*electronics*

MDPI

*Article*

# Three-Stage Rapid Physical Design Algorithm for Continuous-Flow Microfluidic Biochips Considering Actual Fluid Manipulations

**Genggeng Liu [1,2,3], Yufan Liu [1,2,3], Youlin Pan [1,2,3] and Zhen Chen [1,2,3,*]**

[1] College of Computer and Data Science, Fuzhou University, Fuzhou 350116, China; liugenggeng@fzu.edu.cn (G.L.); 211027179@fzu.edu.cn (Y.L.); 210310013@fzu.edu.cn (Y.P.)
[2] Engineering Research Center of Big Data Intelligence, Ministry of Education, Fuzhou University, Fuzhou 350116, China
[3] Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China
* Correspondence: chenzhen@fzu.edu.cn

**Abstract:** With the continuous development of microfluidic technology, continuous-flow microfluidic biochips (CFMBs) are being increasingly used in the Internet of Things. The automation design of CFMBs has also received widespread attention. The architecture design of CFMBs is divided into a high-level synthesis stage and a physical design stage. Among them, the problem of the physical design stage is very complex. At this stage, the chip architecture is generated based on the device library and a set of flow paths, taking into account the actual fluid manipulations, while minimizing the cost of the chip, such as the number of ports, total length of flow channels, number of flow channel intersections. As fabrication technology advances, the number of devices integrated into CFMBs is increasing. The existing physical design algorithms can no longer meet the design requirements of CFMBs in terms of time. Therefore, we propose a three-stage rapid physical design algorithm for CFMBs considering the actual fluid manipulations. The proposed algorithm includes a port-driven preprocessing stage, a force-directed quadratic placement stage, and a negotiation-based routing stage. In the port-driven preprocessing stage, a port-driven preprocessing algorithm is proposed to generate connection matrices between ports and devices to reduce the number of ports introduced. In the force-directed quadratic placement stage, we model the placement problem as an extremum problem of a quadratic cost function, which mathematically reduces the search space significantly and shortens the running time of the algorithm significantly. In the negotiation-based routing stage, a heuristic negotiation-based routing algorithm and a flow channel strategy that prioritizes the construction of parallel execution are proposed to reduce the running time of the algorithm while ensuring that the number of crossings in the routing solution is close to the optimal solution. Experimental results confirm that our proposed method is able to generate the high-quality solutions quickly. Under general scale problems, compared to the existing method based on ILP, our proposed method achieves a speedup ratio of 23,171 in terms of CPU time and optimizations in terms of number of ports and port reuse of 3.18% and 6.52%, respectively. These optimizations come at the cost of only a slight increase in the number of intersections, the flow length, and the number of flow valves. In addition, our proposed method can effectively solve large-scale problems that cannot be solved by existing method based on ILP.

**Keywords:** microfluidic biochips; physical design; force-directed; quadratic placement; routing



**Citation:** Liu, G.; Liu, Y.; Pan, Y.; Chen, Z. Three-Stage Rapid Physical Design Algorithm for Continuous-Flow Microfluidic Biochips Considering Actual Fluid Manipulations. *Electronics* **2024**, *13*, 332. https://doi.org/10.3390/electronics13020332

Academic Editor: Roald M. Tiggelaar

Received: 13 November 2023
Revised: 3 January 2024
Accepted: 9 January 2024
Published: 12 January 2024



## 1. Introduction

Recent years have witnessed the rapid growth of the Internet of Things (IoT), where different types of real-world elements such as wearable sensors are connected and allowed to autonomously interact with each other, promoting the development of fields such as

Healthcare IoT [1], smart cities [2], industrial IoT [3], etc. Continuous-Flow Microfluidic Biochips (CFMBs) have received widespread attention due to their miniaturization and integration. Similar to conventional IoT-embedded devices, CFMBs are nowadays equipped with an array of computer-controlled biosensors and actuators, such as microfluidic electrochemical biosensors [4] and triboelectric nanogenerator [5]. The integration of these components forms a cyber-physical microfluidic system that offers key benefits in terms of fault-tolerance [6] and quantitative analysis [7]. Based on these advantages, CFMBs have been widely used in many fields, such as industrial IoT [8], biochemistry, and biomedicine, including cancer diagnosis [9], cell culture [10], characterization of cell membrane [11], cell separation [12], etc.

CFMBs are structured into a control layer and a flow layer, each with its own network of channels. Flow channels in the flow layer are used to transport samples and reagents. The control channels in the control layer are used to transmit air pressure to control fluid transportation in the flow layer. As shown in Figure 1a, the control layer channels are in yellow and the flow layer channels are in blue. It is important to note that the control channel and the flow channel are not on the same plane. There is an elastic membrane in the intersection region of the control channel and the flow channel, as shown by the red dashed circle up in Figure 1a, and the structure in the intersection region is called a valve. Figure 1b shows a sectional view of the valve, which can be opened and closed by controlling the air pressure in the control channel. Figure 1c illustrates a top view of the appearance of the valve when it is open and closed. When no air pressure is input into the control channel, the elastic membrane maintains its initial position, the flow channel circulates, and fluid can flow in the flow channel. When air pressure is input into the control channel, the elastic membrane in the area where the control channel overlaps the flow channel presses downward on the corresponding flow channel, thus blocking it. By using valves as basic control units, complex microfluidic devices such as multiplexers and mixers [13] can be constructed, and predefined biological assay plans can be automatically executed [14,15].

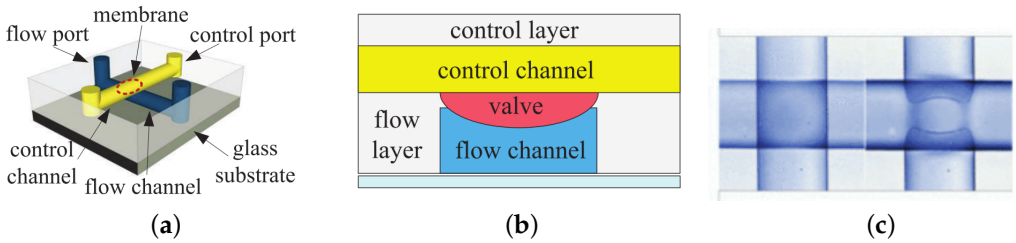


**Figure 1.** (**a**) Schematic of a biochip [16]. (**b**) The cross-sectional view of Figure (**a**) [16]. (**c**) The appearance of the valve [17].

When performing bioassays on CFMBs, fluid samples and reagents need to be transferred between devices to perform different biochemical operations. However, each transport task requires a separate flow path during execution. To drive fluid transfer, one end of the flow path needs to be connected to a flow port to provide a pressure source. In addition, because the flow channels and devices are not initially in a vacuum but filled with air, the waste port needs to be connected to the other end of the flow path to relieve air pressure and prevent channel blockage [18].

As manufacturing technology continues to advance, the feature size of CFMBs has been gradually reduced, especially the size of the valves has been significantly reduced from $15 \times 15$ $\mu m^2$ [19] to $6 \times 6$ $\mu m^2$ [13], and thousands of valves can be fabricated on coin-sized chip platforms, with a density approaching 1 million microvalves per square centimeter [20]. To fulfill the need for more functionality, chip designs have become increasingly complex, with tens of thousands of valves needing to be integrated into chips smaller than a coin. For example, in the field of molecular diagnostics, Fluidigm has developed $96 \times 96$ dynamic valve arrays capable of performing 9216 polymerase chain

reaction (PCR) experiments in parallel, but they require more than 25,000 valves for a variety of fluidic operations [21]. In addition, both device placement and flow channel routing have proven to be NP-complete problems during the physical design phase of the chip-flow layer, which makes algorithms that traverse the entire space difficult to scale to larger-scale problems. Nowadays, chip design requires timeliness. Therefore, designing a polynomial-time-efficient algorithm can fulfill the efficiency requirements of the design.

