CrossRouter: A Droplet Router for Cross-Referencing Digital Microfluidic Biochips

Zigang XIAO

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
zgxiao@cse.cuhk.edu.hk

Evangeline F.Y. Young

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
fyyoung@cse.cuhk.edu.hk

Abstract— Digital Microfluidic Biochip (DMFB) has drawn lots of attention today. It offers a promising platform for various kinds of biochemical experiments. DMFB that uses cross-referencing technology to drive droplets movements scales down the control pin number on chip, which not only brings down manufacturing cost but also allows large-scale chip design. However, the cross-referencing scheme that imposes different voltage on rows and columns to activate the cells, might cause severe *electrode interference*, and hence greatly decreases the degree of parallelism of droplet routing. Most of the previous papers get a direct-addressing result first, and then convert to cross-referencing compatible result. This paper proposes a new method that solves the droplet routing problem on cross-referencing biochip directly. Experimental results on public benchmarks demonstrate the effectiveness and efficiency of our method in comparison with the latest work on this problem.

I. Introduction

A. Digital Microfluidic Biochip

Digital Microfluidic Biochip shows great advantages in medical, pharmaceutical and environmental monitoring applications. Due to its flexible design, many tasks can be performed without using expensive equipment and human resource. However, highly automated CAD design support is strongly needed as in traditional VLSI design. Fig.1(a) gives a schematic diagram of a DMFB[1]. The DMFB can be viewed as a 2D-array, which allows various types of biochemical reaction taking place. There are dispensing reservoirs and optical detectors in a DMFB, which allow many kinds of laboratory research being carried out on the chip as in a miniaturized laboratory, i.e., lab-on-a-chip. The *discrete* liquid objects that contain chemical materials being operated on chip are called *droplets*. They are the basic units to carry out biochemical operations, such as *merging, spliting* and *diluting*.

B. Droplet Routing Problem in CAD for DMFB

A novel two level top-down CAD methodology for DMFB is proposed in [2]. *Operation scheduling* and *resource binding* are performed in the first level *architectural synthesis*, analogous to high-level synthesis in VLSI CAD. Then *module placement* and *droplet routing* are done in the second level *geometry synthesis*, analogous to physical design in VLSI CAD. The output of module placement consists of the time durance and locations of a series of consecutive sub-operations. Before an operation can take place, the droplets have to be transported to the corresponding locations. Hence valid routes for them without causing unexpected mixture must be found during droplet routing. Meanwhile, a certain amount of time interval will be reserved for routing the droplets between two successive sub-operations, which forms the *timing constraint* of droplet routing. Although this time interval is relatively small comparing to the duration of sub-operations, it is desired to be minimized to prevent spoilage and ensure the correctness of subsequent operations. Moreover, the number of cells used by droplets during routing is preferred to be minimized to achieve better fault tolerance and robustness, because cells may be defective due to manufacturing issues. Finally, after the solution (routes of droplets) is found, it can be stored into the DMFB to perform pre-programmed biochemical operations. Hence droplet routing is a fundamental design step and crucial to the reconfigurability of DMFB. [1]

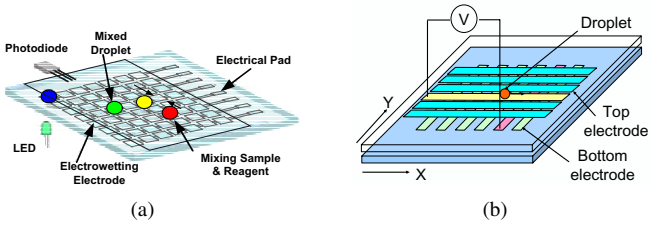


Fig. 1. (a) Schematic view of DMFB. (b) A Cross-referencing biochip.

