Physical Synthesis of Flow-Based Microfluidic Biochips Considering Distributed Channel Storage

Zhisheng Chen[†], Xing Huang[ŧ], Wenzhong Guo[†], Bing Li[‡], Tsung-Yi Ho[ŧ], and Ulf Schlichtmann[‡]

[†]College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
[ŧ]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[‡]Chair of Electronic Design Automation, Technical University of Munich, Munich, Germany
Email: xing.huang1010@gmail.com

***Abstract*—Flow-based microfluidic biochips (FBMBs) have attracted much attention over the past decade. On such a micrometer-scale platform, various biochemical applications, also called bioassays, can be processed concurrently and automatically. To improve execution efficiency and reduce fabrication cost, a distributed channel-storage architecture (DCSA) can be implemented on this platform, where fluid samples can be cached temporarily in flow channels close to components. Although DCSA can improve the execution efficiency of FBMBs significantly, it requires a careful arrangement of fluid samples to enable the channels to fulfill the dual functions of transportation and caching. In this paper, we formulate the first flow-layer physical design problem considering DCSA, and propose a top-down synthesis algorithm to generate efficient solutions considering execution efficiency, washing, and resource usage simultaneously. Experimental results demonstrate that the proposed algorithm leads to a shorter execution time, less flow-channel length, and a higher efficiency of on-chip resource utilization for biochemical applications compared with a direct approach to incorporate distributed storage into existing frameworks.**

I. Introduction

Flow-based microfluidic biochips (FBMBs) are increasingly used in various fields to execute bioassays such as DNA analysis [1], point-of-care diagnostics [2] and drug discovery [3]. Hundreds of such assays can be integrated now into a micrometer-scale microfluidic platform and automatically completed based on a pre-customized assay plan [4].

Conventional FBMBs utilize dedicated storage units to cache intermediate fluids generated by operations in bioassays. This concept of dedicated storage, however, suffers from several limitations: 1) constrained capacity of the storage unit; 2) limited access bandwidth at input/output ports of the storage unit; 3) large area on the chip occupied by the storage unit. To overcome these problems, distributed channel-storage architecture (DCSA) [5] can be used to cache fluids temporarily in flow channels instead of in a dedicated storage unit. Consequently, biochips with such a new architecture not only improve the execution efficiency of bioassays, but also reduce fabrication costs.

Since FBMBs with the distributed storage architecture should guarantee that fluids can be stored safely in transportation channels and should not be contaminated during the storage time, physical design of such chips should be performed carefully to consider the storage requirements. Over the past few years, a number of methods have been proposed for the physical design of FBMBs [6]–[8]. In [6] a top-down design flow is proposed to generate a biochip architecture while minimizing the execution time of the bioassay. In [7] a close-to-optimal placement/routing algorithm is proposed to solve the physical design problem using a SAT formulation. Moreover, a top-down physical synthesis flow considering valve-switching minimization has been introduced in [8]. All these methods, however, ignore the combined physical design and storage problem and thus suffer a high efficiency degradation when fluids are cached.

In view of the advantages of FBMBs with DCSA, we focus on the physical design of such chips in this paper. The key contributions of this paper are summarized as follows.

- In DCSA, dedicated storage units are completely removed. Hence, transportation paths, caching locations, as well as the corresponding time slots of fluids in flow channels have to be determined in the design process. In this paper, we formulate the first practical flow-layer physical design problem for DCSA-based biochips to consider these requirements together.
- The dual functions of flow channels in DCSA, i.e., transportation and caching, require a careful arrangement of resource usage to avoid conflicts. Accordingly, we introduce a top-down synthesis algorithm that can optimize the execution time of the bioassay, channel length, and resource utilization simultaneously.
- Reducing wash time for contamination removal in DCSA becomes more important and complex, since flow channels are used both for transportation and caching. Thus, wash time optimization is also considered in our design flow for FBMBs.
- Experiments on multiple benchmarks confirm that the proposed method produces a shorter execution time, shorter channel length, and higher efficiency of resource utilization compared to a direct approach to incorporate DCSA into existing frameworks.

The remainder of this paper is organized as follows. Section II presents an overview of DCSA-based biochips and analyzes the corresponding design challenges. Section III provides the problem formulation. Section IV describes the proposed synthesis algorithms in detail. Section V reports the experimental results. Finally, conclusions are drawn in Section VI.

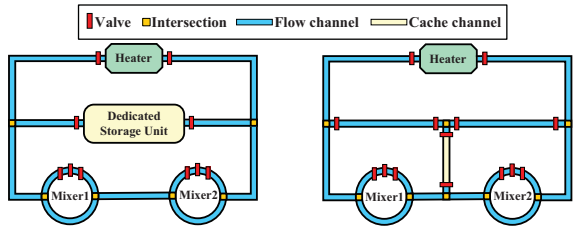
II. Background and Design Challenges with DCSA

In this section, we first explain the background of FBMBs and DCSA. We then discuss the estimation of wash time for contamination removal. Finally, we analyze the challenges involved in DCSA-based physical design.