In recent years, many solutions have been proposed for the physical design problems in the architecture design of CFMBs [22–26]. In [22], a top-down holistic design process for CFMBs was proposed for the first time. In [23], a flow-layer architecture design called MiniControl was proposed for generating chip architectures under strict port constraints. The first practical timing-driven flow-channel network construction problem for FBMBs was presented in [24], and a performance-driven placement and routing algorithm was proposed to solve this problem. Meanwhile, the chip architecture generated by the above methods did not consider the actual manipulations of fluid, and most of them assumed that the fluid ports were implicitly connected to the flow channels.

In [25], an architecture design called PathDriver was proposed, which for the first time considered actual fluid manipulations, making the generated flow path network capable of manipulating actual fluid transport and removal. To meet the needs of actual fluid manipulations, separate flow paths need to be constructed for fluid transport, and excess fluid samples and reagents need to be removed after transport. PathDriver formalized the problem into an Integer Linear Programming (ILP) model in the physical design phase and then used the Gurobi solver to solve it. ILP can ensure optimal or nearly optimal solutions, but it cannot provide polynomial complexity solutions for NP-complete problems. When solving large-scale problems, the solution time can be very long, and there may even be cases where the calculation cannot be completed within an acceptable time frame. In [26], an improved synthesis process named PathDriver+ was proposed, which further considers the implementation of diagonal routing and the minimization of the number of fluid ports. However, the ILP method is still used in the physical design phase, and the issue of the high time complexity of the physical design algorithm has not been resolved.

In view of the high time complexity of current flow layer physical design algorithms for CFMBs that consider actual fluid manipulations, which makes it difficult to apply them to large-scale biochip designs, we propose a three-stage rapid physical design algorithm. Under general scale problems, our proposed algorithm can approach the solution quality of ILP. The main contributions of this paper are as follows:

- We propose a three-stage rapid physical design algorithm that divides the physical design problem into the port-driven preprocessing stage, force-directed quadratic placement stage, and negotiation-based routing stage. The algorithm can obtain an optimal chip physical design solution in polynomial time.
- We propose a port-driven preprocessing stage during the physical design phase to determine the connections between ports and devices, effectively reducing the number of ports.
- We propose a force-directed quadratic placement algorithm that considers parallel task constraints and obtains placement feasible solutions in polynomial time.
- We employ five real-world biological benchmarks and ten synthetic benchmarks to evaluate the proposed method. The experimental results demonstrate the effectiveness and time superiority of the three-stage rapid physical design algorithm proposed in this paper.

The rest of this paper is organized as follows. Section 2 discusses the design challenges and problem model of the physical design phase. Section 3 presents the proposed three-stage rapid physical design algorithm. Section 4 exhibits the experimental results to demonstrate the effectiveness of the proposed algorithm. Section 5 draws conclusions.

## 2. Design Challenges and Problem Model

### *2.1. Connection between Devices and Ports*

In the binding and scheduling scheme of bioassays, bioassay operations are bound to specific devices in the device library, and we refer to all transport tasks, excess liquid removal tasks and waste fluid removal tasks as liquid transport/removal tasks. Each fluid transport/removal task requires a flow port to be connected at one end of the flow path during execution to provide a pressure source to drive fluid transport. At the same time, the other end of the flow path needs to be connected to a waste port to release the air pressure.

Take the binding and scheduling scheme of bioassays shown in Figure 2 as an example. This is a binding and scheduling scheme at the high-level synthesis stage of a group of bioassays and the device library used. $O_1$–$O_9$ represents nine bioassay operations, each of which is bound to a specific device for execution. For example, operation $O_1$ is bound to Filter 1, operations $O_3$ and $O_6$ are bound to Heater 1, and operations $O_8$ and $O_9$ are bound to Mixer. The binding and scheduling scheme includes a total of 16 fluidic transport/removal tasks, including 10 fluidic transport tasks ($\#_1$–$\#_{10}$), 5 excess liquid removal tasks ($*_1$–$*_5$), and 1 waste fluid removal task ($\$_1$). These 16 fluidic transport/removal tasks require the construction of flow paths containing flow ports and waste ports.

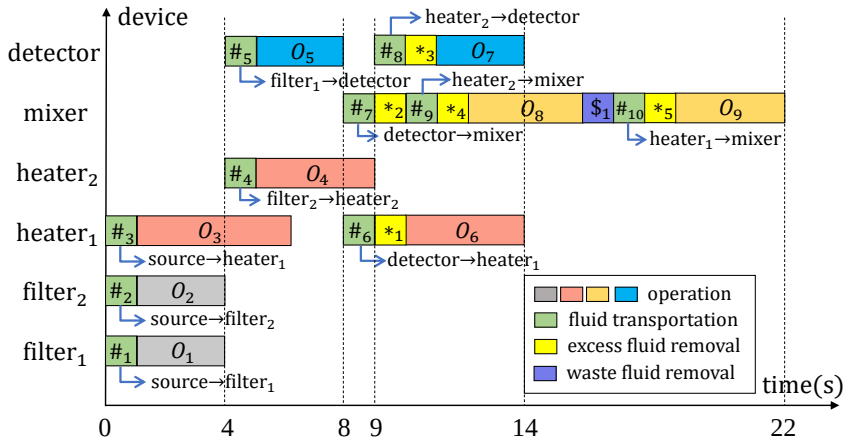


**Figure 2.** Binding and scheduling scheme of a bioassay.

The purpose of fluid transport tasks is to transfer fluid samples and reagents from the source devices to the target devices. Therefore, the complete fluid transport task must include a flow port, source devices, target devices, and a waste port in the flow path.

As the method of injecting air pressure from an external pressure source is used to drive fluid transport in the flow channel [27], it is necessary to avoid air entering the target devices during fluid transport. Therefore, the capacity constraints of the devices should be strictly met during fluid transport so that air can be completely pushed out of the devices. Biological assays should only be performed after excess liquid removal. Figure 3 shows the working process of the mixer. First, input the green fluid that needs to be blended into the mixer, and then remove the excess fluid from both ends of the mixer. Next, input the yellow fluid into the mixer for blending while simultaneously discharging the excess fluid from the yellow fluid. Therefore, the complete excess liquid removal task must include the flow path of the excess liquid channel, a flow port, and a waste port.

When an operation without successor operations is completed, the fluid sample and reagents occupy the device bound to that operation. At this time, the fluid sample and reagents need to be output from the device to the outside of the chip. Therefore, the flow path of the waste removal task must include a flow port, the device where the waste is located, and a waste port.

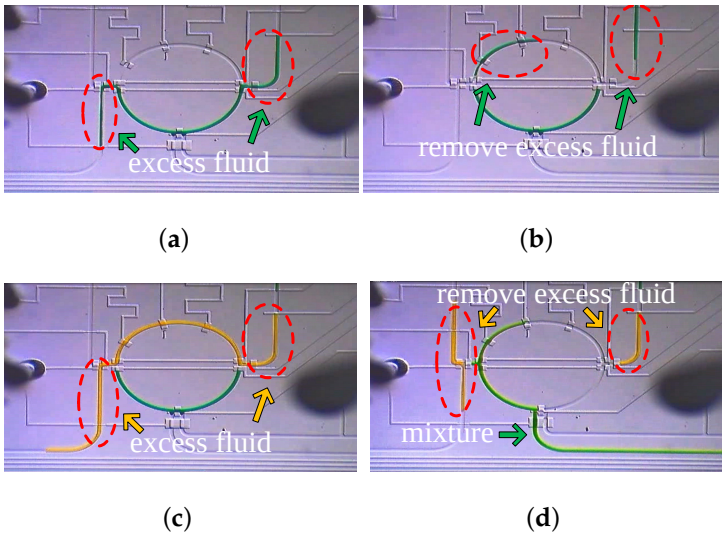


**Figure 3.** Snapshots of fluid manipulation when performing the mixing operation on a biochip [25]. (**a**) Step 1: loading the first fluid flow into the lower half of the mixer, (**b**) step 2: removing the excess fluids left behind by the first fluid, (**c**) step 3: loading the second fluid flow into the upper half of the mixer and starting the mixing operation, and (**d**) step 4: recovering the resulting mixture and removing the corresponding excess fluids.

