PACOR: Practical Control-Layer Routing Flow with Length-Matching Constraint for Flow-Based Microfluidic Biochips*

Hailong Yao[1], Tsung-Yi Ho[2], and Yici Cai[1]
1. Tsinghua University 2. National Chiao Tung University
hailongyao@tsinghua.edu.cn,tyho@cs.nctu.edu.tw, caiyc@tsinghua.edu.cn

ABSTRACT

In flow-based microfluidic biochips, microvalves on the control layer need to be connected to control pins via control channels. In application-specific and portable microfluidic devices, critical microvalves need to switch at the same time for correct functionality. Those microvalves are required to have equal or similar channel lengths to the control pin, so that the control signal can reach them simultaneously. This paper presents a practical control-layer routing flow (PACOR) considering the critical length-matching constraint. Major features of PACOR include: (1) effective candidate Steiner tree construction and selection methods for multiple microvalves based on the deferred-merge embedding (DME) algorithm and maximum weight clique problem (MWCP) formulation, (2) minimum cost flow-based formulation for simultaneous escape routing for improved routability, and (3) minimum-length bounded routing method to detour paths for length matching. Computational simulation results show effectiveness and efficiency of PACOR with promising matching results and 100% routing completion rate.

1. INTRODUCTION

Flow-based microfluidic biochip, also known as Lab-on-a-Chip (LoC), has emerged as a revolutionary technique toward miniaturization for the automation of laboratory in biochemistry [1–3]. LoC integrates different biochemical analysis modules, such as mixer, separator, chamber, etc., into a single chip, and hence reduces samples/reagents to microliter or even nanoliter scale [4]. Compared with the traditional laboratory procedures, LoC greatly improves the sensitivity, precision, and throughput, as well as reduces the analysis time and sample/reagent consumption [5]. As a result, microfluidic biochips have many promising applications in biochemical analysis including enzymatic assays, DNA sequencing, cell-based assays and immunoassays. As LoC gets larger and bioassay gets more complicated, computer-aided design (CAD) methods are becoming necessary for reliable automatic LoC design with reduced time to market.

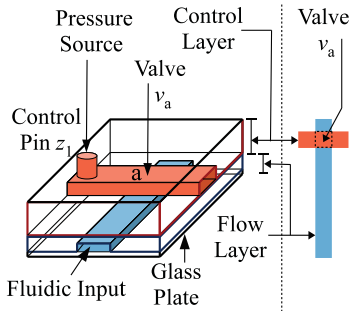

Figure 1: Schematic of flow-based microfluidic biochip [8].

Large scale integration for flow-based microfluidic biochips is based on the new technology called *soft lithography*, which can fabricate all necessary microfluidic units within a single biochip using elastomer material (polydimethylsiloxane, PDMS) [6]. Figure 1 shows an example of the flow-based microfluidic biochip, which includes microchannels on both *flow layer* and *control layer* [5, 7, 8]. Hydraulic or pneumatic *microvalve* (referred to as *valve* hereafter) based on flexible membranes is manufactured between the two layers. Valves are open or closed by the pressure in the control channels, which is injected from *pressure source* via the *control pin*. Different complex units can be formed in a single biochip by combining several valves, such as micropumps, mixers, multiplexers, etc [5]. By controlling the complex units through different actuation patterns on valves, fluids (i.e., samples/reagents) are manipulated to flow in microchannels on the flow layer for different fluidic operations, e.g., splitting, mixing, filtering, metering, etc. Thus, an assay can be realized by executing the sequential fluidic operations automatically in a programmed way [9]. With the advancements of fabrication techniques in multilayer soft lithography, the size of valves has been reduced to 8×8 and $6 \times 6 \mu m^2$ [10, 11]. And the valve density has reached 1 million valves per cm^2, which makes manual design impossible for current microfluidic very large-scale integration (mVLSI). Therefore, CAD methods for mVLSI are necessary.

Control-layer routing is one of the most critical steps in the CAD flow for flow-based microfluidic biochips, which determines whether valves can work as expected for correct functionality in the different microfluidic devices. This paper focuses on the control-layer routing problem: *Given the positions of the valves and the valve switching time table, compute the assignment from valves to different control pins*[1] *and the control channels connecting the clusters and control pins, satisfying the length-matching constraint and design rules. The objective is to improve the overall routability including clusters with the length-matching constraint, and minimize the total length of control channels.*

*The work of H. Yao was supported in part by Tsinghua University Initiative Scientific Research Program (20141081203), by Doctoral Fund of Ministry of Education of China (20111011328), and by the National Natural Science Foundation of China (61106104). The work of T.-Y. Ho was supported in part by the Taiwan Ministry of Science and Technology under grant no. MOST 102-2221-E-009-194-MY3, 103-2220-E-009-029, and 103-2923-E-009-006-MY3. The work of Y. Cai was supported by the National Natural Science Foundation of China (61274031).

DAC'15, June 07–11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2744887.

[1]Each control pin is connected to a pressure source, which may drive multiple valves according to valve switching compatibility.