A. Flow-Based Microfluidic Biochips and DCSA

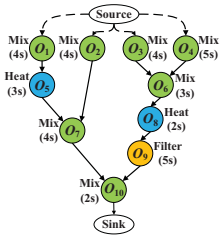
FBMBs utilize a flow-channel network to transport and manipulate fluids such as samples and reagents. Each channel is connected to a flow port, through which external pressure can be injected to push the movement of fluids [8]. During executing

978-3-9819263-2-3/DATE19/©2019 EDAA



(a) (b)

Fig. 1: Conventional FBMBs and DCSA. (a) FBMB with a dedicated storage unit. (b) The DCSA-based layout.



Biochemical operation			
Diffusion coefficient (cm^2/s)/Estimated wash time (s)			
O_1	2×10^{-9}/10	O_6	5×10^{-6}/2
O_2	1×10^{-7}/3	O_7	5×10^{-6}/2
O_3	5×10^{-8}/6	O_8	1×10^{-6}/1
O_4	5×10^{-6}/2	O_9	5×10^{-5}/0.5
O_5	1×10^{-6}/1	O_{10}	1×10^{-7}/3

(a) (b)

Fig. 2: (a) Sequencing graph of a bioassay. (b) Diffusion coefficients and the corresponding wash time of fluids generated in the bioassay described in (a).

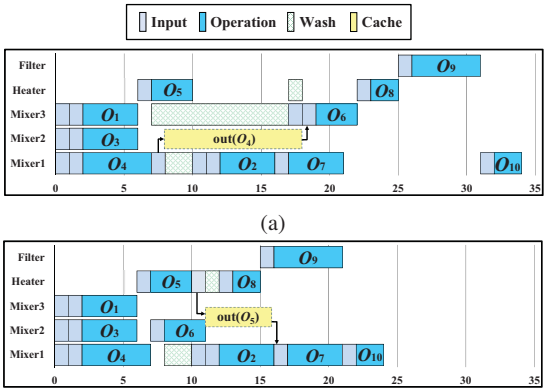
operations of a bioassay, intermediate results are usually stored in a dedicated storage unit, as shown in Fig. 1(a). At the entrance and the exit of the storage unit, multiplexer-like control valves are needed to address the expected storage cell. This port multiplexing allows only one fluid to enter or leave the storage unit and thus limits its bandwidth. Consequently, the execution efficiency of the bioassay is bottlenecked by the dedicated storage.

Unlike electronic systems, FBMBs actually allow fluids to be stored inside a segment of transportation channel anywhere in the chip. During a given period, if a channel segment is not used for transportation, the fluid residing inside it can be considered as stored or cached. Taking advantage of this characteristic, a DCSA can be implemented to use flow channels as temporary storage segments. Thus, the same channels can switch between the roles of transportation and storage as required. Since the bandwidth limitation of dedicated storage is removed from this architecture, the efficiency of the system can be improved significantly [5]. The DCSA-based biochip corresponding to the traditional design in Fig. 1(a) is shown in Fig. 1(b), where the highlighted channel in the middle stores a fluid sample temporarily.

In order to map a bioassay to a DSCA-based biochip, synthesis techniques are needed to schedule and bind the operations in the bioassay to on-chip resources. In addition, distributed storage needs to be allocated efficiently to reduce the overall execution time of the bioassay. Note that, the results of these steps should be implemented in the physical design properly, since distributed storage interacts with fluid transportation closely.

B. Wash Time

In FBMBs, a channel must be cleaned by washing after being used to avoid contamination in the following transportation of fluids. For DCSA-based biochips, this is even more important because distributed storage, fluid transportation, and washing must be coordinated properly to avoid contamination and conflicts.



(a)

(b)

Fig. 3: Two scheduling schemes for the bioassay described in Fig. 2(a). (a) A scheduling with longer execution time and lower on-chip resource utilization. (b) An improved scheduling with shorter execution time and higher on-chip resource utilization.

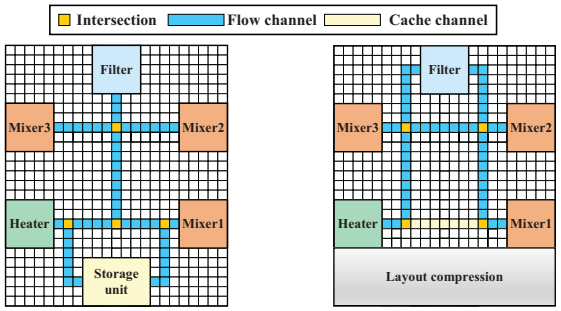
Washing channels is performed by injecting a buffer flow with a given pressure for a given time. Usually, wash time is affected by four factors, i.e., the channel length, the channel width, the applied pressure of buffer flow, and the diffusion coefficient of contaminant. In practice, it has been shown by the experiments in [9] that the diffusion coefficient dominates wash time, i.e., the other three factors can be ignored accordingly in estimating wash time. In particular, a lower diffusion coefficient implies a longer wash time of the corresponding contaminant. For example, small molecules such as a lysis buffer, have a high diffusion coefficient (typically 10^{-5} cm^2/s) and short wash time (typically 0.2 s) [10]. In contrast, cells such as tobacco mosaic virus, have a low diffusion coefficient (typically 5×10^{-8} cm^2/s) and long wash time (typically 6 s) [10].