In the binding and scheduling scheme of a biochip, the connection relationships between devices in fluid transport tasks as well as the locations of excess fluid and waste fluid are provided, but the binding ports for fluid transport/removal tasks have not been determined, nor have the connection relationships between ports and devices. During the physical design phase, it is necessary to obtain the binding relationships between fluid transport/removal tasks and ports, as well as the connection relationships between ports and devices in order to achieve the best placement and routing solution. When only considering the connection relationships between devices without taking into account the connection relationships between ports and devices, it is easy to fall into local optimal solutions to the problem. Therefore, when solving physical design problems, it is necessary to analyze all fluid transport/removal tasks, bind them with ports, and determine the number of ports and the connection relationships between ports and devices. Then, we can conduct placement and routing design based on this information.

### *2.2. Constraints for Parallel Tasks*

The fluid transport/waste removal tasks and biochemical operations that are performed simultaneously are called parallel tasks. Because the method of injecting air pressure from an external source is used to drive the transport of fluid between different devices, each fluid transport/removal task occupies the entire flow path separately during execution. Therefore, in the binding and scheduling scheme of the bioassay, there are constraints on the parallel task for the corresponding flow path of the parallel task.

Parallel task constraints can be divided into two types. In the first, those parallel flow paths cannot overlap, which means that the flow ports, waste ports, source, and target devices used in the flow path of parallel fluid transport/removal tasks cannot be shared. The other is the constraint between the fluid transport/removal task and the device performing the bioassay operation at the same time, which means that the device performing the bioassay operation at that moment cannot be included in the flow path occupied by the transport task being executed at the same time. Specifically, when multiple fluid transport/removal tasks are executed in parallel, the flow ports, waste ports, source, and target devices in the flow path cannot be shared, and their flow channels cannot have intersection points. At the same time, devices that are in a closed state due to executing an operation cannot appear on the flow path of parallel fluid transport/removal tasks. Therefore,

during the physical design phase of a biochip, parallel task constraints should be strictly followed to ensure the implementation of the binding and scheduling scheme for bioassays.

Taking the parallel fluid transport/removal task flow path planning scheme shown in Figure 4a as an example, there are two fluid transport tasks at the same time. One is to transport the fluid in filter 1 to the mixer, and the other is to transport the fluid in the detector to filter 2. The fluid transport/removal task also uses the heater as part of the flow path. Since the two fluid transport tasks are executed at the same time, the valves at the intersection will be opened, which will cause the mixing of the fluids of both tasks and even lead to the situation where the fluid enters non-targeted devices. For example, the fluid in filter 1 should be transported to the mixer, but due to the opening of the valve at the intersection, the fluid is transported to filter 2 or to the working heater, affecting the heating operation. To avoid such situations, we introduce parallel task constraints. The flow ports, waste ports, devices, and flow paths of parallel fluid transport/removal tasks cannot be shared between flow paths, and fluid transport/removal tasks cannot pass through working devices. The correct parallel transport flow path planning scheme should be as shown in Figure 4b.

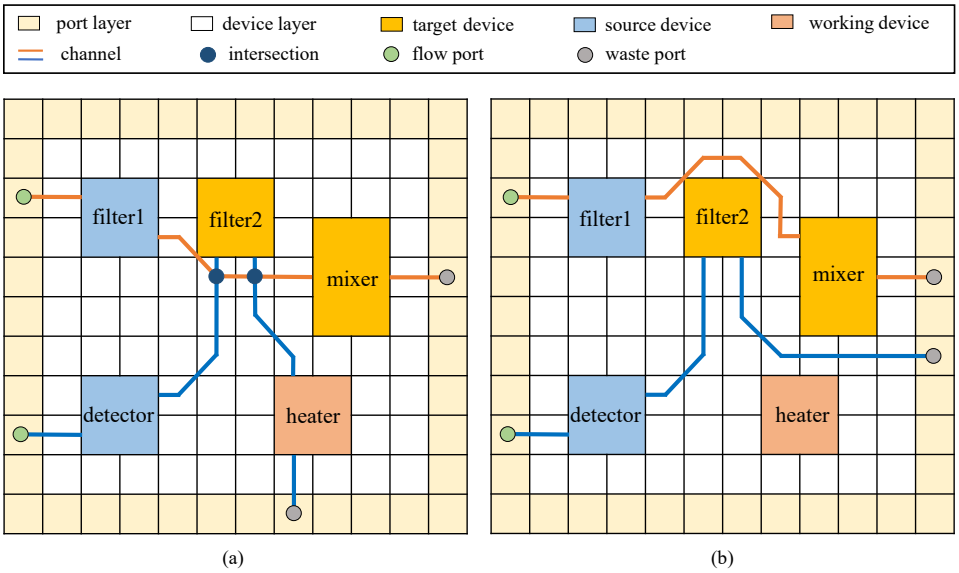


**Figure 4.** (**a**) A fluid path planning scheme that takes into account fluid conflicts. (**b**) The fluid path planning scheme that does not involve any fluid conflicts.

### *2.3. Problem Model*

Based on the above analysis, the physical design problem for CFMBs considering actual fluidic manipulations can be formulated as follows:

***Inputs:***

- A binding and scheduling scheme considering actual fluid manipulations, including the start and end times of all operations and the start and end times of fluid transport/removal tasks.
- A set of flow paths to be constructed considering actual fluid manipulations, including component information used and location information of excess liquid and waste fluid.

***Outputs:***

Flow path planning scheme, including:

- The placement position of ports and devices used in each flow path.
- Routing scheme of the flow channel.

***Objectives:***
Minimize the following indicators.

- Number of fluid ports.
- Total length of the flow channel.
- Number of the flow channel intersections.

## 3. Details of the Proposed Algorithm

In this section, we will discuss the details of the proposed algorithm. In order to obtain a physical design solution in polynomial time, we divide the physical design stage into three substages: the port-driven preprocessing stage, the force-directed quadratic placement stage, and the negotiation-based routing stage.

### *3.1. Port-Driven Preprocessing Stage*

In this section, given the binding and scheduling scheme $T$ for biological assays, the goal of the port-driven preprocessing stage is to determine the number of ports and the connection matrix $C$ between devices and ports, such that the binding and scheduling scheme for biological assays can be implemented through this connection relationship while minimizing the number of fluid ports and connections between devices and ports.

To implement the given binding and scheduling scheme for biological assays, we need to allocate ports for all transport flow paths and determine the connection relationship between ports and devices. Considering those parallel task constraints cannot share devices and ports in transport flow paths, we use the maximum parallel transport time slot priority strategy in this stage to determine the devices and ports connected in all flow paths.

Algorithm 1 shows the pseudocode for the proposed port-driven preprocessing algorithm.

**Algorithm 1** Port-driven preprocessing algorithm

**Input:** binding and scheduling scheme $T$;
**Output:** a set of ports $P$; connection matrix $C$ between flow ports, waste ports, and devices;

$P, C \longleftarrow \varnothing$;
Iterate through each moment of $T$ and place the transportation tasks under that moment into the same task group. Denote the set of all task groups as $T'$;
**while** not all task groups in $T'$ have been traversed **do**
    $t' \longleftarrow$ The task group with the largest number of unexplored parallel fluid transport/removal tasks is included in $T'$;
    **if** $P$ and $C$ do not satisfy the parallel transportation task constraints of $t'$ **then**
        $cost_{min} \longleftarrow \infty$;
        **for** $S_i$ **do**
            Calculate the $cost(i)$ of $S_i$ by (1);
            **if** $cost(i) < cost_{min}$ **then**
                $P' \longleftarrow$The port set $P_i$ for $S_i$;
                $C' \longleftarrow$The connection matrix $C_i$ for $S_i$;
                $cost_{min} \longleftarrow cost(i)$;
            **end if**
        **end for**
        $P \longleftarrow P'$;
        $C \longleftarrow C'$;
    **end if**
**end while**

When determining connections relationship between ports and devices, due to the parallel transport flow path constraints, fluid transport/removal tasks executed at the same time need to occupy a complete flow path separately. In order to reduce the number of introduced ports, we propose the maximum parallel transport time slot priority strategy, which prioritizes analyzing the time slots with the most parallel fluid transport/removal

tasks, and assigns ports to each fluid transport/removal task in this time slot. When analyzing subsequent parallel fluid transport/removal tasks, previously assigned ports are likely to be used, which can increase the port reuse rate and reduce the introduction of new ports. However, when the existing ports and the connection between ports and devices cannot meet the demand, we propose two solutions: $S_1$ is to add a new port and establish a connection between the new port and the device; $S_2$ is to add a connection between an existing port and the device.