Using the flexible PDMS material, pressure propagation is very slow from the control pin to the corresponding valve(s) through the control channel [12]. Moreover, the propagation time will greatly increase due to reduced driving force in future portable microfluidic devices. Thus, a critical issue emerges for those functional units with synchronization requirements. For two valves with the synchronization requirement, it is very important to guarantee the pressure from the control pin to reach the valves simultaneously. Otherwise, functional errors will occur, which will result in wrong assay results. Therefore, the lengths of the control channels from the control pin to different valves are required to be equal or at least within a given bound on delta length difference. In this paper, we call such critical requirements for control channels as *length-matching constraint*. To the best of our knowledge, this paper is the first one that addresses the length-matching constraint during control-layer routing.

In the past decade, noticeable advances have been made in CAD methodology for digital microfluidic biochips, including resource binding, operation scheduling, module placement, and droplet routing [15–19]. However, only a few works addressed the CAD problems for flow-based biochips. Minhass et al. presented a constraint programming based synthesis method for flow-based biochips [20]. Lin et al. presented a flow channel routing method, which simultaneously minimizes the total length and the maximum length of flow channel [21]. Amin et al. presented the first control-layer routing method for flow-based microfluidic biochips [22]. Minhass et al. presented a control synthesis method, which addresses the assignment from valves to control pins for minimized number of control pins [8]. In [23], a control-layer routing algorithm is presented to minimize the pressure-propagation delay of the control signals. However, no existing works consider the length-matching constraint in the control-layer routing for flow-based microfluidic biochips. And there is no complete and robust control-layer routing flow which simultaneously optimizes the total channel length and routability with the length-matching constraint.

In this paper, we present the first practical control-layer routing flow called PACOR, which addresses the length-matching constraint. Using PACOR, each cluster of valves with the length-matching constraint is routed with matched length, i.e., the length differences for all the valves within the cluster to the corresponding control pin are less than a given threshold value. Moreover, the total control channel length is minimized. Major contributions of the paper are as follows.

- The first practical control channel routing flow considering the length-matching constraint is presented with enhanced routing completion rate and minimized channel length.
- Effective obstacle-avoiding candidate Steiner tree construction and selection methods are presented to facilitate length-matching and enhance routability.
- A minimum cost flow-based routing method is presented for escape routing for control pins, which effectively improves routability with minimized channel length.
- A minimum-length bounded routing method is presented for detouring the routing paths with pre-specified lower bound on the path length.

The remainder of this paper is organized as follows. Section 2 presents the problem formulation. Section 3 presents the overall flow of the control channel routing system PACOR. Section 4 presents the length-matching aware cluster routing method. Section 5 presents the control pin routing based on minimum cost flow formulation. Section 6 presents the minimum-length bounded routing method for path detouring and length matching. Section 7 presents and discusses the computational simulation results. Finally, conclusion is drawn in Section 8.

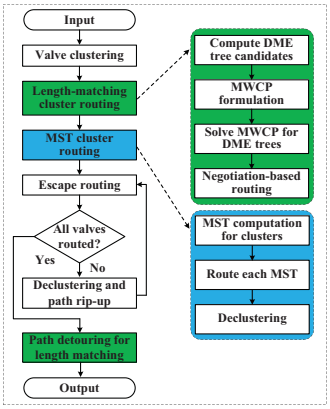

Figure 2: Design flow of our approach.

2. PROBLEM FORMULATION

In flow-based biochips, each valve is driven by a "0-1-X" sequence measured by units of time steps [8]. Here, "0" denotes the valve is open, "1" denotes the valve is closed, and "X" denotes the valve is either open or closed.

DEFINITION 1 (*Activation sequence*). *$S(v) = a_1, a_2, \ldots, a_n$ for valve v is called an* activation sequence*, where a_i denotes the* activation status *("0", "1", or "X") at time step i, and n denotes the total number of time steps.*

The activation sequences for all the valves in the flow-based biochip are of equal length, which are obtained by the resource binding and scheduling process [8].

DEFINITION 2 (*Compatible activation status*). *Activation status a_i and a_j are called* compatible*, denoted as $a_i \cong a_j$ if and only if any of the following conditions are satisfied: (1) $a_i = a_j$, (2) $a_i = X$, (3) $a_j = X$.*

DEFINITION 3 (*Compatible activation sequence*). *Activation sequences $S(v_k) = \{a_{k1}, a_{k2}, \ldots, a_{kn}\}$ and $S(v_l) = \{a_{l1}, a_{l2}, \ldots, a_{ln}\}$ are called* compatible*, denoted as $S(v_k) \cong S(v_l)$, if and only if $a_{ki} \cong a_{li}$ ($\forall 1 \le i \le n$).*

DEFINITION 4 (*Compatible valve*). *Valves v_k and v_l are called* compatible*, denoted as $v_k \cong v_l$, if and only if $S(v_k) \cong S(v_l)$.*

With the above definitions, this paper addresses the following problem:

Control Synthesis and Routing Problem with Length-Matching Constraint.

Given: All the valves V with coordinates, the valve compatibility information (i.e., pairs of valves that are compatible with each other), the clusters of valves $\mathcal{M}(V)$ with length-matching threshold value δ, the feasible control pin positions CP, and the design rules for minimum channel spacing and minimum channel width.