C. Challenges of DCSA-Based Design

Let C be a set of allocated components. We model a biochemical application as a directed acyclic sequencing graph $G(O, E)$. Fig. 2(a) shows the sequencing graph of a bioassay, where each vertex $o_i \in O$ represents an operation and is associated with a parameter to indicate the corresponding execution time. Each edge in E defines the dependency between operations. The output of an operation o_i is denoted by $out(o_i)$. The design automation framework for DCSA-based biochips consists of two steps: 1) resource binding and scheduling and 2) placement and routing.

1) Requirements of DCAS in Scheduling and Binding: The resource binding and scheduling stage has two major objectives: 1) find a binding function Φ: $O \rightarrow C$, through which each operation can be bound to a specific component, and 2) find a scheduling scheme such that all the operations can be executed efficiently while satisfying the aforementioned dependencies. Note that the caching requirements of fluids in flow channels should also be determined in the scheduling procedure.

Take the bioassay shown in Fig. 2(a) as an example. The diffusion coefficients of fluids of different operations and their corresponding wash times are shown in Fig. 2(b). Fig. 3 shows two resource binding and scheduling schemes, where five components are allocated to execute the bioassay. In Fig. 3(a), o_6 is bound to $Mixer3$ and it takes 10 s to wash the residue left by o_1. Since $Mixer1$ is also allocated to execute other operations,



(a) (b)

Fig. 4: Two placement and routing solutions for the scheduling shown in Fig. 3(b). (a) Solution generated by traditional physical design methods. (b) A DCSA-based solution.

e.g., o_2 and o_7, $out(o_4)$ needs to be cached in flow channel for 10 s, thereby prolonging the execution of subsequent operations. The total execution time in Fig. 3(a) is 37 s. In Fig. 3(b), since o_6 is bound to $Mixer2$, $out(o_4)$ only needs to be transported from $Mixer1$ to $Mixer2$, and $out(o_3)$ can be directly cached in $Mixer2$ waiting for the execution of o_6. Moreover, since washing for the residue left in $Mixer1$ and the corresponding flow channel connected to $Mixer2$ can be completed in only 2 s, o_2 can be executed on $Mixer1$ earlier. Consequently, the total execution time is reduced to only 24 s in Fig. 3(b).

In FBMBs, the utilization of on-chip resources, denoted by U_r, can be defined as the average usage efficiency of all components:

$$U_r = \frac{1}{|C|} \cdot \sum_{i=1}^{|C|} \frac{T_a(i)}{T_{le}(i) - T_{fs}(i)} \quad (1)$$

where $|C|$ represents the number of allocated components. $T_a(i)$ represents the actual execution time of component $c_i \in C$, which is the sum of execution times of operations bound to c_i. $T_{le}(i)$ and $T_{fs}(i)$ represent the end time of the last operation and the start time of the first operation performed on c_i, respectively. Thus, the utilization of on-chip resources of the two schemes shown in Fig. 3 are 62% and 82%, respectively.

2) Requirements of DCAS in Placement/Routing: The goal of the placement and routing stage is to assign exact locations for components while establishing efficient interconnection among them. If designed improperly, transportation conflicts may appear in an unoptimized solution in the following ways: 1) two transportation tasks compete for a channel segment at the same time; 2) fluid of a transportation task flows through a caching channel; and 3) fluid of a transportation task flows through a channel segment that needs to be washed.

Take the bioassay described in Fig. 2(a) and the scheduling schemes shown in Fig. 3 as an example again. Fig. 4(a) shows a solution generated by traditional design methods [7], where both channel intersections and channel length are minimized. However, since transportation conflicts among channels are not considered, the execution of some operations may be postponed, leading to delay and even execution failure of the complete scheduling procedure. For example, since the transportation tasks from o_1 to o_5 ($Mixer3$ to $Heater$) and from o_4 to o_6 ($Mixer1$ to $Mixer2$) share the same channel segment in Fig. 4(a), the latter has to be postponed since it takes 10 s to wash the residue left by the first task. For comparison, the solution using DCSA is shown in Fig. 4(b), which takes both transportation and caching tasks into account, so that the transportation conflicts discussed above are completely eliminated. Moreover, the chip area is effectively reduced due to the removal of dedicated storage.