C. Cross-Referencing DMFB

In a DMFB, droplets' movements are controlled by the electro-hydrodynamic forces generated by the electrodes, which cause the droplets to move horizontally or vertically. Currently, there exists two kinds of technology for the manipulation of droplet movement. The first one is called *direct-addressing*. In this scheme, an electrode is fixed under each cell of the chip. Each cell can thus be addressed independently to bring about the droplets' movement. It is simple, yet expensive and rather limited for large scale product. In order to scale down the control pin on chip, *pin-constrained* design of biochips has been proposed, in which one control pin may cor-

[1] Some other works will consider cell contamination and try to make routing path disjoint so as to avoid the residue left on the cross site corrupts the latter passing droplets. This can be addressed in our framework by modifying the weight function to discourage cell reuse, or by representing the paths of routed nets as blockages.

978-1-4244-5766-3/10/$26.00 ©2010 IEEE

responds to more than one electrode. However, the flexibility of the chip is sacrificed. Another more promising method is called *cross-referencing*, which applies high/low voltages to a row electrode and a column electrode respectively, so as to *activate* the cell at their intersection point. Fig.1(b) gives an example of a cross-referencing biochip [3]. 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 as *electrode interference* and *electrode constraint* 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.

D. Previous Work

Different methods [1, 4, 5, 6, 7, 8, 9] have been proposed to solve the droplet routing problem on direct-addressing biochip. In this paper, we will focus on the droplet routing problem on cross-referencing biochip. Griffith tackles this problem using a graph coloring approach [10]. 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 [7]. Then the solution is converted to a cross-referencing result by coloring a interference graph. Su proposed a two stage process to solve the direct addressing droplet routing problem [1], then the result is modified to adapt to cross-referencing DMFB by Xu , who used a *clique partitioning* approach [11]. Yuh proposed a state-of-the-art integer linear programming (ILP) based method to solve the problem [3]. They claim 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 as a heuristic to evaluate congestion when solving the ILP. This method works directly on cross-referencing biochip, and hence better result can be obtained. However, their approach cannot guarantee to finish all the routings within a given timing constraint and this might eventually result in failure of the whole bio-application. Moreover, the number of cells used is not taken into account, which is an important objective that need to be considered in droplet routing. Note that except for the paper [3], previous methods all try to find a direct-addressing result first and then convert it to a cross-referencing one. It might be myopic since it loses global view of the problem.

E. Our Contributions

In this paper, we propose a method called CrossRouter that directly solves the droplet routing problem on cross-referencing biochips by using a weighted maze routing framework. Our goal is to simultaneously move a group of droplets as much as possible to maximize the parallelism and minimize the total transportation time, as well as minimizing the total number of cells used during the routing process. Furthermore, in order to model the operations such as mixing, 3-pin net is also handled in CrossRouter. The major contributions of our work include:

- Apart from the progressive ILP [3], our CrossRouter is designed specifically for droplet routing on cross-referencing biochips.
- Public benchmarks are used to test CrossRouter. Comparison is made between CrossRouter and [3, 8]. The experimental result shows that CrossRouter can achieve shorter average routing time and satisfy timing constraints in all problems in comparison with [3]. Also, the number of cells used in the solution generated by CrossRouter is close to [8], although [8] is working on direct-addressing DMFB that do not have those stringent electrode constraints.
- We use an elegant 2-coloring approach to address the electrode constraint issue. At each time step, a *constraint graph* is constructed to determine the voltage assignment feasibility. In addition, an *incremental approach* is developed to avoid constructing the graph from scratch for each checking.

The remaining of the paper is organized as follows. In Section II we provide a formulation of the problem. Section III presents the details of our approach. Experimental result is given in Section IV. Finally, conclusions are drawn in Section V.

II. Droplet Routing On Cross-Referencing Biochip

A. Cross-Referencing DMFB

A Cross-referencing DMFB is given as a 2D array with W rows and H columns. Each cell on the chip can be referred to as (X, Y). There are D droplets and B blockages on the chip. The fluidic ports on the boundary of the microfluidic modules are called *pins*. Droplet routes between the pins of different modules are modeled as *nets* [1]. Each droplet is either in a *2-pin net* or a *3-pin net*. There are totally N nets on the chip. Given a droplet d_i (net i), we denote the start location and the end location of d_i as *source* s_i and *sink* t_i. Furthermore, there may be L *waste reservoirs* on the chip. When droplets reaches such locations, they will be removed from the chip at the next time step. Finally, a time limit T is given, which is an upper bound on the total amount of time used to route all the nets.