Find: The control channel routing paths connecting valves to control pins with minimized total channel length.

Subject to: (i) The design rules must be satisfied. (ii) All the valves connected to the same control pin must be pairwise compatible with each other. (iii) For each cluster of valves $m(V) \in \mathcal{M}(V)$, the condition $|l(v_i) - l(v_j)| \le \delta$ ($\forall v_i, v_j \in m(V)$) must be satisfied, where $l(v_i)$ denotes the routed channel length from valve v_i to the corresponding control pin $cp_k \in CP$.

Please note that the length-matching constraint must conform with the valve compatibility information, i.e., each pair of valves with the length-matching constraint should also be compatible with each other, and hence it is meaningful to connect a single control pin to the valves with matched lengths.

3. OVERALL FLOW

Figure 2 shows the overall flow of PACOR for control channel routing with the length-matching constraint.

Valve clustering: For control-layer routing under the broadcast addressing scheme, the clusters of valves are first computed. Here, clusters with the length-matching constraint are maintained. All

the valves in each cluster should be compatible with each other. The objective is to minimize the number of clusters so as to minimize the the number of control pins. We adopt the max-clique formulation to solve the valve clustering problem. As the problem is NP-complete [29], a fast heuristic algorithm is used to compute the clusters.

Length-matching cluster routing: The most difficult and challenging part is to route multiple clusters with the length-matching constraint on a single layer. We present a new length-matching aware cluster routing method, which includes four modules: (1) candidate Steiner trees construction based on the deferred-merge embedding (DME) algorithm [24], (2) maximum weight clique problem (MWCP) formulation for candidate Steiner tree selection, (3) solve MWCP problem to obtain the Steiner tree solutions, and (4) negotiation-based routing algorithm for improved routability. Details of this stage are described in Section 4.

MST-based cluster routing: The remaining clusters without the length-matching constraint are routed using the minimum spanning tree (MST)-based cluster routing method. An MST is constructed to determine the connection topology. Then the MST edges are sequentially routed using the A* search algorithm [26]. Point-to-point, point-to-path, and path-to-path A* search algorithms are used to enhance routability and reduce the total channel length. When there are failed MST edges, the corresponding cluster will be de-clustered into smaller ones.

Escape routing for control pins: When the valves within the clusters are connected internally, we start the escape routing process to connect the clusters to the control pins. The minimum cost flow formulation is proposed to solve the escape routing problem, as described in Section 5.

De-clustering and path rip-up: During escape routing, certain valves or clusters may fail to be routed to the control pin. In that case, the blocking paths are ripped up, and another round of escape routing will be started. During rip-up and rerouting process, the clusters with length-matching constraint can also be ripped up to improve the routing completion rate, but at higher rip-up cost. The rip-up and rerouting process is iterated until all the valves are successfully routed or a predefined iteration threshold is reached.

Path detouring for length-matching: When the clusters are routed to the control pins, we start the path detouring process for the clusters with length-matching constraint. We experimentally verify that for single layer routing problem, in most cases path detouring is easier than improving the routing completion rate. Therefore, path detouring process is performed as the final step. For effective path detouring, a minimum-length bounded routing method is presented in Section 6.

4. LENGTH-MATCHING AWARE CLUSTER ROUTING

To address the length-matching constraint for clusters with more than two valves, we present the following methods for solving the problem: (1) DME-based candidate Steiner tree construction, (2) maximum weight clique problem formulation (MWCP) for Steiner tree selection and integer linear programming (ILP) based solution, and (3) negotiation-based rip-up and reroute for improving routability. For clusters with two valves, the edge connecting the two valves is directly obtained without performing the DME-based algorithm.

4.1 DME-Based Steiner Tree Construction

For a given set of valves, the deferred-merge embedding (DME) algorithm is adopted for computing the candidate Steiner trees to satisfy the length-matching constraint [24]. The DME algorithm was originally presented to embed a given connection topology for a clock tree with zero skew and minimized total wire length, where the connection topology can be computed using the balanced bipartition (BB) approach [24]. We adopt the same BB algorithm, which recursively bipartitions the given set of valves into two subsets with minimized sum of diameters of the subsets. In the BB algorithm, the capacitance for each sink (valve) is equally set to 1, such that the computed connection topology will be a balanced binary tree when there is a even number of valves in the cluster.

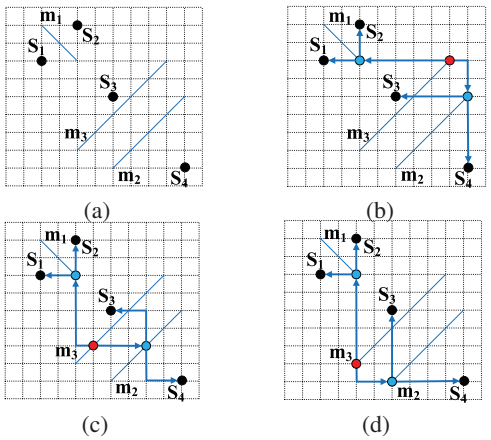

Figure 3: Candidate Steiner trees computed by the DME algorithm. (a) merging segments for the sinks, (b)-(d) candidate Steiner tree solutions with length-matching constraint.