III. Problem Formulation

The flow-layer physical design for DCSA-based biochips can be formulated as follows:

Given: (1) A biochemical application modeled as a sequencing graph $G(O, E)$; (2) A component library C; (3) The diffusion coefficient $D = \{d_{i,j}\}$ of fluids generated by operations in O, where $d_{i,j}$ represents the diffusion coefficient of the j-th fluid generated by operation o_i.

Find: (1) An optimized resource binding and scheduling scheme; (2) An enhanced placement and routing solution.

Objective: (1) Minimize the completion time of biochemical application; (2) Minimize the total length of flow channels; (3) Maximize the on-chip resource utilization.

IV. Algorithm Design

Through the analysis in Section II, it can be seen that traditional physical design methods cannot be directly applied to DCSA. Thus, in this section, we propose a top-down synthesis algorithm for the flow-layer physical design of DCSA-based biochips.

A. Resource Binding and Scheduling

Based on the input information described in Section III, the resource binding and scheduling problem is to find a function Φ: $O \rightarrow C$ and determine the start time of each operation in the sequencing graph. Each operation $o_i \in O$ has a corresponding ready time $t_{ready}(o_i)$, start time $t_{start}(o_i)$, and end time $t_{end}(o_i)$; $t_{ready}(o_i)$ is the time that the last father operation of o_i has been finished; $t_{start}(o_i)$ and $t_{end}(o_i)$ are determined by the scheduling scheme and the execution time of o_i. Each fluidic dependency $e_{j,k} \in E$ in the sequencing graph may be associated with a transportation or cache task. Assume that operations o_j and o_k are bound to components c_{b_1} and c_{b_2}, respectively, $out(o_j)$ then needs to be transported from c_{b_1} to c_{b_2}. Since transportation time is closely related to the lengths of flow channels, which usually remain undetermined during the scheduling process, we assume that the transportation time between components is a constant t_c defined by users [5]. If a scheduling produces a transportation time which is larger than t_c, the corresponding fluid should be cached in channels. Note that existing work assumes that the transportation of $out(o_i)$ can be avoided if o_i and o_j are bound to the same component [5]. However, this assumption is true if and only if the component is not allocated to other operations at the time interval between $t_{end}(o_i)$ and $t_{start}(o_j)$. Moreover, as discussed previously, wash time for contamination removal should be minimized in the scheduling procedure. Reducing wash time allows contaminated components and flow channels to be re-utilized as early as possible, thereby improving the utilization of on-chip resources and accelerating the completion of a bioassay.

In this part, our goal is to find an optimized resource binding and scheduling scheme for a given biochemical application in DCAS, such that the completion time of the bioassay is minimized and the utilization of on-chip resources is maximized. We present a resource-utilization-aware method by extending the well-known List Scheduling Algorithm [11] (Algorithm 1). Each operation o_i is associated with a priority value (line 1-2 in

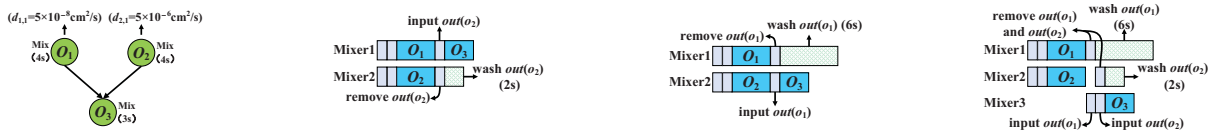


Fig. 5: Illustration of resource binding and scheduling strategy for case I. (a) A part of the sequencing graph of a bioassay. (b) Scheduling result generated by our method. (c) and (d) Scheduling results when o_3 is bound to $Mixer2$ and $Mixer3$, respectively.

Algorithm 1: Binding and scheduling for DCSA-biochips

Input: The sequencing graph $G(O, E)$, the allocated component C
Output: An optimized resource binding and scheduling scheme

1 **for** *each operation* $o_i \in O$ **do**
2 $Compute\_priority\_value(o_i)$;
3 Insert all ready operations into a priority queue Q;
4 **while** $Q \neq \emptyset$ **do**
5 $o_i \leftarrow Dequeue(Q)$;
6 **if** $O'_s \neq \emptyset$ **then**
7 Select $o_b \in O'_s$ with the lowest diffusion coefficient;
8 $Bind(o_i) \leftarrow Bind(o_b)$;
9 **else**
10 Select the qualified component $c_j \in C$ that has the earliest ready time;
11 $Bind(o_i) \leftarrow c_j$;
12 **for** *each* $e_{k,i} \in E$ **do**
13 Transport $out(o_k)$ to $Bind(o_i)$;
14 Execute(o_i);
15 **for** *each child operation* o_p *of* o_i *in* G **do**
16 **if** o_p *is ready* **then**
17 $Enqueue(o_p, Q)$;

Algorithm 1), which is specified as the length of the longest path from o_i to the sink in sequencing graph. For example, the longest path from o_1 to sink in Fig. 2(a) is $o_1 \rightarrow o_5 \rightarrow o_7 \rightarrow o_{10} \rightarrow sink$, and the priority value of o_1 is 21 if the transportation time corresponding to each edge in E is set to 2 s (i.e., $t_c = 2$). Then, operations that have been ready for execution are inserted into a queue (line 3, 15-17 in Algorithm 1) and processed in non-increasing order according to their priority values (line 5-14 in Algorithm 1). Accordingly, operations that dominate the completion time of the bioassay are executed first.