B. Fluidic Constraint

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 [1]: 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:

- $|x_i^t - x_j^t| \geqslant 2$ **or** $|y_i^t - y_j^t| \geqslant 2$, **and**
- $|x_i^t - x_j^{t-1}| \geqslant 2$ **or** $|y_i^t - y_j^{t-1}| \geqslant 2$

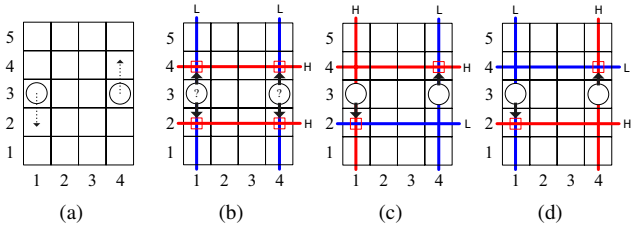


Fig. 2. (a) The two droplets are moving from current cell to the arrow-pointed cell. (b) Electrode interference happens, droplets' movements are intervened. (c) Solution: apply voltage appropriately. (d) An alternative solution.

C. Electrode Constraint

In a cross-referencing chip, we will apply high, low or ground voltages to the rows and columns. The cells on a cross-referencing chip can only be activated if the corresponding row and column are set to high and low (or low and high) voltage, respectively. However, this may introduce unwanted effects if the voltages are not set appropriately, since some cells near a droplet may be activated unintentionally. This imposes severe constraints during routing. An example is given in Fig.2. Suppose the left droplet needs to be moved from (1,3) to (1,2) and another from (4,3) to (4,4). We assign high voltage to row 2 and 4, as in Fig.2(b), while assigning low voltage to column 1 and 4, so as to activate both cells (1,2) and (4,4). However, cells (1,4) and (4,2) will also be activated, and they are called *extra activated*. Hence the droplets may not be moved as planned. If we can make appropriate voltage assignment as in Fig.2(c), correct movements can be guaranteed. Fig. 2(d) is an alternative solution by flipping all the electrodes in Fig.2(c) to its opposite value. For rows and columns which are not marked as 'H' or 'L', ground voltage is assigned and no cells will be activated along those rows and columns. 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 happen.

D. Timing Constraint

There is an upper bound T for the total droplet transportation time. Comparing to assay operations (e.g. mixing), the droplet routing time is very short. Based on this assumption, the module placement algorithm usually ignored the droplet routing time. However, to ensure that the result acquired is valid, there should be an upper bound value T for the total droplet transportation time, i.e., timing constraint.

III. Our Router for Cross-Referencing Biochip

In this section, we will detail a droplet routing method which is implemented in our CrossRouter. 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.

Our method first gets a reasonable routing order, then each net is processed according to this order, while considering the nets which have already been processed. If all the nets are routed successfully, the routing is finished. Otherwise, rip-up and re-route will be performed, in which some routed net will be ripped up and the routing order is rearranged. The process continues until all the nets are routed. Also, extra check is performed during rip-up and re-route in order to prevent deadlock. 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 CAD stage. Tasks should then be re-assigned or module should be replaced in order to avoid over-congested routing configuration.

The proposed method mainly consists of two stages. The first stage is called *propagation* stage. 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 3D bitmap technique is used to enable quick detection of fluidic constraint violation, while an incremental 2-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.

A. Computing Net Order

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; 2) Nets with a smaller Manhattan distance from the source point to the sink point should be routed earlier.

B. Propagation Stage

In this stage, for a given net d_i, the router tries to find a route $\{d_i^0, d_i^1, \ldots, d_i^T\}$ 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 $(Q, t-1) = parent((P, t))$. There are five possible movements for the droplet at time t. They are *LEFT, RIGHT, UP, DOWN, STALL*. For example, when droplet is at $(3, 6)$ at time=3, then at time=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 are explored iteratively until the sink point is discovered. In order to find a desired path earlier, a *weight* is assigned to each status which is calculated as follows:

$$weight(P, t) = t + MD(P, s_i) + U(P) + Len(P, t)$$

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.

