-tolerance [14].

Note that the blocks in droplet routing are different from the blocks in traditional routing. In fact, they are *on-going operations*. Due to the reconfigurability of DMFB, these operations can be done anywhere on the biochip. In order not to violate the fluidic constraint [12], a *guarding ring* is imposed around a module with on-going operation. During routing, if the droplet inside a module is pulled outside unintentionally, the correctness of the whole bioassay will be ruined. Hence, it is very important to ensure the droplet staying inside a module when the operation is still in progress. However, as introduced in Section I, there may be extra-activated cells on the biochip during droplet routing. They are allowed to exist if and only if no electrode interference will be caused. Nevertheless, previous works on routing did not consider this potential problem. In this paper, we forbid the cells in the

guarding ring to be activated in order to prevent the droplet inside from being pulled out. The cells inside the guarding ring are allowed to be activated because the droplet inside will not be pulled out in any way.

III. PREVIOUS WORKS AND OUR CONTRIBUTION

A. Previous Works on Placement

There are several papers trying to solve the placement problem, but as long as we know, there is no previous work on placement for cross-referencing biochips. Su *et al.* presented a simulated annealing-based algorithm in [1]. They tried to take the fault tolerance issue into consideration by computing a ratio between the number of fault-free cells and the total number of cells. A unified synthesis algorithm was proposed by Su *et al.* in [20], in which placement is one of the steps. In their paper, a parallel recombinative simulated annealing that combines scheduling, resource binding, and placement together is proposed. Later, they even included routing into the annealing process [21]. They claimed that the synthesis and placement of DMFBs is very similar to the operation of dynamically reconfigurable FPGAs [22]. Yuh *et al.* proposed several representations in simulated annealing-based methods for the floor-planning problem, including *3-D-subTCG* [23] and *T-tree* [24]. They adopted the T-tree formulation to solve the placement problem in biochips [25]. This method is derived from the well-known 2-D placement method for reconfigurable devices. Defect cells are also modeled and considered in their work.

B. Previous Works on Routing

Existing methods of solving the cross-referencing droplet routing problem can be classified into two categories. The first category is to get a direct-addressing solution first, and then convert the solution to satisfy the electrode constraint in cross-referencing biochips. Griffith *et al.* tackled the problem using a graph coloring approach [26]. The problem is treated as a direct-addressing one and is solved first using a direct addressing method introduced in one of his previous works [27]. The solution is then converted to a cross-referencing result by coloring an interference graph. Su *et al.* proposed a two-stage process to solve the direct addressing droplet routing problem [12], the result is then modified to adapt to cross-referencing DMFBs by Xu *et al.* using a *clique partitioning* approach [28]. Since there are many existing methods for direct-addressing droplet routing, it is reasonable to adapt them for cross-referencing DMFB. Nonetheless, this kind of approach may be myopic since it loses the global view of the problem. The second category is to solve the problem directly, which is not limited by the direct-addressing result right from the beginning. Yuh *et al.* proposed an ILP based method to solve the problem [17]. They claimed that it is the first method that directly solves the problem, not relying on a direct-addressing result. The movement of droplets at each time step is determined by solving an ILP. A “droplet movement cost” is used to evaluate congestion when solving the ILP. This method works directly on cross-referencing biochips, and hence a better result can be obtained. However,

their approach cannot guarantee to finish all the routings within a given timing constraint and thus may eventually result in failure of the whole bio-application.

C. Our Contributions

The major contributions of our work include the following.

- 1) We propose an ILP-based method that solves the placement problem for cross-referencing biochips. Our method considers the properties of cross-referencing and is droplet-routing-aware. The effectiveness of our method is verified by running the proposed router on the placement results generated by our method and those by previous works, respectively. Experimental results suggest that our method can generate placements that are more suitable for cross-referencing biochip routers. To tackle the complexity of the ILP, a partitioning method is included. A maze routing is utilized in our pin assignment stage to eliminate unroutable solutions. Routability can be improved significantly with these extensions.
- 2) We design a router that directly solves the droplet routing problem in cross-referencing biochips. An elegant two-coloring approach is used in our router to handle the stringent electrode constraints. Experimental results show that the proposed router can achieve shorter routing time on average and can satisfy the timing constraints in all problems. Moreover, the number of cells used in the solution generated by our router is close to that in [29], although [29] is working on direct-addressing biochips that do not have those stringent electrode constraints. The extra-activated cells on the guarding ring of a module (see Section II-C) are handled in our router to avoid unwanted effects. A new rip-up and re-route process is designed for the router in which a probability-based scheme is used to allow non-deterministic ripping, which can break infinite looping effectively.

IV. OUR ROUTER FOR CROSS-REFERENCING BIOCHIP

A. Droplet Routing Problem

Given chip specification, assay description, and placement results, all the nets should be routed within the time limit without violating fluidic and electrode constraints while minimizing the number of cells used for routing. A cross-referencing DMFB can be viewed as a 2-D array with W rows and H columns. Each cell on the biochip can be referred to as (X, Y) . There are D droplets and B blockages. The fluidic ports on the boundary of a module, dispensers, or waste reservoir are called *pins*. For example, if a droplet d_i is to be routed from module M_1 to module M_2 , we need to know the source pin location that is around M_1 and the sink pin location that is around M_2 so as to compute a route. Droplet routes are modeled as *nets* [12]. Each droplet is either in a 2-pin net or a 3-pin net. A 2-pin net refers to a route that a single droplet needs to be routed from its source to sink, while a 3-pin net contains two droplets that start from different sources and reach at the

same sink. Let N be the number of nets. Given a droplet d_i , we denote the *source* and *sink pin* as s_i and *sink* t_i . As we mentioned, there is a *waste reservoir*. When a droplet reaches the reservoir, it will be removed from the biochip. Finally, time limit T is an upper bound on the total amount of time used to route all the nets. The output of the problem will be a *schedule of voltage assignment* at each time step. The schedule of voltage assignment includes a correct manipulation sequence of the electrodes, and it implies the movements of the droplets during the whole routing process. Note that we will target at minimizing the arrival times of the droplets at their destinations besides satisfying the timing constraint, since the time for droplet routing is part of the incoming bio-operation time [12]. Meanwhile, the number of cells used needs to be minimized, since it directly relates to the fault tolerance of the biochip.