When the connection topology is computed from BB, the DME algorithm will start with a bottom-up *merging segment* computation phase and a top-down *merging node* embedding phase. During the merging segment computation, the Manhattan distances to the sinks are recorded such that the computed merging segments are of equal Manhattan distance to the corresponding sinks.

LEMMA 1. *When the Manhattan distance between two on-grid nodes, either internal or leaf nodes (i.e., valves), are of odd length, the merging segment of the two nodes is off-grid.*

Two issues need to be addressed in the original DME algorithm: (1) because the routing process is performed on the uniform routing grids, which are partitioned according to the minimum channel width and spacing design rule, the rounding error is unavoidable, and (2) there are routing blockages which block the merging segments and merging nodes. The above two issues are addressed during the top-down merging node embedding phase. During the merging node embedding, a valid merging node avoiding blockages is searched around the loops encircling the given merging segment with increasing radius. The encircling loop expands outside until a valid merging node is found or reaching the biochip's boundary. This method possibly introduces the delta distance from the computed merging node to the two child nodes. However, the delta distance can be eliminated by detouring the routing paths afterwards.

By selecting different merging nodes on the merging segments, multiple candidate Steiner trees can be computed. Figure 3 shows an example of the partial candiate Steiner tree solutions for a cluster of four valves (S_1, S_2, S_3, and S_4). In Figure 3 (a), merging segments m_1, m_2 and m_3 are computed during the bottom-up merging segment computation phase. Then during the top-down merging node embedding phase, difference choices of the merging nodes will result in different Steiner tree solutions, each of which satisfies the length-matching constraint (see Figure 3 (b), (c), and (d)). We compute different candidate Steiner trees for simultaneous selection with a global view for optimizing the routability and length-matching objective.

4.2 Candidate Steiner Tree Selection

When the candidate Steiner trees are computed for all the clusters, we start to determine one Steiner tree solution for each cluster. There are two key factors in choosing the candidate Steiner tree: (1) the length-matching objective, i.e., the estimated length mismatch in the Steiner tree due to rounding error and obstacle avoidance,

and (2) routability, i.e., routing conflicts between Steiner trees of different clusters are expected to be as few as possible.

DEFINITION 5 (*Full path*). *The* full path P_{F_i} *for valve* v_i *is defined as the sequence of paths from* v_i *upward to the root of the DME-computed Steiner tree.*

According to the above definition, a full path is a set of the routing paths. Assume for valve v_i, the corresponding full path is $P_{F_i} = \{p_{i,1}, p_{i,2}, \ldots, p_{i,n}\}$. Then the length of the full path is computed as $l(P_{F_i}) = \sum_{p_{i,j} \in P_{F_i}} l(p_{i,j})$. Here, $l(p_{i,j})$ denotes the length of path $p_{i,j}$. In the candidate Steiner tree selection process, the path length is estimated by Manhattan distance. The length mismatch of a candidate Steiner tree is defined as

$$\Delta L = max\{l(P_{F_i}) | i \in [1,n]\} - min\{l(P_{F_i}) | i \in [1,n]\} \quad (1)$$

Then the length mismatch cost for candidate Steiner tree T_j is defined as

$$C_{m_j} = -\lambda \cdot \frac{\Delta L_j}{max\{\Delta L_k | k \in [1,N]\}} \quad (2)$$

where N represents the total number of candidate Steiner trees for all the clusters, and λ is a user-defined parameter.

Given two candidate Steiner trees T_i and T_j for different clusters, the overlap cost between them is computed as

$$C_{o_{i,j}} = -(1-\lambda) \cdot \sum_{e_l \in T_i} \sum_{e_m \in T_j} ol_{cost}(e_l, e_m) \quad (3)$$

where e_l and e_m are the edges of Steiner trees T_i and T_j, respectively. In the experiments, λ is set to be 0.1, which gives higher priority for overlap cost over length mismatch in Equation (2). This is because routability of the length-matching clusters on a single layer is more challenging than the path detouring for length-matching constraint. $ol_{cost}(e_l, e_m)$ denotes the overlap cost between edges e_l and e_m, which is computed as

$$ol_{cost}(e_l, e_m) = \frac{area(overlap(bb(e_l), bb(e_m)))}{min\{area(bb(e_l)), area(bb(e_m))\}} \quad (4)$$

where $bb(e)$ gives the bounding box of an edge e, $overlap(b_1, b_2)$ computes the overlap between two boxes b_1 and b_2, and $area(b)$ computes the area of box b.

Input: Set of tree edges B and the coordinates of the corresponding nodes.
Output: The set of routing paths P for B.

```
1  Initialize history cost C_h for the routing grids;
2  Construct ObsMap for routing obstacles on the routing grids;
3  Set flag done ← false;
4  Set counter r ← 0;
5  while done ≠ true do
6      Set done ← true;
7      for i = 1 to |B| do
8          Routing for edge i with C_h and ObsMap;
9          if Routing successful then
10             Insert the routing path into P;
11             Set the routing path as obstacles in ObsMap;
12         else
13             Set done ← false;
14     Set counter r ← r+1;
15     if r ≥ γ then
16         break;
17     if done ≠ true then
18         Update C_h for all the grids along the paths in P;
19         Remove all the paths in P and reset the obstacle flags in ObsMap.
```

Algorithm 1: Negotiation-based routing.