The first solution solves the demand between parallel transport flow paths at the cost of increasing the number of ports. The second solution only increases the number of connections between existing ports and devices without adding new ports. However, the second solution may easily increase the maximum connection number between ports and devices, where the maximum connection number of a port refers to the number of devices connected to the port that has the most connections among all ports, and the maximum connection number of a device refers to the number of connections with other devices and ports of the device that has the most connections among all devices. When the maximum connection number between ports and devices is too large, the routing resources around the port and device may be tight, and in severe cases, there may be a routing failure due to the lack of routing resources around the port and device. The number of ports and the number of connections between ports and devices are both optimization objectives. The number of ports is one of the direct optimization objectives in the physical design stage of biological chips, and the number of connections between ports and devices can affect the total length of flow channels and the number of intersections of flow channels in subsequent routing. To evaluate the advantages and disadvantages of these two solutions, the cost function is expressed as follows:

$$cost(i) = \alpha N_{p,i} + \beta N^{c}_{d,p,i} + \gamma C^{max}_{p,i} + \delta C^{max}_{d,i} \tag{1}$$

where $cost(i)$ represents the cost of solution $S_i$, $N_{p,i}$ represents the number of ports introduced in solution $S_i$, $N^{c}_{d,p,i}$ represents the total number of connections between devices and components in solution $S_i$, $C^{max}_{p,i}$ represents the maximum number of connections for a port in solution $S_i$, $C^{max}_{d,i}$ represents the maximum number of connections for a device in solution $S_i$, and $\alpha$, $\beta$, $\gamma$, and $\delta$ are the weight coefficients.

### *3.2. Force-Directed Quadratic Placement Stage*

Given the connection matrix $C$ between ports and devices, as well as the device library $D$ and port set $P$, in this section, the force-directed quadratic placement algorithm has the advantage of low CPU time and the ability to obtain high-quality placement solutions [28]. Therefore, we adopt the quadratic force-directed placement algorithm [29].

CFMBs consist of devices and ports, which are connected by flow channels between devices and between devices and ports. For convenience, the ports and devices that need to be laid out are collectively referred to as modules. The force-directed quadratic placement algorithm first uses the center point of each module as its coordinates and then lays out the modules to minimize the quadratic route length function. However, the resulting placement is often unreasonable, with modules concentrated together and overlapping. Therefore, the force-directed algorithm [30] is used to add repulsive forces to each module, allowing the overlapping modules to spread out reasonably in all directions, eliminating overlaps between modules and leaving space for subsequent routing stages.

In the quadratic placement, the connectivity matrix $C$ is used to represent the connection relationship between modules, where each non-zero element in $C$ represents an edge $e$. The rows and columns, respectively, indicate two modules. The edges $e = (p, q)$ in the connection matrix $C$ between ports and devices are represented by a pair of devices or ports $p$, $q$. The position of a device or port (such as $p$) is represented by coordinates $(x_p, y_p)$,

and the quadratic cost function $\Gamma$ is the sum of the squared Euclidean lengths of all edges $e \in C$. Therefore, the representation of $\Gamma$ is as follows:

$$\begin{aligned}\Gamma &= \sum_{e \in C} \omega_{x,p,q}(x_p - x_q)^2 + \omega_{y,p,q}(y_p - y_q)^2 \\ &= \Gamma_x + \Gamma_y\end{aligned} \tag{2}$$

where $\Gamma$ depends on the position of the connection point, and this paper assumes that the connection point is located at the center of the port or device it represents. The cost function can be divided into the x-direction and y-direction. $\Gamma_x$ represents the cost in the x-direction, while $\omega_{x,p,q}, \omega_{y,p,q}$ represent the connection weights in the x-direction and y-direction, respectively. In the problem model presented in this paper, the cost of establishing flow channels in the x-direction and y-direction is the same, so $\omega_{x,p,q} = \omega_{x,p,q} = 1$. The calculation method of $\Gamma_x$ is mainly discussed later and $\Gamma_y$ is similar to it.

During the placement process, devices and ports are divided into movable modules (total number $M$) and fixed modules (total number $F$). Due to the different placement areas of ports and devices, ports are located at the edge of the chip while devices are located in the center of the chip. Therefore, this paper separates port placement from device placement. When conducting port placement, ports are considered as movable modules while devices are fixed modules. When conducting device placement, devices are considered as movable modules while ports are fixed modules. Since only the positions of movable modules need to be determined, the x coordinates of the $M$ movable modules can be represented by vector $\boldsymbol{x}$:

$$\boldsymbol{x} = (x_1, x_2, x_3, \ldots, x_M)^T \tag{3}$$

According to (3), the quadratic cost function $\Gamma_x$ in (2) can be expressed as a quadratic matrix:

$$\Gamma_x = \frac{1}{2}(\boldsymbol{x}^T \boldsymbol{C_M} \boldsymbol{x}) + \boldsymbol{x}^T \boldsymbol{d_x} + const \tag{4}$$

where matrix $\boldsymbol{C_M}$ represents the connection between movable modules, while vector $\boldsymbol{d_x}$ represents the connection between movable modules and fixed modules. $\boldsymbol{C_M}$ is of dimension $M \times M$, and has entry $c_{i,j}$ in row $i$, and column $j$. $\boldsymbol{d_x}$ is of dimension $M$ and has entry $d_i$ in row $i$. The method for determining $\boldsymbol{C_M}$ and $\boldsymbol{d_x}$ is as follows. For example, let us take an arbitrary edge $e = (i, j)$ in the connection matrix $C$ between ports and devices, if both modules $i$ and $j$ are movable, then $\omega_{x,ij}$ is added to the diagonal matrix entries $c_{i,i}$ and $c_{j,j}$, and $\omega_{x,i,j}$ is subtracted from the off-diagonal entries $c_{i,j}$ and $c_{j,i}$. If one module is fixed, e.g., module $j$, then $\omega_{x,i,j}$ is added to $c_{i,i}$, and $x_j \times \omega_{x,i,j}$ is subtracted from the vector entry $d_i$. If both modules are fixed, then $\boldsymbol{C_M}$ and $\boldsymbol{d_x}$ do not change.

The minimum value of $\Gamma_x$ can be obtained by finding the point where the derivative is equal to zero, the derivative of $\Gamma_x$ can be expressed using the nabla operator, denoted as $\bigtriangledown_x = (\partial/\partial x_1, \partial/\partial x_2, \partial/\partial x_3, \ldots, \partial/\partial x_M)^T$:

$$\bigtriangledown_x \Gamma_x = \boldsymbol{C_M} \boldsymbol{x} + \boldsymbol{d_x} = 0 \tag{5}$$

The module x-coordinate placement position with the minimum total length of the flow channel can be obtained by solving the linear equation system (5). The obtained module coordinates with the minimum total length of flow channels will overlap heavily during placement. A force-directed quadratic placement algorithm is used to scatter the modules on the chip using additional repulsive forces to remove overlaps. The force-directed algorithm is based on the principles of universal gravity and repulsion in physics, simulating the interaction between nodes in space, causing the modules with connection relationships in the graph to gather together, and the presence of repulsive forces ensures that the modules do not overlap.