B. Overview of Our Router

In this section, we will detail a droplet routing method that is named CrossRouter. In principle, our method first gets a reasonable routing order, and then each net is processed according to this order, while considering the nets that have already been processed. Rip-up and re-route will be performed if a net is unsuccessfully routed. The process continues until all the nets are routed. 3-pin nets are treated as two 2-pin nets and the two droplets can be merged during the routing. The solution obtained by our method contains a valid voltage assignment sequence in different time steps, which ensures that each droplet reaches its destination cell within time T without causing any violation. If the time limit for rip-up and re-route is exceeded, the routing process will be stopped, and failure will be reported to the previous synthesis stage, i.e., placement, and so on. Tasks should then be re-assigned or module should be replaced in order to avoid over-congested routing configuration.

When routing a net, there are two stages. The first stage is called *propagation*. Our algorithm tries to find a valid shortest path for a net in this stage. The propagation starts from the source of a net, and ends when it reaches the destination. Fluidic and electrode constraint check will be performed in each propagation step. A 3-D bitmap technique is used to enable quick detection of fluidic constraint violation, while an incremental two-coloring method is used during electrode constraint check to seek if there exists a feasible voltage assignment to implement the current set of droplet movements. A penalty count for each previously processed net will be incremented by one if it causes violation when routing the current net. These counts are used in the rip-up and re-route step to select the net to be ripped up. The second stage is the *backtracking* stage. After a valid path is found at the propagation stage, voltage assignment is *incrementally* performed based on the routing result. An overview of our algorithm is given in Algorithm 1.

C. Net Order Computation

The net order is computed before the routing actually begins. We first give a formal definition of the *bounding box of a net*.

Algorithm 1: CROSSROUTER

Input: Chip configuration and net information.
Output: Voltage assignment for each time step.

```

1 queue ← COMPUTENETORDER(N);
2 while queue is not empty do
3   net ← pop out head element of queue;
4   while net is not routed successfully do
5     result ← PROPAGATION(net);
6     if result = Success then BACKTRACK(net);
7     else if no net can be ripped then Report failure;
8     else RIPUPREROUTE(net);
9   end
10 end

```

Definition 1 (Bounding Box): A bounding box of a net is defined as the rectangular area of the maximum extent of the source pin and sink pin.

Intuitively, it is the rectangle defined by $\min(x)$, $\max(x)$, $\min(y)$ and $\max(y)$ where x and y are the x -coordinate and y -coordinate of the pins in a net.

The net order is computed according to the following conditions in descending precedence.

- 1) If a net i 's source point is inside the bounding box of another net j , i should be routed before j . Otherwise, it may be blocked by j during its routing.
- 2) Nets with a larger Manhattan distance from the source point to the sink point should be routed earlier, because all the nets have the same timing limits, but these nets have a longer distance to move.

D. Propagation Stage

In this stage, for a given net d_i , the router tries to find a route $\{d_i^0, d_i^1, \dots, d_i^T\}$ from its source to its sink without violating any constraint or causing any interference. In contrast to traditional routing, the route of a net here is the temporal positions of a droplet within the timing constraint, and is subject to the control of the electrodes.

Here, we present how the possible routes of a net are explored in CrossRouter. During the propagation, a droplet d_i may reach a cell $Q = (Q_x, Q_y)$ at time $t - 1$ and move to one of its adjacent cells P at time t . The pair (Q, t) , called a *droplet movement status*, denotes the position of the droplet at time t . $(Q, t - 1)$ is called the *parent* of (P, t) and is denoted by $(Q, t - 1) = \text{parent}((P, t))$. There are five possible movements for the droplet at time t . They are *LEFT*, *RIGHT*, *UP*, *DOWN*, and *STALL*. For example, when droplet is at (3, 6) at $t = 3$, then at $t = 4$, it can be moved to either one of (2, 6), (4, 6), (3, 7), (3, 5), or stall at the same position (3, 6). By following the parents of a status (P, t) all the way back, one can always trace out the route by which d_i is transported to P . Starting from a source point, a set of statuses is explored iteratively until the sink point is discovered. In order to find a desired path earlier, a *weight* is assigned to each status. It is calculated as follows:

$$\text{weight}(P, t) = t + MD(P, s_i) + Len(P, t) + U(P)$$

where $MD(A, B)$ is the Manhattan distance from A to B , $U(P) = N - \#used$, N is number of nets and $\#used$ is the

number of nets that used this cell before, and $Len(P, t)$ is the length of the current path from the source point to P at t . In this weight function, we want to minimize t so as to satisfy timing limit. $MD(P, s_i)$ is a standard term in a similar maze routing weight function that is used to control the path length. However, detour of cycles may exist in the routing path. $Len(P, t)$ measures the real path length and assigns a higher weight to this kind of path. Finally, $U(P)$ is used to minimize the cells used.

A sorted list of such statuses is maintained to record the routes that have been explored. In each iteration, the status (P, t) with the smallest weight is chosen. If the sink point is reached at time t , fluidic and electrode check should be performed from time $t + 1$ to T so as to ensure that this droplet will not block any processed net. Otherwise, new statuses due to propagation from P at time $t + 1$ will be added into the list, subject to the constraints that we will discuss in the next section. If the sink point is not reached and the list becomes empty, the routing of this net is failed. Rip-up and re-route will be performed. The fluidic and electrode constraint can be checked as follows.

1) *Fluidic Constraint Check:* Fluidic constraint check should be performed in order to prevent unexpected mixing of droplets during their transportation. To be more formal, let $d_i^t = (x_i^t, y_i^t)$ denote the location of droplet i at time t . Note that $s_i = d_i^0$ and $t_i = d_i^T$. The static and dynamic fluidic constraints can be stated as

$$\begin{aligned} |x_i^t - x_j^t| \geq 2 \quad \text{or} \quad |y_i^t - y_j^t| \geq 2 \quad \text{and} \\ |x_i^t - x_j^{t-1}| \geq 2 \quad \text{or} \quad |y_i^t - y_j^{t-1}| \geq 2. \end{aligned}$$

Furthermore, as the routing can be viewed as a 3-D routing, in our implementation, we use a 3-D bitmap to speed up the check.