With the length mismatch cost and overlap cost are computed, the candidate Steiner tree selection problem is formulated as maximum weight clique problem in graph $G(V,E)$ as follows: (1) For each candidate Steiner tree T_i, add a node v_i into V with weight C_{m_i}; (2) For each pair of Steiner trees T_i and T_j from different clusters, add an edge $e_{i,j} = (n_i, n_j)$ into E with weight $C_{o_{i,j}}$; (3) When the maximum weight clique is computed, the nodes in the clique correspond to the selected Steiner trees.

Different methods can be used to solve the maximum weight clique problem [25]. We have implemented the graph-based algorithm, ILP-based method, and unconstrained quadratic programming based method. Finally, we determined to adopt the ILP-based method, which gives the best performance.

4.3 Negotiation-Based Routing

When the Steiner trees are computed, the negotiation-based routing method is performed to route all the clusters with the length-matching constraint. Algorithm 1 shows the iterative routing algorithm for the tree edges based on the negotiation strategy [27]. Different from the original approach, which considers the negotiated congestion for global routing, we present the negotiation-based detailed routing, which directly considers the routability of the routing grids. Therefore, a different cost function for history cost is defined for each routing grid g on the routing paths as

$$C_h(g)^{r+1} = b_g + \alpha \cdot C_h(g)^r \quad (5)$$

where $C_h(g)^{r+1}$ is current history cost of routing grid g for iteration $r+1$, b_g is the base history cost, $C_h(g)^r$ is the history cost in iteration r, and α is a user-define parameter. In the implementation, b_g is set to be 1.0, and α is set to be 0.1.

In Algorithm 1, the history costs for the grids are first initialized to be 0 in *Step* 1. In *Step* 2, the obstacle map *ObsMap*, which is a two-dimensional array of boolean values, is constructed for the routing grids. In *Steps* $3-4$, a boolean flag *done* and an integer counter r are initialized for the following negotiated iterative routing process. Then in *Steps* $7-13$, all the edges are routed one by one using A* search algorithm. If an edge is successfully routed, the routing path will be inserted into P, and the routing grids along the routing path will be set as obstacles in *ObsMap*. Otherwise, the iteration flag will be reset as $false$ for further iteration. In *Steps* $14-16$, the iteration counter r is increased by 1. If the number of iterations exceeds the user-defined threshold γ, the while-loop will be terminated. In that case, the DME tree needs to be reconstructed, and even the valve positions may need to be re-designed. In the implementation, γ is set to be 10. In *Steps* $17-19$, when the number of iterations does not exceed the threshold and there are failed edges, the history cost for all the routing grids along the routed paths will be updated to a larger value using Equation (5). And all the routing paths are removed from P with the obstacle flag reset to $false$ in *ObsMap*. In the next iteration, those routing grids with larger history cost are less likely to be occupied by the routing paths unless there are no alternative routing solutions. Therefore, the negotiation-based routing approach greatly improves routability. Assume the largest number of grids covered by the bounding boxes of the edges is $m \times n$. In the worst case, the time complexity of the routing process is $O(m \cdot n)$. So the time complexity of Algorithm 1 is $O(m \cdot n \cdot |B| \cdot \gamma)$ in the worst case.

5. ESCAPE ROUTING

When the clusters are routed using different methods as presented above, the final routing solution is obtained by escape routing to connect the routed clusters with the control pins: (1) For clusters of multiple valves with the length-matching constraint, the Steiner tree roots are used for routing; (2) For clusters of two valves with the length-matching constraint, the middle points on the paths are used for routing; (3) For clusters without the length-matching constraint, any point on the routed paths can be used for routing. The number of needed control pins is equal to the total number of clusters. The objective of the escape routing problem is to select proper control pins and compute the routing paths for minimizing the total channel length and enhancing routing completion rate. The escape routing problem is formulated as the following minimum cost flow problem:

- **Objective:** Minimize $\sum l_{i,j} \cdot f_{i,j} - \beta \cdot (\sum x_j + \sum x_q)$
- **Subject to:**

$$\sum f_{j,k} \geq x_j \qquad \forall\, g_j \in G_c \cup G_s \quad (6)$$

$$\sum f_{i,j} = 0 \qquad \forall\, g_j \in G_c \cup G_s \quad (7)$$

$$\sum f_{i,j} + \sum f_{j,k} = 0 \qquad \forall\, g_j \in G_o \cup G_b \quad (8)$$