A sorted list of such statuses are maintained to record the routes that have been explored. At 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.

In the following two sub-sections, we will describe how the fluidic and electrode constraint can be checked.

B.1 Fluidic Constraint Check

Fluidic constraint check as described in Section B should be performed in order to prevent unexpected mixing of droplets during their transportation. As this routing can be viewed as a 3D routing, in our implementation, we use a 3D bitmap to speed up the check.

B.2 Electrode Constraint Check

Electrode constraint check is a crucial part in CrossRouter. We can show 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. An example is illustrated in Fig.2. 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 [7], our approach do 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.

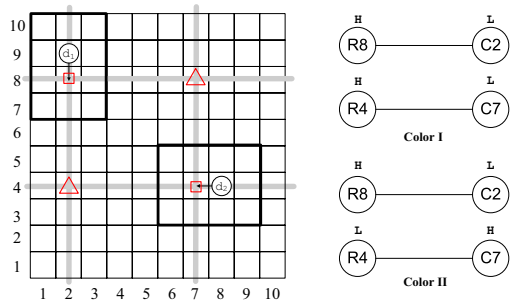
A *constraint graph* $G = (V, E)$ in our context is an undirected graph that consists of two types of edges. The first kind is called *DIFF* edge, which means that the vertices at its two ends must have different colors, and the second kind is called *SAME* edge, which means that the adjacent vertices should have the same color. A 2-color 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. (Note that this concept is important allowing us to be able to check the electrode constraint efficiently using 2-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 2-coloring of the graph and feasible voltage assignment on the chip.* We will explain below how G_t will be constructed. We use $C(X)$ to denote a vertex in G_t that represents 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.

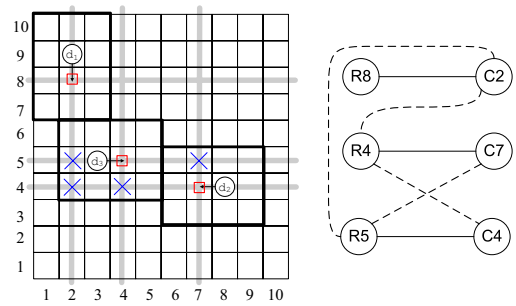
According to the rules above, the constraints can be modeled by adding different types of edges into the graph. First of all, vertices $C(X)$ and $R(Y)$ will be added into G_t if and only if there is a droplet moving to the cell at (X, Y) at time t. We consider the move and stall action separately as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

- 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 $C(Y)$ exists 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.
- If a droplet d_i is stalling at its current position P, we consider its neighboring cells (X, Y). If both $R(X)$ and $C(Y)$ exists in G_t (due to other droplets), add a $SAME$ edge between $C(X)$ and $R(Y)$. This is make sure that the neighboring cells are not activated.

After constructing G_t, we can determine whether G_t has a 2-color, which can be done efficiently. We can easily see that the existence of a 2-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.3 to demonstrate the construction and coloring of the constraint graph. The shaded bars are the rows and 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. In Fig.3(a), we need to activate (2,8) and (7,4), hence $DIFF$ edges are added between $R(8)$ and $C(2)$, and between $R(4)$ and $C(7)$. We can easily see that a 2-color exists in the constraint graph, and thus a valid voltage assignment exists to bring about the movements of the two droplets. Fig.3(b) considers droplet d_3 in addition to d_1 and d_2, new edges are added



(a) Chip scenario and constraint graph considering d_1 and d_2 only. Two asymmetric 2-coloring solution are showed while color I activates more cells(marked with 'Δ')



(b) Chip scenario and constraint graph considering d_1, d_2 and d_3 simultaneously.

Fig. 3. Example of checking electrode interference with constraint graph

into the constraint graph. The cells labeled with an 'X' (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 2-coloring exists, and thus there is no valid voltage assignment such that the three droplets can move simultaneously.