2) *Electrode Constraint Check:* Electrode constraint check is a crucial part in CrossRouter. It can be seen that when there are only two moving droplets, there will always be a valid voltage assignment to move them correctly without causing any electrode interference. However, conflict may happen among three simultaneously moving droplets, because the conditions to activate and deactivate some cells may be contradictory. In this paper, we introduce a succinct graph coloring based method to determine whether the simultaneous movements of a set of droplets are feasible or not, i.e., whether a valid voltage assignment exists to implement the movements of several droplets at the same time. In contrast to the method proposed by Griffith [27], our approach does not attempt to find the chromatic number for a graph, which is NP-hard in general, but rather determines whether a movement is implementable. We will make use of a special type of *constraint graph* to perform this check.

A constraint graph $G = (V, E)$ in our context is an undirected graph that consists of two types of edges. The first kind is called *DIFF* edge, which means that the vertices at its two ends must have different colors, and the second kind is called *SAME* edge, which means that the adjacent vertices should have the same color. A two-coloring in this constraint graph is an assignment of two colors to the vertices such that the vertices sharing a *DIFF* edge have different colors, while

the vertices sharing a *SAME* edge have the same color. As discussed before, we can apply a high, low or ground voltage to a row or column and a cell will be activated when its intersecting row and column are assigned high and low (or low and high), respectively. In our electrode constraint check, we will have a vertex representing a row or a column in the constraint graph if this row or column must be set to a high or a low voltage in order to activate some cells. For those rows or columns which are not represented by a vertex in the graph, they are supposed to be set to the ground voltage and thus will not cause any electrode interference. Clearly, it is a polynomial time reduction from the instance of the voltage assignment constraint to the instance of constraint graph, which is a one-to-one correspondence. This concept is important allowing us to be able to check the electrode constraint efficiently using two-coloring.

At time t , a constraint graph G_t will be constructed according to the droplets' movements. We will insert vertices and edges into G_t in such a way that *there is a one-to-one correspondence between two-coloring of the graph and feasible voltage assignment on the biochip*. We will explain below how G_t will be constructed. We use $C(X)$ to denote a vertex in G_t that represents a column X and use $R(Y)$ to denote a vertex representing row Y . During the transition of a droplet from one cell Q to one of its adjacent cells P , only P is allowed to be activated in the droplet's 4×3 bounding box (BB), which is formed by the neighboring cells of P and Q . Note that the outer row/column perpendicular to the moving direction of the droplets is also forbidden, since its activation may cause unexpected movement of the droplet after it settles in P at time $t + 1$. However, if the droplet is stalling, all the cells in the droplet's 3×3 BB cannot be activated, while the droplet's current location Q can be activated or not.

According to the rules above, the constraints can be modeled by adding different types of edges into the graph. First of all, vertices $C(X)$ and $R(Y)$ will be added into G_t if and only if there is a droplet moving to the cell at (X, Y) at time t . We consider the move and stall action separately as follows.

- 1) If a droplet d_i is moving from Q to $P = (P_x, P_y)$, add a *DIFF* edge between $C(P_x)$ and $R(P_y)$. This is to activate the cell at P . For any neighboring cells (X, Y) of P and Q , if both $R(X)$ and $C(Y)$ exist in G_t (due to d_i or other droplets), add a *SAME* edge between $C(X)$ and $R(Y)$. This is to make sure that the neighboring cells are not activated.
- 2) If a droplet d_i is stalling at its current position P , we consider its neighboring cells (X, Y) . If both $R(X)$ and $C(Y)$ exist in G_t (due to other droplets), add a *SAME* edge between $C(X)$ and $R(Y)$. This is to make sure that the neighboring cells are not activated.

After constructing G_t , we can determine whether G_t has a two-coloring, which can be done efficiently. We can easily see that the existence of a two-coloring in G_t is equivalent to having a feasible voltage assignment to the rows and columns such that all droplets' movements can be achieved simultaneously. We illustrate the idea by giving a concrete example in Fig. 2 to demonstrate the construction and coloring of the constraint graph. The shaded bars are the rows and

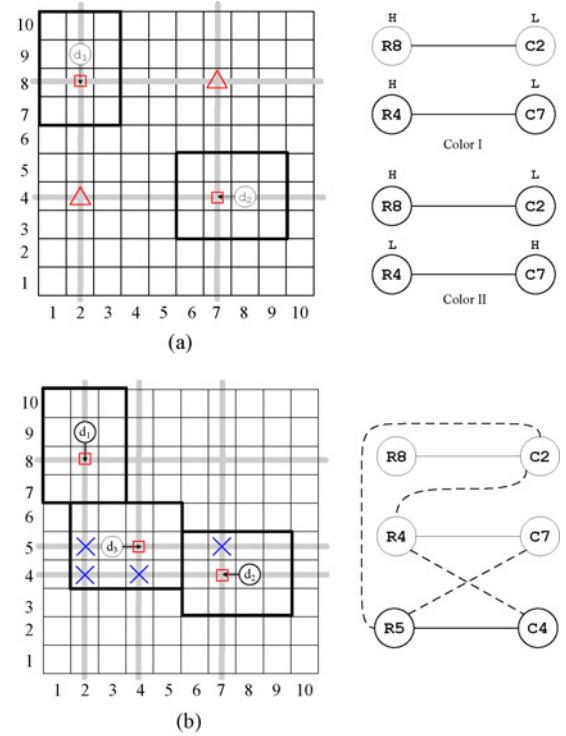


Fig. 2. Example of checking electrode interference with constraint graph. (a) Chip scenario and constraint graph considering d_1 and d_2 only. Two asymmetric two-coloring solution are shown while color I activates more cells (marked with “ Δ ”). (b) Chip scenario and constraint graph considering d_1 , d_2 , and d_3 simultaneously.

columns at which we need to apply high or low voltages. Small rectangles are the cell to be activated. The solid lines in the graphs on the right represent *DIFF* edges while the dotted lines represent *SAME* edges.