Because the placement areas of devices and ports are different, there will be no overlap between devices and ports. Therefore, there is repulsive force between all similar modules, such as between devices and devices, and between ports and ports. There is no repulsive

force between different modules, such as between devices and ports. The repulsive force between any two modules of the same kind can be expressed as:

$$F_{r,u_i,u_j} = \frac{K_r}{l^2_{u_i,u_j}} \tag{6}$$

where modules $u_i, u_j \in P$ or $u_i, u_j \in D$, $l$ is the Euclidean distance between module $u_i$ and module $u_j$, calculated as $l_{u_i,u_j} = \sqrt{(x_{u_i} - x_{u_j})^2 + (y_{u_i} - y_{u_j})^2}$, the direction of the repulsive force $F_{r,u_i,u_j}$ is opposite to the line connecting the two modules, and $K_r$ is the coefficient in force-directed algorithms; in this paper, $K_r = 1$.

Due to the fact that the direction of the repulsive force $F_{r,u_i,u_j}$ is opposite to the line connecting module $u_i$ and module $n_j$, the force can be separated into components in the x-direction and y-direction. Taking the x-direction as an example, the formula for the magnitude of the repulsive force in the x-direction, $F^x_{r,u_i,u_j}$, is calculated as:

$$F^x_{r,u_i,u_j} = F_{r,u_i,u_j} \cdot \frac{\left| x_{u_i} - x_{u_j} \right|}{l_{u_i,u_j}} \tag{7}$$

According to (6) and (7), all the repulsive forces $F_{r,u_i}$ that module $i$ receives can be expressed as:

$$\begin{aligned} F_{r,u_i} &= \sum_{u_j \in E} F_{r,u_i,u_j} \\ &= \sum_{u_j \in E} F^x_{r,u_i,u_j} + \sum_{u_j \in E} F^y_{r,u_i,u_j} \end{aligned} \tag{8}$$

where $E = D$ when module $u_i$ represents a device and $E = P$ when module $u_i$ represents a port.

Due to the significant overlap at the initial positions, a large repulsive force is obtained when calculating the repulsive force for the first time. If the module is moved directly according to the size of the repulsive force, it may lead to excessive movement distance and deteriorate the quality of the final placement solution. Therefore, when moving the module, the movement distance should be multiplied by a reduction factor $\tau$. Taking the x-direction as an example, the movement distance $\triangle x$ of model $u_i$ is expressed as:

$$\triangle x_{u_i} = \tau F^x_{r,u_i} \tag{9}$$

According to (9), the new position of the module after movement is represented by:

$$\boldsymbol{x'} = \boldsymbol{x} - \triangle \boldsymbol{x} \tag{10}$$

In the placement iteration, the placement cost change $\triangle$ and the parameter $\varepsilon$ are added to determine whether to terminate the iteration. The placement cost change $\triangle$ is the absolute value obtained by subtracting the placement cost $\Gamma^{pre}$ of the previous placement iteration from the cost $\Gamma$ of the current placement iteration. The parameter $\varepsilon$ is set in advance. The placement solution is only output if $\triangle \leq \varepsilon$.

Algorithm 2 shows the pseudocode for a force-directed quadratic placement algorithm.

**Algorithm 2** Quadratic placement algorithm based on force-directed approach

**Input:** Set of ports $P$; connection matrix $C$; device library $D$;
**Output:** Placement coordinates $x_p, y_p, x_d, y_d$ of ports and devices on the chip;

Initialize $x_d, y_d, x_p, y_p$;
**while** $\triangle \leq \varepsilon$ **do**
  Treat ports as fixed modules and devices as movable modules;
  $x_d, y_d \longleftarrow$Compute the device coordinates by (5);
  **while** overlap between movable mobile and fixed modules **do**
    $F_r^x, F_r^y \longleftarrow$Calculate the repulsive forces of all modules by Formulas (6) and (7);
    **for** $u_i \in D$ **do**
      $\triangle x_{u_i} \longleftarrow \tau F_{r,u_i}^x$;
      $\triangle y_{u_i} \longleftarrow \tau F_{r,u_i}^y$;
    **end for**
    $x_d' \longleftarrow x_d - \triangle x$;
    $y_d' \longleftarrow y_d - \triangle y$;
    $x_d, y_d \longleftarrow x_d', y_d'$;
  **end while**
  Treat devices as fixed modules and ports as movable modules;
  $x_p, y_p \longleftarrow$Compute the device coordinates by (5);
  **while** overlap between movable mobile and fixed modules **do**
    $F_r^x, F_r^y \longleftarrow$Calculate the repulsive forces of all modules by Formulas (6) and (7);
    **for** $u_i \in P$ **do**
      $\triangle x_{u_i} \longleftarrow \tau F_{r,u_i}^x$;
      $\triangle y_{u_i} \longleftarrow \tau F_{r,u_i}^y$;
    **end for**
    $x_p' \longleftarrow x_p - \triangle x$;
    $y_p' \longleftarrow y_p - \triangle y$;
    $x_p, y_p \longleftarrow x_p', y_p'$;
  **end while**
  $\Gamma \longleftarrow$Calculate the cost of this placement by (2);
  $\Delta = |\Gamma^{pre} - \Gamma|$ ;
  $\Gamma^{pre} \longleftarrow \Gamma$;
**end while**

### *3.3. Negotiation-Based Routing Stage*

In this section, we will use a negotiation-based routing algorithm to perform routing given the coordinates $x_d$, $y_d$, $x_p$, and $y_p$ of ports and devices, connection matrix $C$ for ports and devices, device library $D$, and set of ports $P$.

For the routing structure, the same diagonal routing structure as in [26] was used in this paper, where each grid cell has eight adjacent cells. The structure is shown in Figure 5.

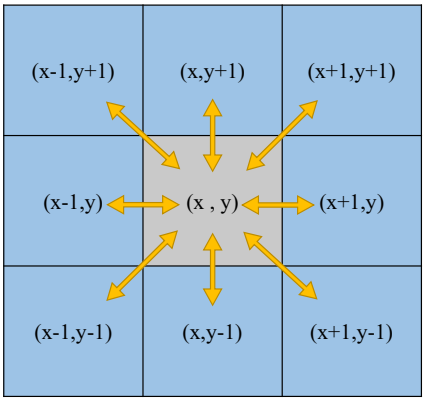


**Figure 5.** The eight adjacent cells of cells (x, y) in the diagonal routing.

Algorithm 3 shows the pseudocode for the negotiation-based routing algorithm.

**Algorithm 3** Negotiation-based routing algorithm

**Input:** The coordinates $x_d, y_d, x_p, y_p$ of ports and devices on the chip; the binding scheduling scheme $T$; the connection matrix $C$;
**Output:** Routing scheme of chips;

```
Iterate through each moment of T and place the transportation tasks under that moment into the same task group. Denote the set of all task groups as T';
while not all task groups in T' have been traversed do
    The parallel transportation flow path L is set to empty;
    t' ⟵ The task group with the largest number of unexplored parallel fluid transport/removal tasks is included in T';
    while not all flow paths in t' have been routed do
        p ⟵ The longest Euclidean distance flow path in t';
        Routing p using A* algorithm without allowing any intersections;
        if routing failure then
            Use the A* algorithm that routes p in cases where parallel transportation flow paths are not allowed to cross;
            L ⟵ newly added flow paths;
        end if
    end while
end while
```

During the routing process, the routing sequence has a significant impact on the quality of the final routing solution. Improper routing sequence can lead to increased routing costs or even routing failure due to constraints in parallel transport tasks where fluid transport/removal tasks executed simultaneously should not occupy intersecting flow channels. When modules in later routing are encircled by parallel flow paths, insufficient routing space may occur and ultimately result in routing failure. Therefore, we propose to prioritize the routing of flow paths with the most parallel fluid transport/removal tasks, and then calculate the Euclidean distance between each set of devices and ports that need to be connected within this group of parallel transport flow paths. The routing is then performed in descending order of distance.