$$\sum f_{j,k} - \sum f_{i,j} = 0 \qquad \forall g_j \in G_r \tag{9}$$

$$\sum_{g_j \in C_q} \sum_k f_{j,k} \geq x_q \qquad \forall C_q \in C \tag{10}$$

$$\sum_{g_j \in C_q} \sum_k f_{i,j} = 0 \qquad \forall C_q \in C \tag{11}$$

$$\sum f_{j,k} + \sum f_{i,j} \leq 2 \qquad \forall g_j \in G_r \tag{12}$$

where $f_{i,j}$ ($f_{i,j} \geq 0$) is a floating variable denoting the flow value on the edge from routing grid g_i to g_j, $l_{i,j}$ is a constant value denoting the grid length from g_i to g_j, x_j and x_q ($0 \leq x_j, x_q \leq 1$) are floating variables related to the number of paths successfully routed to the control pins, and β is a user-define parameter to make the second item ($\sum x_j + \sum x_q$) dominate the first one ($\sum l_{i,j} \cdot f_{i,j}$). The objective is to simultaneously maximize the number of successfully routed paths to control pins and minimize the total channel length. Constraint (6) restricts the total flow outward routing grid g_j by x_j for all the root/middle points computed for the length-matching constraint (G_c) and the single valves (G_s) connecting directly to the control pin. Constraint (7) specifies that the total flow inward routing grid g_j is 0 for $g_j \in G_c \cup G_s$. Constraint (8) specifies that all the inward flows and outward flows are 0 from routing grid g_j if g_j is an obstacle (G_o) or a boundary point that is not a control pin (G_b). Constraint (9) specifies the flow conservation constraint for all the ordinary routing grids (G_r). Similar to Constraint (6), Constraint (10) restricts the sum of all the outward flows by x_q for all the routing grids of the routing paths for any cluster C_q. Therefore, by maximizing $\sum x_j + \sum x_q$, the number of successfully routed paths is maximized. Constraint (11) specifies that each inward flow is 0 for all the routing grids of the routing paths for any cluster C_q. Constraint (12) avoids crossings between routing paths by specifying that the sum of all the inward and outward flows for routing grid g_j is less than or equal to 2.

THEOREM 1. *The presented minimum cost flow formulation for escape routing obtains optimal routing solution with minimized total cost. (Proof is omitted for brevity.)*

6. DETOURING FOR LENGTH MATCHING

When all the clusters are routed to the control pins, we start to detour the paths of the clusters for the length-matching constraint.

DEFINITION 6 (*Path Sequence*). *The sequence of routing paths $(p_{i,1}, p_{i,2}, \ldots, p_{i,n})$ along the full path P_{F_i} is called a path sequence, denoted as P_{s_i}, if the following condition is satisfied: $p_{i,j}$ is before $p_{i,k}$ in the sequence if and only if $p_{i,j}$ is closer to valve v_i than $p_{i,k}$ along the full path P_{F_i} to the tree root.*

Because the paths at the beginning of the path sequences are closer to the valves in the Steiner tree, i.e., closer to the tree sinks, changes to those paths do not affect other disjoint full paths. Therefore, we prefer to rip-up and reroute these paths for quick convergence in length-matching objective.

Algorithm 2 shows iterative rip-up and reroute algorithm to detour the shorter full paths for the length-matching constraint. First, function *checkEqual* is called to check the length-matching constraint. All the short full paths P_{F_s} that need to be detoured are computed, and the maximum length *maxL* of all the full paths is computed. If the lengths of all the full paths are within the range $[maxL - \delta, maxL]$, then *equal* is set to be *true*. Otherwise, *equal* is set to be *false*. Next, the iteration loop is entered with the counter r increased by 1 for each round. In the implementation, θ is set to be 10. Then the shorter full paths are sequentially detoured targeting the range $[maxL - \delta, maxL]$. For each short full path, the corresponding path sequence is obtained and detoured in that order. We present the minimum bounded-length routing algorithm for path detouring, which is designed to compute a path with length not less than the target length L_t. The minimum bounded-length routing algorithm is based on the modified A* search algorithm [26]. The key differences of the presented algorithm from classic A* algorithm are as follows: (1) the G value of current grid records the path length from the source grid, and can only be updated when the value is increased (rather than decreased in traditional method), and (2) the F value of current grid not only includes the sum of G and H values, but also include the penalty cost when the estimated total length is less than the given length bound.

Input: Set of routing paths P corresponding to the Steiner tree T, and the length-matching threshold δ.
Output: The set of detoured routing paths P_d satisfying the length-matching constraint.

```
1  Call function (equal, maxL, P_Fs) ← checkEqual(P, T, δ) ;
2  Set counter r ← 0;
3  while equal ≠ true do
4      if r ≥ θ then
5          break;
6      Set counter r ← r + 1;
7      Initialize vector of detouring flag F_d as false;
8      for i = 1 to |P_Fs| do
9          Set flag success ← false;
10         Compute the target detour length L_t from maxL and δ;
11         Obtain the path sequence P_si;
12         for j = 1 to |P_si| do
13             if F_d[P_si(j)] then
14                 Set success ← true;
15                 break;
16             Minimum bounded-length routing P_si(j) with L_t;
17             if Routing successful then
18                 Add the routed path to P_d;
19                 Set F_d[P_si(j)] ← true;
20                 Set success ← true;
21                 break;
22         if success == false then
23             Restore P_d to the original paths before detouring;
24             return;
25     Call function (equal, maxL, P_Fs) ← checkEqual(P, T, δ).
```

Algorithm 2: Path detouring algorithm for length-matching constraint.

Table 1: Design parameters.