Fig. 2(a) shows an example that the router is routing d_2 at some time step while consider d_1 , which is already routed. Suppose we want to determine if moving d_2 to the desired cells, we need to activate (2, 8) and (7, 4). Four nodes $R(8)$, $C(2)$, $R(4)$, and $C(7)$ are created and *DIFF* edges are added between $R(8)$ and $C(2)$, and between $R(4)$ and $C(7)$. We can easily see that a two-coloring solution exists in the constraint graph, and thus a valid voltage assignment exists to bring about the movements of the two droplets.

Fig. 2(b) illustrates the scenario that d_3 is being routed via some route while d_1 and d_2 are already routed. Specifically, suppose d_3 is routed to the place as shown in the figure, we need to decide the movement of next step of d_3 . There are five possible choices to make, namely *up*, *down*, *left*, *right*, and *stall*. Suppose now we want to determine whether moving right to cell (4, 5) is *implementable*, we then perform the coloring steps as shown in Fig. 2(b). New nodes and edges are added based on the constraint graph in Fig. 2(a). The cells labeled with an \times [cells (2, 5), (2, 4), (4, 4), and (7, 5)] are those that their row and column vertices exist in G_t , i.e., we need to assign a high or low voltage to them, but we want to make sure that they will not be activated. Therefore, *SAME* edges will be added between their row and column vertices. For this constraint graph, we can easily determine that no two-

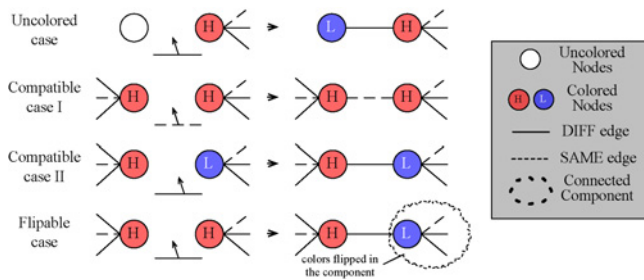


Fig. 3. Four cases of adding edge in incremental two-coloring.

coloring exists, and thus there is no valid voltage assignment such that the three droplets can move simultaneously. Now it turns out that there is no valid assignment, the router will seek for another possible movement, such as moving up to (3, 6). If all of the possible movements are not implementable, the router will drop this movement status and continue to search for other possible status.

3) *Incremental Coloring*: In order to avoid constructing the graph from scratch for every electrode check, we implemented an *incremental* two-coloring in which the two-coloring results (with colors assigned to the vertices) are kept for later use. For each time step t , we keep a colored constraint graph G_t . These graphs will be updated and saved whenever a new droplet is successfully routed. Specifically, every time a new net is routed, the corresponding new vertices and edges will be added into the constraint graph for each time step. The coloring solutions are preserved for later use. Given a time step t , we consider the routing of the next droplet d_j . We will reload the stored constraint graph G_t that contains the vertices and edges due to the movements of droplets d_1, d_2, \dots, d_{j-1} (assume that we process them in this order) at time t . When an edge $e = (u, v)$ is added due to this droplet d_j , we can check its *feasibility* as summarized in Fig. 3.

4) *Waste Reservoir*: Some droplets will be routed to a special location for disposal, i.e., *waste reservoir*. They will be routed to the waste reservoir and be removed from the biochip immediately it reaches there.

E. Backtracking Stage

After the current routing net reaches its sink point, a trace back is performed from its sink to its source to find out the actual path. Recall that to actually implement the routing of this droplet, we need to assign a voltage sequence to bring about the movements. The constraints that caused by this net, or equivalently, the corresponding vertices and edges, will be added into the constraint graph for each time step. Also, the coloring solution is added in an incremental manner. Note that if a droplet reaches its sink point before the time limit T . It will occupy the destination cell and act as a blockage for other droplets. Specifically, suppose a droplet reaches its sink at time t , it should stay at this sink point from time $t + 1$ to T . The fluidic and electrode constraint still apply. This subtle constraint must exist when processing the unrouted nets. Nevertheless, if this net's sink point is a waste reservoir, this droplet will be removed from the biochip at time $t + 1$ and thus no longer be an obstacle to other droplets.

F. Rip-Up and Re-Route Nets

During the routing, for each net, the number of conflict it contributes to the current routing net will be recorded. When a valid path cannot be found for the current net, rip-up and re-route will be performed. The conflict counts will then serve as probabilities to determine which net to be ripped. The reason is that the larger conflict count a net has, the more possible that after this net is ripped, the current net can be successfully routed. More importantly, other nets with smaller conflict counts still have chance to be ripped. This randomness can effectively break the tie when some nets are ripping each other as a loop. Note that we start the conflict count from one in order to give every net a chance to be ripped.

G. Complexity

Our algorithm maintains a queue during execution. At each time step t , there will be $O(WH)$ nodes be added into the queue. When processing a node, a $O(WH)$ two-coloring will be performed. There are N nets to be routed. Therefore, the time and space complexity of the algorithm is $O(TNW^2H^2)$.

V. DROPLET-ROUTING-AWARE PLACEMENT FOR CROSS-REFERENCING BIOCHIP

A. Problem Formulation

In architectural synthesis of DMFB, module placement is the step after task scheduling and resource binding. It can be viewed as a 3-D-packing problem where the third dimension is time. The starting time of each operation corresponds to the z -value of the bottom plane of a specific module. Since the module size and time span have been determined during task scheduling and resource binding, the problem can be reduced to a series of 2-D-packing problems at different time intervals. Specifically, whenever there are new modules that need to be placed into the layout region as time goes by, a new subproblem is created. The set of modules that need to be added are the objects we want to place. The existing on-going modules will act as blockages and the new modules shall not overlap with any of them. Hence, a set of consecutive subproblems will be defined from the given scheduling and binding result. After placement, all the droplets on the biochip must be routed to their destinations subject to some constraints. A formal description of the placement problem is given as follows:

- 1) input: scheduling and resource binding result, chip specification, including time limit, chip size, optical detector number, reservoir/dispenser number;
- 2) output: placement result, including module locations, reservoirs and dispensers and pin locations of each net;
- 3) objective: optimize the placement result such that it is easier for the proposed router to route.