When handling parallel transportation tasks, there are usually multiple flow paths that need to be routed, which are parallel transportation flow paths. In order to meet the requirements of parallel transportation tasks, i.e., no intersection points between parallel transportation flow paths, we add the routed flow channel to the non-intersecting flow path $L$ every time a parallel flow path is routed. When routing other parallel transportation flow paths, the flow paths in the non-intersecting flow path $L$ are all treated as non-intersecting flow paths. When processing the next parallel transportation task, the non-intersecting flow path $L$ is set to empty again.

After determining the routing order, the A* search algorithm [31] is used to establish flow channels between the two modules that need to be routed, without allowing any flow channel intersection. If the routing fails, the A* search algorithm is used again to establish flow channels while allowing non-parallel transport flow path intersections.

### *3.4. Time Complexity Analysis*

The time complexity of the proposed three-stage rapid physical design algorithm in this paper is divided into three parts: the port-driven preprocessing stage, the force-directed quadratic placement stage, and the negotiation-based routing stage. The time complexity of each of these three stages will be analyzed separately below.

In the port-driven preprocessing stage, we need to traverse each transport task to determine the number of ports and the connections between devices and ports. Assuming that there are $L$ transport tasks, there are two solutions when the existing ports cannot meet the demand for parallel tasks. The time complexity of this stage is $O(L)$.

In the force-directed quadratic placement stage, we first use the center point of each module as the coordinate of the module and then lay out the modules to obtain the placement solution that minimizes the quadratic route length function. Next, we use the force-directed algorithm to add repulsive forces to each module so that the overlapping and clustered modules can spread out reasonably in all directions, eliminating the overlaps between modules. Assuming the total number of ports and devices is $N$, the time complexity of solving the quadratic matrix optimization problem is $O(N^3)$, and the time complexity of the force-directed algorithm is also $O(NlogN)$. Assuming the number of iterations is $M$, the time complexity of this stage is $O(M(N^3 + NlogN))$.

In the negotiation-based routing stage, first, all transport tasks are traversed to select the group of flow paths with the most parallel transports. Then, the A* algorithm is used to route for this group of flow paths. The time complexity of the A* algorithm can be expressed as $O(b^d)$, where $b$ is the average branching factor of each node expanded and $d$ is the depth of the shortest path. Since diagonal routing is used and each unit has eight adjacent units, $b = 8$. Therefore, the algorithmic time complexity of this stage is $O(L \times 8^d)$. The total time complexity of the proposed algorithm is $O(M(N^3 + NlogN) + L \times 8^d)$.

ILP problems are NP-complete problems [32], and the branch-and-bound method [33] is usually used to find the exact solution of the ILP problem. Assume that the integer linear programming problem has $N$ variables and the $k$ variable takes values in the range $[0, m_k)$, and the branch-and-bound method requires solving the linear programming problem $\prod_{k=1}^{N} m_k$ times in the worst case. The existing algorithm is able to solve the linear programming problem in $O(N^{2.055})$ [34]. Therefore, the time complexity of the integer linear programming problem is $O(N^{2.055}\prod_{k=1}^{N} m_k)$. The time complexity of solving the physical design problem of CFMBs using ILP is $O([WH(7 + 15|L| + |N|) + 2|N||]^{2.055}(WH)^N 2^{WH})$ [26], where $W$ and $H$ denote the length and width of the chip, respectively.

Therefore, the time complexity of our algorithm outperforms ILP-based physical design algorithm and can be solved in polynomial time.

## 4. Experimental Results

The proposed three-stage rapid physical design algorithm for CFMBs considering actual fluid manipulation was implemented in Python and tested on a PC with a 3.20 GHz CPU and 8 GB of memory. As shown in Table 1, 15 benchmarks were used to validate the proposed method, including PCR (polymerase chain reaction), IVD (in vitro diagnostics), ProteinSplit, Kinase act-1, and Kinase act-2, which are real-world biochemical applications [24], while the other 10 tests are artificially synthesized benchmarks. The parameters $|F|$, $|C|$, $|Device|$, and $|Area|$ in Table 1 are the number of flow paths, parallel transportation tasks, the number of devices, and the area of the chip, respectively. The parameter settings used in the experiment are as follows: $\alpha = 0.4, \beta = 0.1, \gamma = 0.25, \delta = 0.25, \tau = 0.2, \varepsilon = 8$.

**Table 1.** Benchmarks used in the experiments.

| **Benchmarks** $\|F\|/\|C\|/\|Device\|/\|Area\|(\mathbf{mm^2})$ | | | | |
|---|---|---|---|---|
| PCR<br>7/9/4/70 × 70 | IVD<br>4/9/4/70 × 70 | ProteinSplit<br>7/10/4/70 × 70 | Kinase act-1<br>8/17/4/70 × 70 | Kinase act-2<br>10/32/5/70 × 70 |
| Synthetic1<br>8/12/4/70 × 70 | Synthetic2<br>6/19/5/70 × 70 | Synthetic3<br>7/16/4/70 × 70 | Synthetic4<br>11/16/5/70 × 70 | Synthetic5<br>13/28/5/70 × 70 |
| Synthetic6<br>8/18/8/100 × 100 | Synthetic7<br>14/18/8/100 × 100 | Synthetic8<br>16/24/8/100 × 100 | Synthetic9<br>14/20/8/100 × 100 | Synthetic10<br>10/32/10/100 × 100 |

Currently, the ILP algorithm [26] is used for physical design problem that considers practical fluid operations. The ILP algorithm models the problem in the physical design phase as an objective function and a set of linear equations or inequalities, and then finds the optimal solution of integer-valued variables while satisfying these equations and inequalities. In this paper, two algorithms were run on the aforementioned 15 benchmark tests, and Table 2 shows the corresponding comparison results. The columns $N_{port}$, $N_{intersection}$,

$L_{channel}$, $T_{CPU}$, and $S_p$ represent the number of ports, the number of intersection points in the flow channel, the total length of the flow channel, the CPU running time of the algorithm, and the acceleration ratio, respectively. $Im(\%)$ is the improvement of the algorithm proposed in this paper relative to the ILP algorithm. In the first 10 benchmark tests in Table 2, the ILP algorithm's running time limit was 30 min. In the latter five benchmark tests, due to the increase in problem size, the ILP algorithm's running time limit was extended to 3 h.

**Table 2.** Comparison results between ours and ILP [26] with respect to high-level synthesis and physical design.

| Benchmark | $N_{port}$ | | | $N_{intersection}$ | | | $L_{channel}$ (mm) | | | $T_{CPU}$ (s) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **ILP** | **Ours** | ***Im* (%)** | **ILP** | **Ours** | ***Im* (%)** | **ILP** | **Ours** | ***Im* (%)** | **ILP** | **Ours** | ***Sp*** |
| PCR | 6 | 6 | 0.00 | 4 | 4 | 0.00 | 80 | 90 | −12.50 | 1800 | 0.064 | 28,125 |
| IVD | 4 | 4 | 0.00 | 2 | 2 | 0.00 | 60 | 60 | 0.00 | 1800 | 0.063 | 28,571 |
| ProteinSplit | 4 | 4 | 0.00 | 2 | 2 | 0.00 | 60 | 70 | −16.67 | 1800 | 0.064 | 28,125 |
| Kinase act-1 | 6 | 5 | 16.67 | 3 | 2 | 33.33 | 60 | 70 | −16.67 | 1800 | 0.081 | 22,222 |
| Kinase act-2 | 7 | 6 | 14.29 | 3 | 4 | −33.33 | 60 | 50 | 16.67 | 1800 | 0.116 | 15,517 |
| Synthetic1 | 8 | 8 | 0.00 | 5 | 4 | 20.00 | 110 | 130 | −18.18 | 1800 | 0.070 | 25,714 |
| Synthetic2 | 4 | 5 | −25.00 | 3 | 3 | 0.00 | 60 | 70 | −16.67 | 1800 | 0.149 | 12,080 |
| Synthetic3 | 4 | 5 | −25.00 | 2 | 3 | -50.00 | 70 | 60 | 14.29 | 1800 | 0.064 | 28,125 |
| Synthetic4 | 7 | 5 | 28.57 | 4 | 5 | -25.00 | 80 | 80 | 0.00 | 1800 | 0.110 | 16,363 |
| Synthetic5 | 9 | 7 | 22.22 | 3 | 3 | 0.00 | 60 | 70 | −16.67 | 1800 | 0.067 | 26,865 |
| Synthetic6 | - | 8 | - | - | 6 | - | - | 180 | - | 10,800 | 0.091 | - |
| Synthetic7 | - | 8 | - | - | 8 | - | - | 250 | - | 10,800 | 0.090 | - |
| Synthetic8 | - | 9 | - | - | 8 | - | - | 290 | - | 10,800 | 0.148 | - |
| Synthetic9 | - | 8 | - | - | 7 | - | - | 230 | - | 10,800 | 0.097 | - |
| Synthetic10 | - | 12 | - | - | 11 | - | - | 320 | - | 10,800 | 0.183 | - |
| Average | - | | 3.18 | - | | −5.50 | - | | −6.64 | - | | 23,171 |