Note that this constraint graph is similar to the interference graph with coalescing vertices in the *register allocation* problem. It is known that N-coloring on this type of graphs is NP-hard when $N \geqslant 3$, while the corresponding 2-coloring problem is polynomial-time solvable. Adding the SAME edge constraint will not increase the problem difficulty, *i.e.*, 2-coloring of the proposed constraint graph is still polynomial-time solvable. Hence we can easily determine if the current movement of a droplet is feasible very efficiently by using the proposed method. Furthermore, in order to avoid constructing the graph from scratch for every electrode check, we implemented an *incremental* 2-coloring in which the 2-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. When we consider the routing of the next droplet d_j at time t, we will reload the stored constraint graph G_t that contains the vertices and edges due to the movements of droplets $d_1, d_2, \ldots, d_{j-1}$ (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 follows:

1. If an edge exists between u and v already, we check the compatibility of e with the existing one. Report failure if the two edges are of different types.
2. If no edge exist between u and v originally,
   (a) If either u or v has no color and its degree is 0 before adding the edge e, we can safely color it according to the type of e and the color of the other vertex.
   (b) If the type of e is compatible with the color of u and v, e.g., a $SAME$ edge and both u and v are HI, e can be safely added.
   (c) If the type of e is not compatible with the colors of u and v, we will try to *flip* the colors of all the vertices in u's (or v's) connected component. If the color of v (u) does not changed after this flip operation, e can be safely added, otherwise the 2-coloring is infeasible.

B.3 Handling 3-pin net

Some nets may consist of more than one droplets. This kind of net is called 3-pin net, which models the operation of merging. In our method, each subnet is treated as a 2-pin net and routed to its sink separately. During the routing of these subnets, they can be merged when they encounter each other.

C. Backtracking Stage

At this stage, the path of the current routing net will be found by tracing back from the sink. The constraints that caused by this net will be updated to the constraint graphs. If a droplet reaches its sink point before the time limit T. It should occupy the destination cell and serve as a blockage for other droplets. Suppose it reaches its sink at time t, then in the routing path of this net, the droplet's location should be at this sink point from time $t + 1$ to T. Nevertheless, if this net's sink point is a waste reservoir, this droplet will be removed from the chip at time $t + 1$ and thus no long be an obstacle to other droplets.

D. Rip-up and Re-route Nets

When a valid path cannot be found for the current net, rip-up and re-route will be performed. The net that causes the largest number of conflicts during the propagation stage of the current net will be ripped up. By ripping up this net, there will potentially be higher chance for the current net to be routed successfully.

IV. Experimental Results

Real-life bio-assays are used as benchmarks, they are *in-vitro*, *in-vitro-2*, *protein* and *protein-2*. The *timing constraint* for every subproblems in each benchmark is 20 time units.

Table I gives the comparison between progressive ILP [3] and CrossRouter. Both programs are implemented in C++. Their program is executed on a 1.2GHz SUN Blade-2000 machine with 8GB memory, while our CrossRouter is executed on an Intel 1.6GHz machine with 1.5GB memory. The max cycle in Table I stands for the time spent for routing the subproblem that takes longest time to finish. And the avg cycle stands for the average time to route all the subproblem in a benchmark. The result shows that the routability of CrossRouter is

TABLE I
COMPARISON BETWEEN PROGRESSIVE ILP AND CROSSROUTER.

Circuit	# P[a]	Progressive ILP		CrossRouter		Improvement	
		max/avg cycle	CPU time(s)	max/avg cycle	CPU time(s)	avg (%)	time (%)
*In-vitro*_1	11	24/13.09	2.55	20/12.09	0.92	8	64
*In-vitro*_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

[a]Number of subproblems in a benchmark.

TABLE II
COMPARISON BETWEEN [8] AND CROSSROUTER.