B. Overview of Our Placement Method

Our placement method consists of three stages: dispenser and reservoir location generation, solving the placement problem using ILP and pin assignment. The first stage reads the input and decides the location of the dispensers and reservoirs.

TABLE I
NOTATIONS USED IN ILP FORMULATION

M^i	Module i
M_x^i, M_y^i	Lower left coordinate of module M^i
$X(M^i)/Y(M^i)$	Width/height of a mixing module i
W/H	Width/height of the array
$Center(M^i)$	Center point's coordinate of module M^i
L	A large constant
A_{ij}	Extended covered area bound by module/dispenser i and j

After these ports are determined, the placement problem is modeled as an ILP and solved by an ILP solver. Finally, after we obtained the module placement solution from ILP, we need to generate pin locations as the input for the routing problem.

C. Dispenser and Reservoir Location Generation

The dispenser and reservoirs should be located on the boundary of the biochip. They are not fixed before placement. However, once decided, their location is fixed during the bioassay. Here, we do not put them into our ILP formulation, but use a weighted method that based on the number of nets to place them on some specific positions. It is reasonable to distribute them regularly rather than giving them flexibility to locate arbitrarily on the boundary of the biochip, since some of them may cluster together and make the routing regions congested. More importantly, it can significantly reduce the solution space and the overheads by solving the ILP. The disadvantage is that the optimality of the solution may be hindered. But according to our experiment, the final result is mainly decided by the placement result. The algorithm is not discussed here because of the space limit. Details can be found in [32].

D. Solving Placement Problem Using ILP

We formulate an ILP to solve the module placement problem while considering the properties of cross-referencing biochip. The module placement problem in electronic design is known to be NP-hard [30]. In order to solve it efficiently, we try to model the problem with a scalable size of variables and constraints. The core idea in the formulation is the definition of the objective function. In the ILP, the formulation is the sum of a series of *extended covered area* formed by the routes in each subproblem. The *extended covered area* of a rectangle is defined as the vertical and horizontal area span, as illustrated in Fig. 4. Since in cross-referencing DMFB, the droplet movement is controlled by applying different voltages to row and column. If the extended covered area of the bounding box of a route is minimized, the route is shorter. Furthermore, it helps to reduce the interference between routes. The notation used in our ILP formulation is shown in Table I. Note that the index is starting from 0.

1) Constraints:

a) *Validity of modules:* The modules should be inside the biochip. A further requirement is that the module including guarding ring should be at least one space away from the chip boundary. This constraint is needed in order not to block the route from the dispenser nor to the reservoir. Note that this

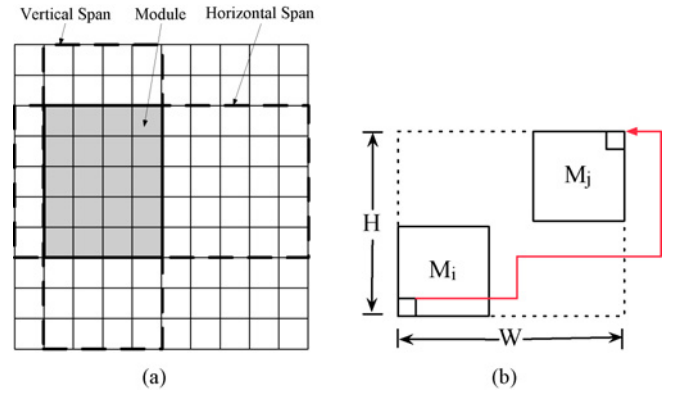


Fig. 4. (a) Illustration of extended covered area. (b) Illustration of droplet routing path, where $W=M_x^j - M_x^i + X(M^j)$, $H=M_y^j - M_y^i + Y(M^j)$. Note that there is detour. Hence, the length of the path is $\bar{W} + H + len_detour$, where len_detour is the length of the detour.

guarding ring can be shared between different modules. For module i , the above requirement is represented as

$$\begin{aligned} M_x^i &\geq 2 \quad \wedge \quad M_x^i + X(M^i) \leq W - 2 \\ M_y^i &\geq 2 \quad \wedge \quad M_y^i + Y(M^i) \leq H - 2. \end{aligned}$$

b) *Non-overlapping and separation of modules:* For modules that co-exist at some time, they must not overlap with each other. Furthermore, there should be a separation cell around each module. For a pair of module M^i and M^j which co-exist at the same time, we have the following constraints:

$$\begin{aligned} (M_x^j + X(M^j) < M_x^i - 1) \quad \vee \quad (M_x^i + X(M^i) < M_x^j - 1) \quad \vee \\ (M_y^j + Y(M^j) < M_y^i - 1) \quad \vee \quad (M_y^i + Y(M^i) < M_y^j - 1). \end{aligned}$$

Note that the “or” constraint cannot be directly written in linear programming and needs to be *linearized*. For this particular constraint, two binary variables are introduced. Let them be c_1 and c_2 , the previous constraint is transformed to

$$\begin{aligned} M_x^i - M_x^j - X(M^j) + L(c_1 + c_2) &> 1 \\ M_x^j - M_x^i - X(M^i) + L(c_1 + 1 - c_2) &> 1 \\ M_y^i - M_y^j - Y(M^j) + L(1 - c_1 + c_2) &> 1 \\ M_y^j - M_y^i - Y(M^i) + L(2 - c_1 - c_2) &> 1 \end{aligned}$$

where L is a large enough number (e.g., $W \times H$). It can be seen that among the four inequalities, only one of them will be left when the value of c_1 and c_2 are decided, others will be automatically satisfied due to the large value of L .

c) *Droplet-routing length constraint:* The maximum length of droplet paths needs to be controlled, because large value of the path length leads to a long routing time, which may cause timing constraint violation. Furthermore, long routing length is prone to be blocked by other blocks on the way. Hence, it is necessary to have a constraint to limit the routing length to be smaller than the timing constraint. Nevertheless, since we are not computing the actual routing path during the placement phase, we will allow some degree of excess, i.e., relaxation of maximum routing length. For a routing path to be completed between two modules M^i and M^j , the relaxation of maximum routing length is determined by the module size of M^j . Without loss of generality, suppose M^i locates to the