From Table 2, it can be seen that the three-stage rapid physical design algorithm proposed in this paper reduces the average number of ports by 3.18%. Additionally, the algorithm proposed in this paper has an average acceleration ratio of 23,171 compared to the ILP algorithm, significantly reducing the algorithm running time in the physical design stage. However, the average number of port intersection points and total flow channel length increased by 5.50% and 6.64%, respectively. The problem size of the five synthesized biological benchmarks in Table 2 is larger than that of the five biological benchmarks that exist in reality. Therefore, the running time of the ILP algorithm is extended to 3 h, but a solution cannot be obtained within the specified time. The three-stage rapid physical design algorithm proposed in this paper can be solved in an average of 0.122 s.

We analyze this result for three reasons: (1) Because the three-stage rapid physical design algorithm prioritizes reducing the number of ports in the port-driven preprocessing stage, the non-parallel transport flow paths connected to the same port increase, and the port reuse rate increases during placement and routing, thereby reducing the number of introduced ports. (2) The ILP algorithm is a method of accurately solving discrete optimization problems, which considers all possible solutions and checks each solution when solving the problem. Therefore, this algorithm needs to traverse the solution space, and its running time increases exponentially with the problem size. The three-stage rapid physical design algorithm proposed in this paper can be solved in polynomial time, greatly reducing the CPU time in the physical design stage. (3) The small scale of the experimental examples results in little difference between data, but large differences in percentage. For example, although the number of intersection points changes from 2 to 3, the difference is only 1, but it translates to −50% as a percentage. In the first 10 examples, ILP cannot traverse the entire solution space in 30 min, so it may return a suboptimal solution. When solving larger experimental examples, ILP cannot solve them within the specified time, and the experimental data cannot be compared, resulting in an increase in the number of intersection points and total flow channel length for the three-stage rapid physical design algorithm proposed in this paper.

Figure 6 shows the comparison results of the port reuse rate between the three-stage physical design algorithm proposed in this paper and ILP. In continuous microfluidic biochips, ports can be used not only to input fluid samples and reagents into the device but also to input air pressure to control fluid transport. The higher the port reuse rate, the higher the frequency of sharing ports among flow paths, and the fewer ports introduced. In the port-driven preprocessing stage, a port-driven preprocessing algorithm is proposed to generate connection matrices between ports and devices to reduce the number of ports introduced. It can be seen that the three-stage physical design algorithm proposed in this paper achieved an average increase of 6.52% in port reuse rate.

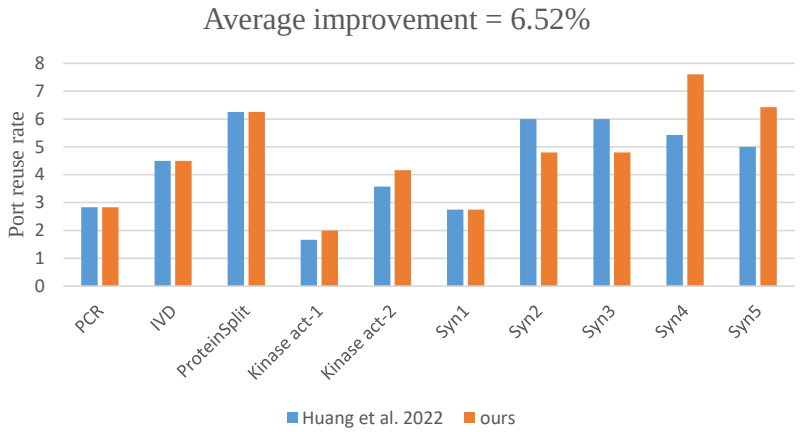


**Figure 6.** Results of port reuse rate as compared with Huang et al., 2022 [26].

Figure 7 shows the comparison results of the number of valves introduced between the three-stage physical design algorithm proposed in this paper and ILP. The three-stage physical design algorithm proposed in this paper deteriorated by 7.01% in the number of valves introduced. The analysis is as follows: since the reuse rate of the ports was considered at the beginning of the algorithm, the number of flow paths connected to the same port has increased. When different flow paths are connected to the same port, intersection points occur, and valves need to be placed at the intersection points to guide the direction of fluid transport and avoid conflicts with other fluid transport/removal tasks, ultimately leading to an increase in the number of introduced valves.

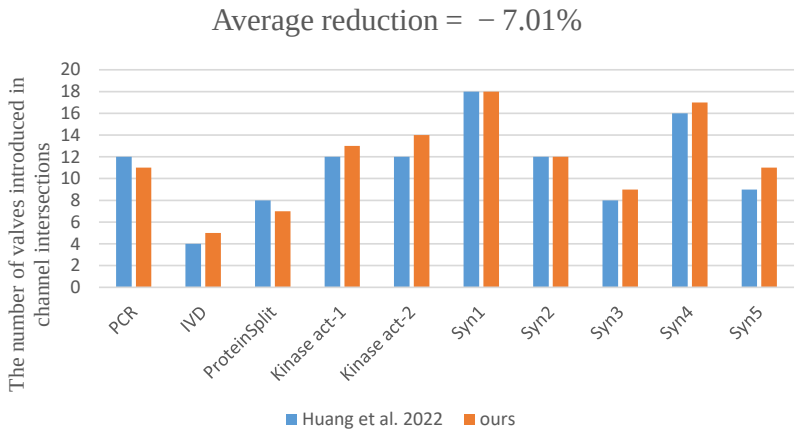


**Figure 7.** Results of the number of extra introduced valves as compared with Huang et al., 2022 [26].

## 5. Conclusions

We investigate the physical design issues of CFMBs considering the actual manipulation of fluid and propose a three-stage rapid physical design algorithm for CFMBs, which

can systematically solve such problems. The proposed algorithm divides the physical design problem of CFMBs considering real fluid operations into three stages: port-driven preprocessing, force-directed quadratic placement, and negotiation-based routing, providing high-quality physical design solutions and excellent computational efficiency. The effectiveness of the proposed method is validated by five realistic biochemical applications and ten synthetic benchmarks. Experimental results confirm that our proposed method is able to generate high-quality solutions quickly. Under general scale problems, the solution quality of the proposed method is close to that of the state-of-the-art method based on ILP, but the time to run the proposed method is drastically reduced. In addition, our proposed method can solve large-scale problems that cannot be solved by the existing method based on ILP.