Binding and scheduling strategy: Each component $c_i \in C$ is also associated with a ready time $t_{ready}(c_i)$, which represents the time that c_i becomes available and can be computed as:

$$t_{ready}(c_i) = t_{remove}(o_j) + wash(o_j) \quad (2)$$

where o_j is the previous operation bound to c_i, $t_{remove}(o_j)$ is the time point that $out(o_j)$ is removed from c_i completely, and $wash(o_j)$ is the time for washing the residue left by o_j.

For an operation o_i just dequeued from the queue, assume that o_i has a set $O_p = \{o_1, o_2, \ldots, o_k\}$ $(k > 0)$ of father operations in the sequencing graph, and $O_s \subset O_p$ is a set of operations that has the same type as o_i. We consider how to select the best component for executing o_i. There are two cases:

- *Case I*: There exists at least one operation in O_s $(O_s \neq \emptyset)$ whose output fluid is still inside the component to which it is bound (we denote these operations by a set O'_s).
- *Case II*: The output fluids of all operations in O_s $(O_s \neq \emptyset)$ have been removed from the components to which they are bound, or O_s is an empty set.

The strategy for case I: we select the component whose output fluid has the lowest diffusion coefficient, from the components that are used for executing operations in O'_s, and bind o_i to this component (line 7-8 in Algorithm 1). This brings two main benefits: 1) since the selected component is used to execute a father operation of o_i, the time for transporting the output fluid can be eliminated; 2) since the selected component does not need to be washed, the total time for washing contaminated components is reduced significantly. Then, for each fluidic dependency $e_{k,i} \in E$ in the sequencing graph, $out(o_k)$ is scheduled and transported to the selected component (line 12-13 in Algorithm 1). Once the transportation of all the required input fluids is completed, the execution of o_i can be started immediately, i.e., the start time $t_{start}(o_i)$, and ended at time $t_{end}(o_i)$, which can be computed as $t_{start}(o_i) + t_i^j$.

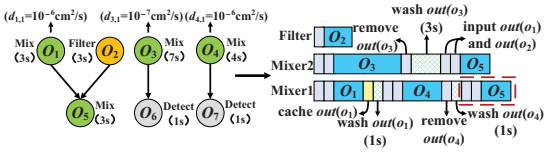


Fig. 6: Illustration of binding and scheduling strategy for case II

For example, Fig. 5(a) shows a part of the sequencing graph of a bioassay, where o_1 and o_2 are bound to $Mixer1$ and $Mixer2$, respectively. Assume that $out(o_1)$ and $out(o_2)$ are still inside $Mixer1$ and $Mixer2$ when o_3 is ready. Fig. 5(b) shows the scheduling result generated by our method, where o_3 is bound to $Mixer1$. We can see that the transportation of $out(o_1)$ is eliminated and the total wash time is only 2 s. However, if o_3 is bound to $Mixer2$ (see Fig. 5(c)), the total wash time increases to 6 s due to a lower diffusion coefficient of $out(o_1)$. Moreover, if o_3 is bound to another mixer which is not used for executing any father operation of o_3, e.g., $Mixer3$ in Fig. 5(d), the total wash time increases to 8 s and the completion of o_3 is also postponed.

The strategy for case II: we select the component (the same type as o_i) that has the earliest ready time, i.e., $c_j \in C$ with the minimum value of $t_{ready}(c_j)$ (line 10-11 in Algorithm 1). Then, all the required input fluids are scheduled and transported to the selected component to start the execution of o_i.

Take the sequencing graph described in Fig. 6 as an example. Assume that $out(o_1)$ has been removed from $Mixer1$ when o_5 is ready. To accelerate the execution of bioassay, o_5 is then bound to $Mixer2$ since $t_{ready}(Mixer2)$ is earlier than $t_{ready}(Mixer1)$.

The time complexity of the resource binding and scheduling method is $O(|O| \times (|C| + |E|))$, where $|O|$ and $|E|$ are the number of operations and edges in the sequencing graph and $|C|$ is the number of allocated components.

B. Placement and Routing

After solving the binding and scheduling, the interconnections among allocated components are available, i.e., routing nets $N = \{n_{i,j} | 0 < i < j \leq |C|\}$. The stage assigns exact locations for components and compute flow channels for nets.