TABLE II
COMPARISON BETWEEN PROGRESSIVE ILP AND CROSSROUTER

Circuit	#P ^a	Progressive ILP		CrossRouter		Improvement	
		Max/Avg Cycle	CPU (s)	Max/Avg Cycle	CPU (s)	Avg (%)	CPU (%)
<i>In-vitro</i> _1	11	24/13.09	2.55	20/12.09	0.92	8	64
<i>In-vitro</i> _2	15	22/11.00	2.53	19/10.73	1.21	2	52
Protein_1	64	26/16.15	15.36	20/15.52	7.76	4	49
Protein_2	78	26/10.23	6.70	20/9.86	2.22	4	69

^aNumber of subproblems in a benchmark.

lower left of M^j , source pin and sink pin are generated around M^i and M^j , respectively. In the worst case, the droplets have to departure from the lower left corner of M^i , and arrives at the upper right corner of M^j . Hence, the relaxation value is set to $X(M^j) + Y(M^j) + d$, where d is a fixed parameter that can be tuned to allow extra relaxation for a congested layout area (recall that the droplet may encounter blockages during routing). An illustration is given in Fig. 4.

The constraint is modeled as follows:

$$M_x^j - M_x^i + M_y^j - M_y^i + X(M^j) + Y(M^j) + d \leq T$$

where T is the timing constraint of the biochip. This constraint helps us to retain routability in the final placement result.

d) *Optical detector resource constraint*: Because optical detectors are limited and fixed in some specific locations on the chip, the modules that are assigned to the same optical detector must have the same coordinate on the chip. Let Dt denotes a set of modules that are bound to the same optical detector, we have

$$(M_x^i, M_y^i) = (M_x^f, M_y^f), i \in Dt$$

where M^f is a module in Dt that first appear on the time line.

2) *Objective*: We use the bounding box as an estimation of the routes generated. Suppose that there is an operation, in which droplet i introduced by some module M^i is needed to be routed somewhere to form as input to M^j , we use the center points of both modules to form a bound box to model the route. Let $Center(M^i) = (x_i, y_i)$ and $Center(M^j) = (x_j, y_j)$, two pairs of variables, (x_{ll}, y_{ll}) and (x_{ur}, y_{ur}) , are introduced to denote the bottom left corner and upper right corner of the bounding box formed by both center points

$$\begin{aligned} x_{ll} &\leq x_i, & x_{ll} &\leq x_j, & x_{ur} &\geq x_i, & x_{ur} &\geq x_j \\ y_{ll} &\leq y_i, & y_{ll} &\leq y_j, & y_{ur} &\geq y_i, & y_{ur} &\geq y_j. \end{aligned}$$

Then, we can compute the extended covered area bound by this route, denoted as A_{ij}

$$A_{ij} = W(x_{ur} - x_{ll}) + H(y_{ur} - y_{ll}).$$

The objective is to minimize the sum of all the extended covered area $\min \sum_k A_k$, where $A_k, k = 1, 2, \dots$ is a set of all the extended covered area in the subproblems.

3) *Problem Partition*: Note that for some bioassays, there might be a large number of subproblems. When modeled as one ILP, the number of variables and constraints may be so huge that it is impossible to solve it efficiently. For example, there can be up to two thousands of variables and five

TABLE III
COMPARISON BETWEEN [29] AND CROSSROUTER

Circuit	Size	# Cells used	
		HPDRA [29]	CrossRouter
<i>In-vitro</i> _1	16 × 16	258	246
<i>In-vitro</i> _2	14 × 14	246	254
Protein_1	21 × 21	1688	1668
Protein_2	13 × 13	963	936

thousands of constraints in the largest benchmarks that we use. Hence, it is necessary to control the problem size. The whole problem can be partitioned into manageable problem sub-sets if the original problem set is too large. An ILP can be set up and solved for problem sub-set, and the ILP result from the previous problem sub-sets should be input as constraints for the current problem sub-set. Different partitioning strategies can be adopted here. In our implementation, we adopt a simple partitioning scheme that regularly divides the original problem set into sub-sets of the same size. Each sub-set may contain seven to 15 subproblems according to the chip size and the number of total subproblems in the benchmark. This strategy is simple yet working well according to the experimental result.

E. Pin Assignment

After the exact locations of the modules are obtained by solving the above ILP formulation, the result is not complete enough for routing. According to the placement result from [25], mixing module is modeled as a 3-pin net, dilute module is modeled as two 2-pin nets, optical detection and storage are modeled as 2-pin nets. The source pin and sink pin should locate around the bounding cells of the guarding ring of a module. Then, we have a set of possible locations for any specific pin. We will use a maze routing algorithm to help us to determine the best pin location that have a shorter routing distance. In particular, for a source pin and a sink pin, we have a set of possible source pin locations and a set of possible sink pin locations. We perform maze routing for each pair of the source and sink pin locations, and select the pair that has the shortest routing distance as the final pins. Note that the pin locations should satisfy the fluidic constraint. For example, two source pins are not allowed to be adjacent. Otherwise, error may occur if two droplets come out from the two pins in the same subproblem.

VI. EXPERIMENTAL EVALUATION

A. Overview of the Experiments

We use real-life bioassays as benchmarks. There are four sets of benchmarks, namely *in-vitro*, *in-vitro-2*, *protein*, and *protein_2*, and there are 11, 15, 64, and 78 sub-problems in each benchmark, respectively. The *timing constraint* for every subproblem in each benchmark is 20 time units. To evaluate the proposed routing method, we will compare with the state-of-the-art work in paper [17]. Both of our and their programs are implemented in C++. Their program is executed on a 1.2 GHz SUN Blade-2000 machine with 8 GB memory, while our CrossRouter is executed on an Intel 1.6 GHz machine¹

¹We use the slowest machine that available to us to make fair comparisons.