**Author Contributions:** Conceptualization, G.L. and Y.L.; methodology, G.L., Y.L. and Y.P.; software, Y.L.; validation, Y.L. and Y.P.; formal analysis, Y.L. and Y.P.; investigation, Y.L. and Y.P.; resources, G.L.; data curation, Y.L. and Y.P.; writing—original draft preparation, G.L. and Y.L.; writing—review and editing, G.L., Y.P. and Z.C.; visualization, Y.L. and Y.P.; supervision, Z.C.; project administration, Z.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported by the Fujian Natural Science Funds under Grant No. 2023J06017 and the National Natural Science Foundation of China under Grant No. 62372109.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Kumar, M.; Kumar, A.; Verma, S.; Bhattacharya, P.; Ghimire, D.; Kim, S.h.; Hosen, A.S. Healthcare internet of things (H-IoT): Current trends, future prospects, applications, challenges, and security issues. *Electronics* **2023**, *12*, 2050. [CrossRef]
2. Ali, J.; Zafar, M.H.; Hewage, C.; Hassan, S.R.; Asif, R. The advents of ubiquitous computing in the development of smart cities—A review on the internet of things (IoT). *Electronics* **2023**, *12*, 1032. [CrossRef]
3. Rajawat, A.S.; Goyal, S.; Chauhan, C.; Bedi, P.; Prasad, M.; Jan, T. Cognitive adaptive systems for industrial internet of things using reinforcement algorithm. *Electronics* **2023**, *12*, 217. [CrossRef]
4. Wagh, M.D.; H, R.; Kumar, P.S.; Amreen, K.; Sahoo, S.K.; Goel, S. Integrated microfluidic device with MXene enhanced laser-induced graphene bioelectrode for sensitive and selective electroanalytical detection of dopamine. *IEEE Sensors J.* **2022**, *22*, 14620–14627. [CrossRef]
5. Zhu, Z.; Zeng, F.; Pu, Z.; Fan, J. Conversion electrode and drive capacitance for connecting microfluidic devices and triboelectric nanogenerator. *Electronics* **2023**, *12*, 522. [CrossRef]
6. Hu, K.; Ibrahim, M.; Chen, L.; Li, Z.; Chakrabarty, K.; Fair, R. Experimental demonstration of error recovery in an integrated cyberphysical digital-microfluidic platform. In Proceedings of the 2015 IEEE Biomedical Circuits and Systems Conference (BioCAS), Atlanta, GA, USA, 22–24 October 2015; pp. 1–4.
7. Ibrahim, M.; Chakrabarty, K. Cyberphysical adaptation in digital-microfluidic biochips. In Proceedings of the 2016 IEEE Biomedical Circuits and Systems Conference (BioCAS), Shanghai, China, 17–19 October 2016; pp. 444–447.
8. Ibrahim, M.; Gorlatova, M.; Chakrabarty, K. The internet of microfluidic things: Perspectives on system architecture and design challenges. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–8.
9. Lin, C.S.; Tsai, Y.C.; Hsu, K.F.; Lee, G.B. Optimization of aptamer selection on an automated microfluidic system with cancer tissues. *Lab Chip* **2021**, *21*, 725–734. [CrossRef] [PubMed]
10. Azizipour, N.; Avazpour, R.; Sawan, M.; Ajji, A.; H. Rosenzweig, D. Surface optimization and design adaptation toward spheroid formation on-chip. *Sensors* **2022**, *22*, 3191. [CrossRef]
11. Bettenfeld, R.; Claudel, J.; Kourtiche, D.; Nadi, M.; Schlauder, C. Design and modeling of a device combining single-cell exposure to a uniform electrical field and simultaneous characterization via bioimpedance spectroscopy. *Sensors* **2023**, *23*, 3460. [CrossRef]
12. Kim, U.; Oh, B.; Ahn, J.; Lee, S.; Cho, Y. Inertia–acoustophoresis hybrid microfluidic device for rapid and efficient cell separation. *Sensors* **2022**, *22*, 4709. [CrossRef]
13. Thorsen, T.; Maerkl, S.J.; Quake, S.R. Microfluidic large-scale integration. *Science* **2002**, *298*, 580–584. [CrossRef]
14. Chen, Z.; Huang, X.; Guo, W.; Li, B.; Ho, T.Y.; Schlichtmann, U. Physical synthesis of flow-based microfluidic biochips considering distributed channel storage. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 1525–1530.
15. Liu, C.; Huang, X.; Li, B.; Yao, H.; Pop, P.; Ho, T.Y.; Schlichtmann, U. DCSA: Distributed channel-storage architecture for flow-based microfluidic biochips. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *40*, 115–128. [CrossRef]

16. Zhu, Y.; Huang, X.; Li, B.; Ho, T.Y.; Wang, Q.; Yao, H.; Wille, R.; Schlichtmann, U. Multicontrol: Advanced control-logic synthesis for flow-based microfluidic biochips. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *39*, 2489–2502. [CrossRef]
17. McDaniel, J.; Parker, B.; Brisk, P. Simulated annealing-based placement for microfluidic large scale integration (mLSI) chips. In Proceedings of the 2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC), Playa del Carmen, Mexico, 6–8 October 2014; pp. 1–6.
18. Li, M.; Tseng, T.M.; Ma, Y.; Ho, T.Y.; Schlichtmann, U. VOM: Flow-path validation and control-sequence optimization for multilayered continuous-flow microfluidic biochips. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–8.
19. Araci, I.E.; Quake, S.R. Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves. *Lab Chip* **2012**, *12*, 2803–2806. [CrossRef]
20. Hong, J.W.; Quake, S.R. Integrated nanoliter systems. *Nat. Biotechnol.* **2003**, *21*, 1179–1183. [CrossRef] [PubMed]
21. Perkel, J.M. Life science technologies: Microfluidics—Bringing new things to life science. *Science* **2008**, *322*, 975–977. [CrossRef]
22. Minhass, W.H.; Pop, P.; Madsen, J.; Blaga, F.S. Architectural synthesis of flow-based microfluidic large-scale integration biochips. In Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems, Raleigh, NC, USA, 7–12 October 2012; pp. 181–190.
23. Huang, X.; Ho, T.Y.; Guo, W.; Li, B.; Schlichtmann, U. MiniControl: Synthesis of continuous-flow microfluidics with strictly constrained control ports. In Proceedings of the 56th Annual Design Automation Conference 2019, San Francisco, CA, USA, 2–6 June 2019; pp. 1–6.
24. Huang, X.; Ho, T.Y.; Chakrabarty, K.; Guo, W. Timing-driven flow-channel network construction for continuous-flow microfluidic biochips. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *39*, 1314–1327. [CrossRef]
25. Huang, X.; Pan, Y.; Zhang, G.L.; Li, B.; Guo, W.; Ho, T.Y.; Schlichtmann, U. PathDriver: A path-driven architectural synthesis flow for continuous-flow microfluidic biochips. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual Event USA, 2–5 November 2020; pp. 1–8.
26. Huang, X.; Pan, Y.; Zhang, G.L.; Li, B.; Guo, W.; Ho, T.Y.; Schlichtmann, U. PathDriver+: Enhanced path-driven architecture design for flow-based microfluidic biochips. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2021**, *41*, 2185–2198. [CrossRef]
27. Programmable Microfluidics. Available online: http://groups.csail.mit.edu/cag/biostream/ (accessed on 9 June 2023).
28. Kim, M.C.; Lee, D.J.; Markov, I.L. SimPL: An effective placement algorithm. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2011**, *31*, 50–60. [CrossRef]
29. Spindler, P.; Schlichtmann, U.; Johannes, F.M. Kraftwerk2—A fast force-directed quadratic placement approach using an accurate net model. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2008**, *27*, 1398–1411. [CrossRef]
30. Fruchterman, T.M.; Reingold, E.M. Graph drawing by force-directed placement. *Softw. Pract. Exp.* **1991**, *21*, 1129–1164. [CrossRef]
31. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
32. Lewis, H.R. Michael R. ΠGarey and David S. Johnson. Computers and intractability. A guide to the theory of NP-completeness. W. H. Freeman and Company, San Francisco 1979, x + 338 pp. *J. Symb. Log.* **1983**, *48*, 498–500. [CrossRef]
33. Mitchell, J.E.; Floudas, C.; Pardalos, P. Integer programming: Branch and cut algorithms. *Encycl. Optim.* **2009**, *2*, 519–525.
34. Jiang, S.; Song, Z.; Weinstein, O.; Zhang, H. A faster algorithm for solving general LPs. In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual, Italy, 21–25 June 2021; pp. 823–832.