1) Placement of components: We present an effective placement method based on the classic simulated annealing (SA) [6], [12] (line 1-8 in Algorithm 2). Our algorithm starts with a

Algorithm 2: Placement and routing design for DCSA-biochips

Input: Resource binding and scheduling results, allocated components C
Output: Placement and routing solutions

1 Initialize $T = T_0$, $P \leftarrow Random\_Placement(C)$;
2 **while** $T > T_{min}$ **do**
3 **for** $i = 1 \rightarrow I_{max}$ **do**
4 $P^{'} \leftarrow Transformation\_operations(P)$;
5 $\Delta \leftarrow Energy(P^{'}) - Energy(P)$;
6 **if** $\Delta < 0 \vee RandomValue(0,1) < e^{-\Delta/T}$ **then**
7 $P \leftarrow P^{'}$;
8 $T \leftarrow \alpha \times T$;
9 **for** *each grid cell* ce_i **do**
10 Initialize $w(i)$ to a constant and $T_i = \emptyset$;;
11 Sort transportation tasks in non-decreasing order according to their start times;
12 j=1;
13 **for** *the* $j-th$ *transportation task* **do**
14 $path(j) \leftarrow A^*\_routing(j)$;
15 **for** *each cell* ce_i *along* $path(j)$ **do**
16 Update $w(i)$;
17 Insert (st_i^j, et_i^j) into T_i;
18 $j = j + 1$;

randomly generated placement of components. The iteration loop is then entered with an initial temperature T=T_0. The reduction rate of T is controlled by a parameter α. For each temperature iteration (including I_{max} times of optimization), new placement solutions are iteratively generated by performing a series of *transformation operations*, such as rotation, translation, etc. The evaluation of placement results is guided by an energy function, which can be computed as follows:

$$Energy(P) = \sum_{n_{i,j} \in N} mdis(i,j) \cdot cp(i,j) \tag{3}$$

where $mdis(i,j)$ and $cp(i,j)$ represent the Manhattan distance and connection priority between components c_i and c_j.

Formula (3) implies that two components are more likely to be placed close to each other if they have a large connection priority. The connection priority of a net can be determined according to the following analysis: 1) to reduce the transportation conflicts among flow channels, $cp(i,j)$ should be set to a large value if the transportation tasks between c_i and c_j are performed concurrently with multiple other tasks and 2) to potentially reduce the cache time of fluids, $cp(i,j)$ should be set to a large value if the fluids transported between c_i and c_j have a lower diffusion coefficient. Thus, $cp(i,j)$ can be computed as:

$$cp(i,j) = \sum_{k=1}^{q} (\beta \cdot nt_k + \gamma \cdot wt_k) \tag{4}$$

where q is the number of transportation tasks between c_i and c_j in the scheduling, β and γ are two weighting factors, nt_k is the number of transportation tasks that are performed concurrently with the k-th task between c_i and c_j, and wt_k is the wash time for removing the residue left by the k-th task in flow channels.

The time complexity of the proposed placement method is $O(\frac{\log T_{min} - \log T_0}{\log \alpha} \times I_{max} \times |E| + |C| + |E|^2)$, where T_{min} is the termination temperature, T_0 is the initial temperature, $|E|$ is the number edges in the sequencing graph, and $|C|$ is the number of allocated components.

2) Routing of flow channels: After obtaining the placement solution, routing is performed to compute the flow channels for each transportation task specified in the scheduling scheme. We present a transportation-conflict-aware routing algorithm to find feasible routing paths with minimized total channel length while avoiding transportation conflicts (line 9-18 in Algorithm 2).

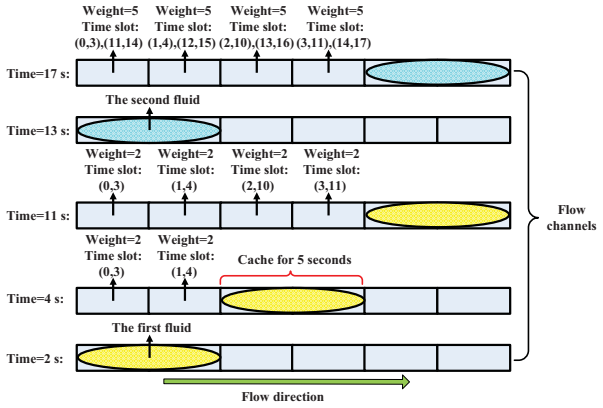


Fig. 7: Illustration of weight and time slot updating of cells.