TABLE IV
 RUNNING CROSSROUTER UPON PLACEMENT [25] AND UPON OUR PLACEMENT

Name	#P*	Size	Routing Result of [25]			Routing Result of Our Placement			Improvement			
			Max/Avg Cycle	SS [†]	Cell Used	Max/Avg Cycle	SS [†]	Cell Used	Max	Avg	SS	Cell Used
<i>in-vitro</i> _1	11	16 × 16	20/12.09	26	246	17/9.55	9	148	15%	21%	65%	40%
<i>In-vitro</i> _2	15	14 × 14	20/11.07	37	254	15/5.13	3	84	25%	54%	92%	67%
Protein_1	64	21 × 21	20/15.63	49	1668	20/11.66	33	925	0%	25%	33%	45%
Protein_2	78	13 × 13	20/9.86	42	936	19/8.14	31	662	5%	17%	26%	32%
Average									11%	29%	54%	46%

*Number of subproblems.

[†]Number of stalling steps.

TABLE V
 RUNNING [29] UPON PLACEMENT [25] AND UPON OUR PLACEMENT

Name	#P*	Size	Routing Result of [25]		Routing Result of Our Placement		Improvement		
			Max/Avg Cycle	Cell Used	Max/Avg Cycle	Cell Used	Max	Avg	Cell Used
<i>in-vitro</i> _1	11	16 × 16	19/14.55	258	15/8.09	138	21%	44%	47%
<i>In-vitro</i> _2	15	14 × 14	20/11.87	246	18/6.07	115	10%	49%	53%
Protein_1	64	21 × 21	20/16.59	1688	20/10.53	983	0%	34%	42%
Protein_2	78	13 × 13	20/11.79	963	20/10.3	845	0%	13%	12%
Average							8%	35%	38%

*Number of subproblems

with 1.5 GB memory. To evaluate our placement method, comparisons will be made between our work and the method proposed in paper [25]. Specifically, we will use our proposed CrossRouter to route the placement generated by [25] and the placements generated by our approach. Then, the solution qualities are compared. Our placement method is implemented in the C++ programming language, and the ILP part is solved using `lp_solve 5.5` [31] and is run on a 2.4 GHz Intel Core II Duo machine with 1.5 GB memory. We limit the ILP solver to run for a given amount of time. The time limit is set to 30 min here, since from our empirical results, the quality of the solution is almost unchanged after 30 min.

B. Experimental Results

Table II gives the comparisons between progressive ILP [17] and CrossRouter. The “max cycle” in Table II stands for the time spent for routing the subproblem that takes the longest time to finish. And the “Avg cycle” stands for the average time to route a subproblem in a benchmark. The result shows that the routability of CrossRouter is better, since it can route all the subproblems within the timing constraint, while the progressive ILP approach in [17] has its maximum routing time exceeding the timing constraint. For instance, in the benchmark *in-vitro*_1, their router got a maximum finishing time of 24 cycles for some subproblems while ours are within the time limit of 20 cycles. Better results on average cycle time and CPU runtime are obtained, which demonstrates the good quality of our CrossRouter. Since the work in [17] does not optimize the number of cells used, we made another comparison with the high performance droplet routing algorithm proposed in paper [29], which is a droplet router for direct addressing biochips. Note that in direct addressing biochips, each cell can be activated independently, so there is no electrode interference and the droplet routing problem is

much simpler. Table III shows that although the constraints are much harsher in the problem we are dealing with, our router can get better result in terms of the number of cells used among the four benchmarks, except for *In-vitro*_2. The reason is that in subproblem 2 of this benchmark, their router found an earlier merging point of a 3-pin net in comparison with ours.

Table IV summarizes the comparisons of our placement method and the approach in paper [25]. To compare both methods, we run the proposed router on the placement results generated from both methods, respectively. In the experiment, we define the stalling steps (SS) as the total number of stalling in the route that a droplet has taken. Stalling means that at a certain step, a droplet has to wait for the path to be cleared, either for waiting some other droplet to pass through, or by queuing outside the waste reservoir, and so on. The major reason of stalling is due to the fluidic constraint, or more possibly, due to the electrode constraint for cross-referencing routing. For comparison, we run the router in [29] on the placement generated by [25] and on the placements generated by our approach. We can see from Table V that by using the placement result generated by our proposed method, the max/avg cycles have been improved and the number of stalling steps is reduced. Moreover, it is shown that the number of cell used has been reduced by more than a half comparing with the routing result of the placement generated by [25]. According to the experiment result of max/avg cycle, we can see that the router can finish routing in a relatively shorter time, which indicates shorter route and less congestion.

VII. CONCLUSION

We presented a droplet-routing-aware placement method and a systematic routing method for cross-referencing digital

microfluidics biochip. Electrode constraint is the most important issue in this type of DMFB. Our proposed router can handle the constraint elegantly and generates good quality solutions. To further improve the routing result, our placement method tries to utilize the property of cross-referencing DMFBs and generates placements that are more suitable for the router. The experiment result demonstrates the efficiency and effectiveness of the methods. By combining the effort of the placement and routing tool, the flexibility of cross-referencing DMFBs can be further exploited and the design and implementation of such chips can also be simplified.