Design	Size	#Valves	#Control pin	#Obs
Chip1	179×413	176	556	1800
Chip2	231×265	56	495	1863
S1	12×12	5	14	9
S2	22×22	10	40	54
S3	52×52	15	93	0
S4	72×72	20	139	27
S5	152×152	40	306	135

Assume the total number of routing grids is $m \times n$ in Algorithm 2. The time complexity of the path detouring algorithm is dominated by the minimum-length bounded routing algorithm, which runs in $O(m \cdot n)$ in the worst case. Then the worst-case time complexity of Algorithm 2 is $O(m \cdot n \cdot |P_{F_s}| \cdot |P_{s_i}| \cdot \theta)$, which is polynomial and very efficient in real applications.

7. COMPUTATIONAL SIMULATION RESULTS

We have implemented our practical control layer routing system PACOR with length-matching constraint in C++. PACOR is tested on a 2.13GHz Intel Xeon Linux server with 8 cores and 37GB memory. The Gurobi optimizer is used to solve the ILP and linear programming problems [28]. Table 1 shows the details of the benchmarks, where "Design" gives the names of the benchmarks including two real biochips and five synthesized testcases, "Size" gives the sizes of the chips represented in routing grids, "#Valves" gives the number of valves to be routed, "#CP" gives the number of candidate control pins, and "#Obs" gives the number of obstructed routing grids. From Table 1, there are only a few routing obstacles in the control layer for real biochips. In the experiments, the length-matching threshold value δ is set to be 1.

Table 2: Computational simulation.

Design	#Clusters	#Matched Clusters			Total matched channel length			Total channel length			Runtime (s)		
		w/o Sel	Detour First	PACOR	w/o Sel	Detour First	PACOR	w/o Sel	Detour First	PACOR	w/o Sel	Detour First	PACOR
Chip1	40	13	20	24	1422	1525	2412	11011	9495	10929	305.78	376.5	201.26
Chip2	22	22	22	22	1262	1262	1262	3612	3612	3612	31.97	35.55	35.14
S1	2	2	2	2	28	28	28	36	36	36	0.02	0.01	0.01
S2	2	1	1	1	71	40	40	168	109	105	0.18	0.18	0.11
S3	5	4	4	4	264	161	161	425	277	277	1.35	1.36	1.3
S4	7	6	6	6	1371	595	531	1547	809	888	2.98	1.45	1.39
S5	13	3	4	5	293	830	1065	2945	3153	3110	58.41	51.15	62.65
Avg.		0.80	0.92	1	0.86	0.81	1	1.04	0.92	1	1.33	1.54	1

Table 2 shows the computational simulation results, where the columns under "#Clusters" give total number of clusters with at least two valves.We aim to route as many clusters as possible under the length-matching constraint. If a cluster fails to be routed with length-matching constraint, it will be routed as ordinary cluster using the MST-based cluster routing method. "#Matched Clusters" gives total number of successfully routed clusters satisfying the length-matching threshold value δ. Total matched channel length and total channel length are reported, respectively. Because PACOR is the first work to address the control-layer routing problem with length-matching constraint, there is no comparison target. So we perform self-comparison to verify the effectiveness. As all the methods obtain 100% routing completion rate, the routability information is not reported. In the table, "w/o Sel" gives the results without the candidate Steiner tree selection strategy as described in Section 4.2, "Detour First" is to detour the paths for length-matching constraint immediately after the negotiation-based routing process (Section 4.3). From the results, PACOR, which is equipped with all the effective modules of the overall flow in Figure 2, obtains the best results in terms of the number of matched clusters. Without the candidate Steiner tree selection strategy, the results become worse with decreased number of matched clusters, as well as increased wire length. It seems that the detour-first method obtains improved total wire length than our presented final-stage detouring strategy. However, it is at the cost of decreased number of matched clusters. Moreover, the detour-first method obtains fairly good solution due to our network-flow based escape routing and rip-up and rerouting methods, which rip up long detoured nets for improved routing completion rate. All the three methods obtain same solution quality on Chip2. This is because Chip2 has abundant routing resource and there are only clusters with two valves, which are easily matched and routed. Finally, PACOR obtains the best runtime because the presented strategies work well as a practical flow, which reduce the number of rip-up and rerouting iterations. Considering the notable difficulty in length-matching routing on a single layer, the results show PACOR is very effective and efficient.

8. CONCLUSION

We have proposed a practical control-layer routing flow, called PACOR, for flow-based biochips. PACOR practically and effectively addresses the length-matching routing constraint and improves routing completion rate. Computational simulation results show that all valves are successfully routed to control pins with significant portions of clusters routed with length-matching constraint, confirming the effectiveness of PACOR.

9. REFERENCES

[1] F. K. Balagadde, L. You, C. L. Hansen, F. H. Arnold, and S. R. Quake, "Long-Term Monitoring of Bacteria Undergoing Programmed Population Control in a Microchemostat," *Science*, vol. 309, no. 5731, pp. 137-140, 2005.

[2] G. M. Whitesides, "The Origins and the Future of Microfluidics," *Nature*, vol. 442, no. 7101, pp. 368-373, 2006.