The layout of the routing plane is partitioned into an array of rectangular cells (see Fig. 4). The weight $w(i)$ of each cell ce_i is initialized to a constant w_e, and each cell is associated with a set $T_i = \{(st_i^j, et_i^j)\}$ of time slots, which represents that ce_i is occupied by the j-th transportation task (referred to as tk_j hereafter) from time st_i^j to et_i^j. Then, we sort all transportation tasks in non-decreasing order according to their start times and sequentially find a routing path for each of them using an improved A* algorithm. Note that, since multiple transportation tasks may be performed simultaneously, to eliminate transportation conflicts, these tasks should avoid sharing the same cells on the routing plane, i.e., $\left(\mathrm{st}_i^j, et_i^j\right) \cap \left(\mathrm{st}_i^k, et_i^k\right) = \emptyset$ for a grid cell ce_i, if transportation tasks tk_j and tk_k are performed in parallel. Once the routing path for a transportation task is found, the weight of each cell ce_i along the routed path is then updated to the wash time required to remove the residue left by the current task, and the new occupation time slot of ce_i (including the cache time) is inserted into. Consequently, flow channels that require shorter wash time are more likely to be used for the next transportation task, thus increasing the length of shared paths among those non-conflicting transportation tasks. More importantly, transportation conflicts can be eliminated among parallel tasks. An example for illustrating the updating process of grid cells is shown in Fig. 7.

In the proposed A* path-finding method, when finding the routing path for the transportation task tk_j, assume that Pr_j is set of parallel tasks corresponding to tk_j. The total cost of the current searching cell ce_k can then be expressed as:

$$Cost(k) = \begin{cases} h(k)+g(k)+w(k), \forall tk_i \in Pr_j, (st_k^i, et_k^i) \cap (st_k^j, et_k^j) = \emptyset \\ +\infty, \quad otherwise \end{cases} \tag{5}$$

where $h(k)$ represents the path length from the source cell to ce_k, $g(k)$ is the estimated path length from ce_k to the target cell, and $w(k)$ is the weight of ce_k.

The time complexity of our routing method is $O(|E| \times (log|E| + |C|^3))$, where $|C|$ and $|E|$ are the number of allocated components and edges in the sequencing graph, respectively.

V. Experimental Results

The proposed algorithms was implemented in C language on a PC with a 3.20 GHz CPU and 8 GB memory. Three benchmarks from real-life biochemical applications and four synthetic benchmarks were used to validate the proposed algorithm [5]. Because there is no existing work targeting the DCSA-based

TABLE I: Comparisons on the execution time, resource utilization, total channel length, and CPU time

Benchmark	Operations	Allocated Components	Execution time (s)			Resource utilization (%)			Total channel length (mm)			CPU time (s)	
			Ours	BA	Imp (%)	Ours	BA	Imp (%)	Ours	BA	Imp (%)	Ours	BA
PCR	7	(3,0,0,0)	30	30	0.0	47.8	47.8	0.0	420	420	0.0	0.00	0.00
IVD	12	(3,0,0,2)	39	39	0.0	61.6	61.6	0.0	240	240	0.0	0.00	0.00
CPA	55	(8,0,0,2)	96	102	5.9	69.5	57.4	21.1	1490	1530	2.6	0.02	0.03
Synthetic1	20	(3,3,2,1)	39	43	9.3	72.7	69.3	4.9	1060	1170	9.4	0.01	0.01
Synthetic2	30	(5,2,2,2)	49	54	9.3	49.5	43.8	13.0	1780	1900	6.3	0.01	0.01
Synthetic3	40	(6,4,4,2)	46	51	9.8	48.7	43.0	13.3	2000	2260	11.5	0.01	0.02
Synthetic4	50	(7,4,4,3)	51	57	10.5	47.5	35.1	35.3	2490	2780	10.4	0.02	0.03
Average			–	–	6.4	–	–	12.5	–	–	5.7	–	–

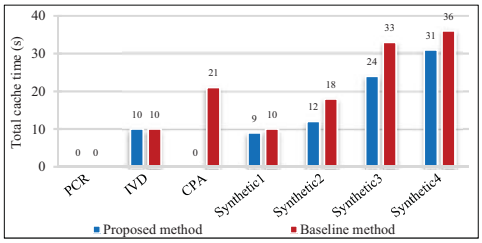


Fig. 8: Comparison on the total cache time in flow channels.