REFERENCES

- [1] M. Pollack, A. Shenderov, and R. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab Chip*, vol. 2, no. 2, pp. 96–101, 2002.
- [2] E. Verpoorte and N. De Rooij, "Microfluidics meets MEMS," *Proc. IEEE*, vol. 91, no. 6, pp. 930–953, Jun. 2003.
- [3] T. Schulte, R. Bardell, and B. Weigl, "Microfluidic technologies in clinical diagnostics," *Clin. Chim. Acta*, vol. 321, nos. 1–2, pp. 1–10, 2002.
- [4] V. Srinivasan, V. Pamula, M. Pollack, and R. Fair, "Clinical diagnostics on human whole blood, plasma, serum, urine, saliva, sweat, and tears on a digital microfluidic platform," in *Proc. μ TAS*, 2003, pp. 1287–1290.
- [5] A. Guiseppi-Elie, S. Brahim, G. Slaughter, and K. Ward, "Design of a subcutaneous implantable biochip for monitoring of glucose and lactate," *IEEE Sensors J.*, vol. 5, no. 3, pp. 345–355, Jun. 2005.
- [6] R. Fair, A. Khlystov, T. Tailor, V. Ivanov, R. Evans, P. Griffin, V. Srinivasan, V. Pamula, M. Pollack, and J. Zhou, "Chemical and biological applications of digital-microfluidic devices," *IEEE Design Test Comput.*, vol. 24, no. 1, pp. 10–24, Jan.–Feb. 2007.
- [7] E. Ottesen, J. Hong, S. Quake, and J. Leadbetter, "Microfluidic digital PCR enables multigene analysis of individual environmental bacteria," *Science*, vol. 314, no. 5804, pp. 1464–1467, 2006.
- [8] Y. Zhao and S. Cho, "Microparticle sampling by electrowetting-actuated droplet sweeping," *Lab Chip*, vol. 6, no. 1, pp. 137–144, 2006.
- [9] 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, 2004.
- [10] K. Chakrabarty, "Design automation and test solutions for digital microfluidic biochips," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 57, no. 1, pp. 4–17, Jan. 2010.
- [11] *Global In Vitro Diagnosis Market Analysis PRLog Free Pree Release* [Online]. Available: <http://www.prlog.org/10080477-global-in-vitro-diagnostic-market-analysis.html>
- [12] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proc. Conf. DATE*, vol. 1, 2006, pp. 1–6.
- [13] J. Ding, K. Chakrabarty, and R. Fair, "Scheduling of microfluidic operations for reconfigurable 2-D electrowetting arrays," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 12, pp. 1463–1468, Dec. 2001.
- [14] 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.
- [15] T. Xu and K. Chakrabarty, "Droplet-trace-based array partitioning and a pin assignment algorithm for the automated design of digital microfluidic biochips," in *Proc. 4th Int. Conf. Hardw./Softw. Codesign Syst. Synthesis*, 2006, pp. 112–117.
- [16] T. Xu and K. Chakrabarty, "Broadcast electrode-addressing for pin-constrained multi-functional digital microfluidic biochips," in *Proc. 45th Annu. Des. Autom. Conf.*, 2008, pp. 173–178.
- [17] P. Yuh, S. Sapatnekar, C. Yang, and Y. Chang, "A progressive-ILP-based routing algorithm for cross-referencing biochips," in *Proc. 45th Annu. Des. Autom. Conf.*, 2008, pp. 284–289.
- [18] M. Sarrafzadeh and C. Wong, *An Introduction to VLSI Physical Design*. New York: McGraw-Hill Higher Education, 1996.
- [19] F. Su and K. Chakrabarty, "Design of fault-tolerant and dynamically-reconfigurable microfluidic biochips," in *Proc. Conf. DATE*, 2005, pp. 1202–1207.
- [20] F. Su and K. Chakrabarty, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. 42nd Annu. Conf. Des. Autom.*, 2005, pp. 825–830.
- [21] T. Xu and K. Chakrabarty, "Integrated droplet routing in the synthesis of microfluidic biochips," in *Proc. 44th Annu. Conf. Des. Autom.*, 2007, pp. 948–953.
- [22] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "Fast template placement for reconfigurable computing systems," *IEEE Design Test Comput.*, vol. 17, no. 1, pp. 68–83, Jan.–Mar. 2000.
- [23] P. Yuh, C. Yang, Y. Chang, and H. Chen, "Temporal floorplanning using 3D-subTCG," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2004, pp. 725–730.
- [24] P. Yuh, C. Yang, and Y. Chang, "Temporal floorplanning using the T-tree formulation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 300–305.
- [25] P. Yuh, C. Yang, and Y. Chang, "Placement of defect-tolerant digital microfluidic biochips using the T-tree formulation," *ACM J. Emerging Technol. Comput. Syst.*, vol. 3, no. 3, p. 13, 2007.
- [26] E. Griffith, S. Akella, and M. Goldberg, "Performance characterization of a reconfigurable planar-array digital microfluidic system," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 2, pp. 345–357, Feb. 2006.
- [27] E. Griffith and S. Akella, "Coordinating multiple droplets in planar array digital microfluidic systems," *Int. J. Robot. Res.*, vol. 24, no. 11, pp. 933–949, 2005.
- [28] T. Xu and K. Chakrabarty, "A cross-referencing-based droplet manipulation method for high-throughput and pin-constrained digital microfluidic arrays," in *Proc. DATE Conf. Exhibit.*, 2007, pp. 1–6.
- [29] M. Cho and D. Pan, "A high-performance droplet router for digital microfluidic biochips," in *Proc. Int. Symp. Phys. Des.*, 2008, pp. 200–206.
- [30] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of Computers and Intractability*. San Francisco, CA: Freeman, 1979.
- [31] M. Berkelaar, K. Eikland, and P. Notebaert, *lp_solve Version 5.5*, Des. Automat. Sec., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2006.
- [32] Z. Xiao and E. F. Young, "Placement and routing for cross-referencing digital microfluidic biochips," Masters thesis, Chinese Univ. Hong Kong, Shatin, Hong Kong, Aug. 2010.



Zigang Xiao received the B.S. degree in computer science from Sun Yat-sen (Zhongshan) University, Guangzhou, China, in 2008, and the M.Phil. degree in computer science from the Chinese University of Hong Kong, Shatin, Hong Kong, in 2010. He is currently working toward the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana.

His current research interests include physical design of very large scale integration circuits and

algorithm design.



Evangeline F. Y. Young received the B.S. and M.Phil. degrees in computer science from the Chinese University of Hong Kong (CUHK), Shatin, Hong Kong, and the Ph.D. degree from the University of Texas, Austin, in 1999.

Currently, she is an Associate Professor with the Department of Computer Science and Engineering, CUHK. She is now working actively on floorplanning, placement, routing, and algorithmic designs. Her current research interests include algorithms and computer-aided design of very large scale integration circuits.

Dr. Young has served on the technical program committees of several major conferences, including ICCAD, ASP-DAC, ISPD, and GLSVLSI, and also served on the Editorial Board of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.