Circuit	size	# cells used	
		HPDRA[a]	CrossRouter
*In-vitro*_1	16×16	258	246
*In-vitro*_2	14×14	246	250
Protein_1	21×21	1688	1664
Protein_2	13×13	963	952

[a]High Performance Droplet Routing Algorithm for direct-addressing DMFB [8].

better, since it can route all the subproblems within the timing constraint, while the approach in [3] has its maximum routing time exceeding the timing constraint, e.g., in the benchmark *in-vitro*_1, their router got a finishing time of 24 for some subproblems while ours are within the time limit 20. Better result on average cycle time and better CPU runtime are obtained, which demonstrates the good quality of our solutions. Since the paper [3] does do not optimize the number of cells used in [3], we made another comparison with HPDRA [8], 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 II 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 than ours. Fig.4 is the routing result at time zero for the 8^{th} sub-problem of the *protein*_1 benchmark generated by CrossRouter. We can see that 6 droplet are moving at the same time. Many extra cells are activated, but no electrode interference has ever been caused.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a systematic routing method to solve the droplet routing problem on cross-referencing digital microfluidic biochip. We first formulated the droplet routing problem on cross-referencing DMFB, where our goal is to route all the droplets within the timing constraint while satisfying the fluidic and electrode constraints. Real-life benchmarks are used to evaluate our method. Experimental results illustrate the advantages of our method. To increase routability, the

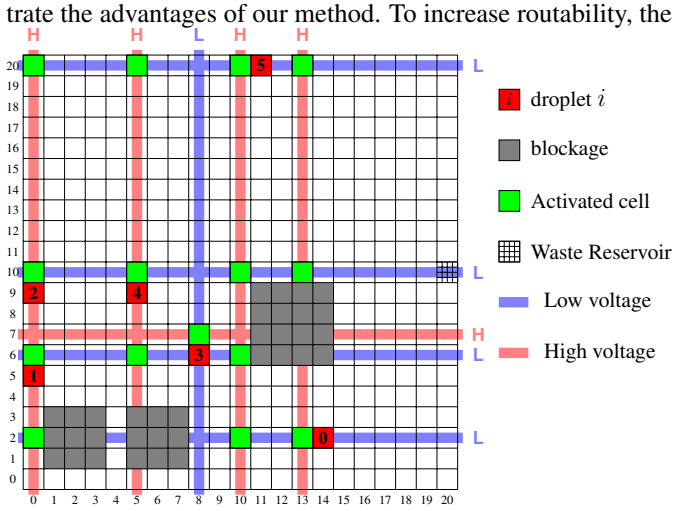


Fig. 4. Routing solution from the 8^{th} sub-problem of *protein_1* benchmark.

previous steps before routing, i.e., task scheduling, placement, etc., should consider potential electrode conflict beforehand, and take congestion into account, in order to decrease fluidic and electrode constraint during the routing stage. This will be further discussed in our later works.

REFERENCES

[1] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," *Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, p. 73, 2006.

[2] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," *IEEE/ACM International Conference on Computer Aided Design*, pp. 223–228, Nov. 2004.

[3] P. Yuh, S. Sapatnekar, C. Yang, and Y. Chang, "A progressive-ilp based routing algorithm for cross-referencing biochips," *Design Automation Conference*, pp. 284–289, 2008.

[4] K. Bohringer, "Optimal strategies for moving droplets in digital microfluidic systems," in *Seventh International Conference on Miniaturized Chemical and Biochemical Analysis Systems (MicroTAS'03)*, 2003.

[5] K. F. Bohringer, "Towards optimal strategies for moving droplets in digital microfluidic systems," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1468–1474, 2004.

[6] K. Bohringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Transactions on Computer Aided Design of integrated Circuits and Systems*, vol. 25, no. 2, p. 329, 2006.

[7] E. Griffith and S. Akella, "Coordinating multiple droplets in planar array digital microfluidic systems," *The International Journal of Robotics Research*, vol. 24, no. 11, p. 933, 2005.

[8] M. Cho and D. Pan, "A high-performance droplet router for digital microfluidic biochips," *Proceedings of the 2008 International Symposium on Physical design*, pp. 200–206, 2008.

[9] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Bioroute: a network-flow based routing algorithm for digital microfluidic biochips," *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design*, pp. 752–757, 2007.

[10] E. Griffith, S. Akella, and M. Goldberg, "Performance characterization of a reconfigurable planar-array digital microfluidic system," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, p. 340, 2006.

[11] T. Xu and K. Chakrabarty, "A cross-referencing-based droplet manipulation method for high-throughput and pin-constrained digital microfluidic arrays," *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 552–557, 2007.