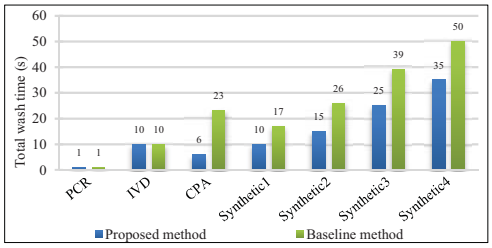


Fig. 9: Comparison on the total wash time of flow channels.

physical design problem considering both wash optimization and transportation conflicts, we compare the proposed algorithm with a baseline algorithm (BA), which binds each ready operation to a qualified component that has the earliest ready time, and then generates a placement and routing solution without transportation conflicts using a construction-by-correction approach (i.e., generating an initial solution and then correct those unsatisfactory component postions/routing paths sequentially). The parameters used in this paper are set as follows: $\alpha = 0.9$, $\beta = 0.6$, $\gamma = 0.4$, $T_0 = 10000$, $I_{max} = 150$, $T_{min} = 1.0$, $t_c = 2.0$, and $w_e = 10$. Table I shows the details of aforementioned benchmarks and the experimental results, where column 2 is the number of operations contained in each assay, column 3 shows the list of allocated components in the following format: (Mixers, Heaters, Filters, Detectors), and the remaining columns show the experimental results in terms of execution time, resource utilization, total channel length, and runtime, respectively. Imp (%) in Table I provides the relatively improvement of our algorithm over BA.

It can be seen from Table I that the proposed algorithm achieves a 0.0%-10.5% execution time reduction across all the seven benchmarks and 6.4% reduction rate on average. Moreover, the proposed algorithm outperforms BA by 12.5% and 5.7% on average in terms of resource utilization and total channel length, respectively. The last two columns of Table I show the CPU times of the two algorithms, we can see that the proposed algorithm is sufficiently efficient.

In addition, Fig. 8 shows the total cache time in flow channels (i.e., the sum of fluid cache times), which is effectively reduced by the proposed algorithm, particularly in the benchmarks with large scale input. Fig. 9 shows the comparison on the total wash time of flow channels. Obviously, the proposed algorithm improves wash efficiency among different operation/transportation tasks without introducing any transportation conflict.

VI. Conclusion

We have formulated the first practical flow-layer physical design problem for DCSA-based biochips, and presented a top-down synthesis flow to generate an efficient solution for biochips with distributed storage. Three benchmarks from real-life biochemical applications and four synthetic benchmarks have been used to validate the effectiveness of proposed algorithm. Experimental results have shown that the proposed algorithm leads to a shorter execution time, less flow-channel length, and a higher on-chip resource utilization. Future work will consider the optimization of control logic [13] to reduce the overall complexity of such platform.

Acknowledgment

The work of T.-Y. Ho was supported in part by the Technical University of Munich – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763. The work of B. Li and U. Schlichtmann was supported in part by Deutsche Forschungsgemeinschaft (DFG) through TUM International Graduate School of Science and Engineering (IGSSE).

References

[1] J. W. Hong, Y. Chen, W. F. Anderson, and S. R. Quake, "Molecular biology on a microfluidic chip," *J. Phys.-Condes. Matter*, vol. 18, no, 18, pp. 691-701, 2006.
[2] C. D. Chin et al, "Microfluidics-based diagnostics of infectious diseases in the developing world," *Nat. Med.*, vol. 17, no. 8, pp. 1015-1019, 2011
[3] S. Einav et al., "Discovery of a hepatitis C target and its pharmacological inhibitors by microfluidic affinity analysis," *Nat. Biotechnol.*, vol. 26, no. 9, pp. 1019-1027, 2008.
[4] J. Melin and S. R. Quake, "Microfluidic large-scale integration: The evolution of design rules for biological automation," *Annu. Rev. Biophys. Biomol. Struct*, vol. 36, pp. 213-231, 2007.
[5] C. Liu, B. Li, H. Yao, P. Pop, T.-Y. Ho, and U. Schlichtmann, "Transport or store?: Synthesizing flow-based microfluidic biochips using distributed channel storage," *Proc. of DAC*, 2017, pp. 49:1-49:6.
[6] W. H. Minhass, P. Pop, and J. Madsen, "Architectural synthesis of flow-based microfluidic large-scale integration biochips," *Proc. of CASES*, 2012, pp. 181-190.
[7] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," *Proc. of ASP-DAC*, 2017, pp. 530-535.
[8] K. H. Tseng, S. C. You, J. Y. Liou, and T.-Y. Ho, "A top-down synthesis methodology for flow-based microfluidic biochips considering valve-switching minimization," *Proc. of ISPD*, 2013, pp. 123-129.
[9] K. Hu, T.-Y. Ho, and K. Chakrabarty, "Wash optimization and analysis for cross-contamination removal under physical constraints in flow-based microfluidic biochips," *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 4, pp. 559-572, 2016.
[10] D. Brune and S. Kim, "Predicting protein diffusion coefficients," *Proc. Nat. Acad. Sci*, vol. 90, no. 9, pp. 3835-3839, 1993.
[11] G. D. Micheli, "Synthesis and optimization of digital circuits," *McGraw-Hill Higher Education*, 1994.
[12] S Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.
[13] Q. Wang, S. Zuo, H. Yao, T.-Y. Ho, B. Li, U. Schlichtmann, and Y. Cai, "Hamming-distance-based valve-switching optimization for control-layer multiplexing in flow-based microfluidic biochips," *Proc. of ASP-DAC*, 2017, pp. 524-529.