[3] P. Yager, T. Edwards, E. Fu, et al., "Microfluidic Diagnostic Technologies for Global Public Health," *Nature*, vol. 442, no. 7101, pp. 412-418, 2006.

[4] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic Large-Scale Integration," *Science*, vol. 298, no. 5593, pp. 580-584, 2002.

[5] D. Mark, S. Haeberle, G. Roth, et al. "Microfluidic Lab-on-a-Chip Platforms: Requirements, Characteristics and Applications," *Chemical Society Reviews*, vol. 39, no. 3, pp. 1153-1182, 2010.

[6] J. A. Rogers and R. G. Nuzzo, "Recent Progress in Soft Lithography," *Materials Today*, vol. 8, no. 2, pp. 50-56, 2005.

[7] V. Studer, G. Hang, A. Pandolfi, et al., "Scaling Properties of a Low-actuation Pressure Microfluidic Valve," *Journal of Applied Physics*, vol. 95, no. 1, pp. 393-398, 2004.

[8] W. H. Minhass, P. Pop, J. Madsen, and T.-Y. Ho, "Control Synthesis for the Flow-based Microfluidic Large-Scale Integration Biochips," *Proc. ASP-DAC*, 2013, pp. 205-212.

[9] T. M. Squires and S. R. Quake, "Microfluidics: Fluid Physics at the Nanoliter Scale," *Reviews of Modern Physics*, vol. 77, pp. 977-1026, 2005.

[10] M. A. Unger, H.-P. Chou, T. Thorsen, et al., "Monolithic Microfabricated Valves and Pumps by Multilayer Soft Lithography," *Science*, vol. 288, no. 5463, pp. 113-116, 2000.

[11] I. E. Araci and S. R. Quake, "Microfluidic Very Large Scale Integration (mVLSI) with Integrated Micromechanical Valves," *Lab Chip*, vol. 12, no. 16, p. 2803, 2012.

[12] Y. C. Lim, A. Z. Kouzani, and W. Duan, "Lab-on-a-Chip: A Component View," *Microsystem Technologies*, vol. 16, no. 12, pp. 1995-2015, 2010.

[13] W. H. Minhass, P. Pop, and J. Madsen, "System-Level Modeling and Synthesis of Flow-Based Microfluidic Biochips," *Proc. International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2011, pp. 225-233.

[14] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga, "Architectural Synthesis of Flow-Based Microfluidic Large-Scale Integration Biochips," *Proc. International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2012, pp. 181-190.

[15] F. SU and K. Chakrabarty, "Architectural-Level Synthesis of Digital Microfluidics-Based Biochips," *Proc. ICCAD*, 2004, pp. 223-228.

[16] M. Cho and D. Z. Pan, "A High-Performance Droplet Routing Algorithm for Digital Microfluidic Biochips," *IEEE Trans. on CAD*, vol. 27, no. 10, pp. 1714-1724, 2008.

[17] F. SU and K. Chakrabarty, "Unified High-Level Synthesis and Module Placement for Defect-Tolerant Microfluidic Biochips," *Proc. DAC*, 2005, pp. 825-830.

[18] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Placement of Defect-Tolerant Digital Microfluidic Biochips Using the T-Tree Formulation," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 3, no. 3, Artical No. 13, 2007.

[19] T.-W. Huang and T.-Y. Ho, "A Two-Stage Integer Linear Programming-Based Droplet Routing Algorithm for Pin-Constrained Digital Microfluidic Biochips," *IEEE Trans. on CAD*, vol. 30, no. 2, pp. 215-228, 2011.

[20] W. H. Minhass, P. Pop, and J. Madsen, "Synthesis of Biochemical Applications on Flow-Based Microfluidic Biochips Using Constraint Programming," *Proc. Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, 2012 , pp. 37-41.

[21] C.-X. Lin, C.-H. Liu, I.-C. Chen, D. T. Lee, and T.-Y. Ho, "An Efficient Bi-criteria Flow Channel Routing Algorithm for Flow-Based Microfluidic Biochips," *Proc. DAC*, 2014, pp. 1-6.

[22] N. Amin, W. Thies, and S. Amarasinghe, "Computer-Aided Design for Microfluidic Chips Based on Multilayer Soft Lithography," *Proc. ICCD*, 2009, pp. 2-9.

[23] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-Layer Optimization for Flow-Based mVLSI Microfluidic Biochips," *Proc. International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2014.

[24] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, and A. B. Kahng, "Zero Skew Clock Routing with Minimum Wirelength," *IEEE Trans. on Circuits and Systems II*, vol. 39, no. 11, pp. 799-814, 1992.

[25] B. Alidaee, F. Glover, G. Kochenberger, and H. Wang, "Solving the Maximum Edge Weight Clique Problem via Unconstrained Quadratic Programming," *European Journal of Operational Research*, vol. 181, no. 2, pp. 592-597, 2006.

[26] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100-107, 1968.

[27] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," *Proc. ACM Symposium on Field-Programmable Gate Arrays*, 1995, pp. 111-117.

[28] Gurobi Optimizer. http://www.gurobi.com/.

[29] M. R. Garey, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-completeness", Freeman, New York, 1979, pp. 53-56.