Automatic Design for Modular Microfluidic Routing Blocks

Philipp Ebner*, Maria Emmerich†, Eric Safai‡, Aniruddha Paul§, Mathieu Odijk¶,
Joshua Loessberg-Zahl‡§ and Robert Wille†‖
*Institute for Integrated Circuits and Quantum Computing, Johannes Kepler University Linz, Austria
†Chair for Design Automation, Technical University of Munich, Germany
‡Department of Bioengineering Technologies, University of Twente, The Netherlands
§BIOS Lab on Chip Group, Mesa+ Institute of Nanotechnology, University of Twente, The Netherlands
¶Integrated Devices and Systems (IDS), University of Twente, The Netherlands
‖Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
philipp.ebner@jku.at, maria.emmerich@tum.de, { e.r.safai, a.paul, m.odijk, j.t.loessberg-zahl }@utwente.nl,
robert.wille@tum.de
https://www.cda.cit.tum.de/research/microfluidics/

***Abstract*—Microfluidics is a rapidly growing field that aims to simplify complex analytical procedures by moving them to small-scale devices. A particularly interesting application of microfluidics are so-called *Organs-on-Chips*, i.e., microfluidic devices that mimic the structure and function of human organs and, therefore, allow studying the effects of drugs and diseases. Recent recognition of the need for standardization in this domain has led to the generation and uptake of new ISO standards—providing the basis of modular and reusable *microfluidic building blocks* that allow for various organ-on-chip setups. However, designing these building blocks, especially so-called *routing blocks* that interconnect pumps, cell cultures, and other modules is a cumbersome, repetitive task that is still conducted manually. In this work, we propose a design and routing method that significantly simplifies the design of such routing blocks by fully automating the process of interconnecting components of a microfluidic chip. The evaluation of physical, fabricated routing blocks that were designed using the proposed method showcases its feasibility in real-world applications and its potential to reduce design effort and time significantly. In order to make the work accessible to the microfluidic community, we provide implementations of the resulting methods in the form of a user-friendly, interactive online tool, provided as part of the *Munich Microfluidic Toolkit* (MMFT).**

***Index Terms*—Microfluidics, Routing Block, Organ-on-Chip, Design Method**

I. Introduction

Microfluidic chips, typically called *Lab-on-a-Chip* (LoC), can manipulate fluids at the microscale level, allowing for precise control and analysis of biological and chemical samples [1]–[3]. LoCs have various applications, such as drug delivery, point-of-care diagnostics, or environmental monitoring [4]–[6].

One promising application of microfluidics is provided through the development of *Organs-on-Chip* [7]–[11], which are microscale models of human organs that mimic their physiological functions. These devices allow researchers to study the effects of drugs, toxins, and diseases on specific organs in a more accurate and controlled manner. It is hoped that they can largely replace animal testing and, therefore, enable a more ethical approach in drug development.

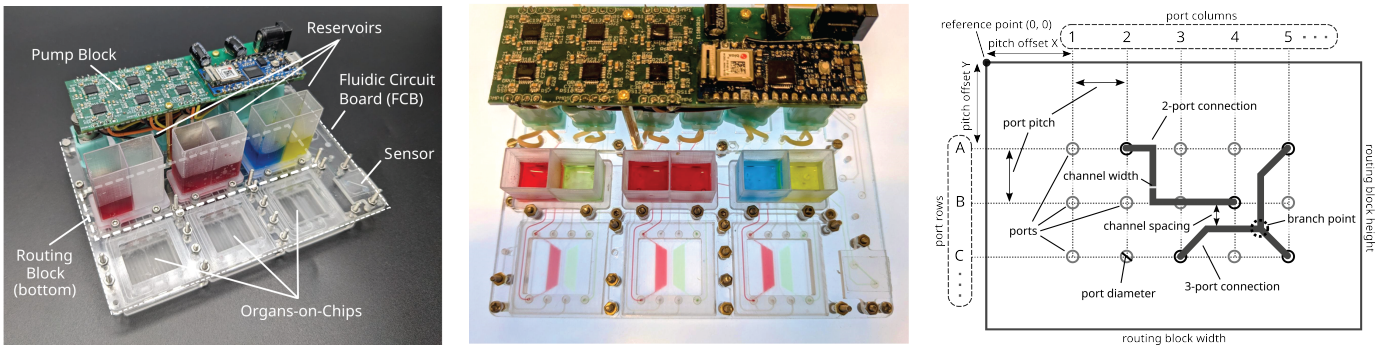
Until recently, microfluidics in general, and the organs-on-chip subfield in particular, suffered from a lack of standardization, leading to a limited interoperability of different devices. Fortunately, this is beginning to change with the introduction of *ISO 22916:2022* [12], an international standard that aims to improve this and, therefore, enable more complex and interconnected microfluidic devices. Derived from this ISO standard, additional design rulesets, such as the *Translational Organ-on-chip Platform* (TOP) *Design Rules* (TDRs), were recently introduced [13].

TOP is an open-source design architecture for modular microfluidics to enable the integration of *microfluidic building blocks* (i.e., modules) essential for organ-on-chip experiments including organs-on-chips, pumps, sensors, liquid storage – onto a microtiter-sized *Fluidic Circuit Board* (FCB), fitting parts of the *ANSI-SLAS* standards [14] for microtiter plates. Any microfluidic component that implements the ISO-compliant TDRs is compatible. In this work, the *Stand-Alone Reconfigurable and Translational* (STARTER) platform [15], a TOP-based platform, is considered as a representative implementation of these standards. STARTER increases device interoperability and design freedom for the user due to its modularity, reconfigurability, and material-agnostic interfacing.

The platform's architecture allows for experiment flexibility through a central component, the so-called *routing block*, which dictates fluidic connections between the interfaced components. Such connections are established by microfluidic channels that carry the fluid between different ports (i.e., interfaces to other components) of the routing block. Depending on the complexity of the experiment setup, and, therefore, on the number and characteristics of desired connections to be implemented, designing these routing blocks constitutes a complex task.

Despite this complexity, the design of routing blocks is still conducted manually, resulting in a time-consuming and repetitive process. Although there are many design automation tools for certain microfluidic applications [16], e.g., for flow-based chips [17]–[24], valve-controlled chips [25]–[29] or digital biochips [30]–[33], they do not yet consider the constraints of the ISO 22916 standard due to its novelty. While there is little preliminary work that does consider the standard [34], it is not specific enough to be employable for routing blocks and lacks interfaces that make it accessible to the microfluidic community [35]. Ultimately, for the problem presented herein, no automatic process exists yet.

In this work, we present a design method that can solve this task in a fully automatic fashion. To this end, we employ a procedure that takes advantage of the design characteristics of routing blocks in order to efficiently route microfluidic channel connections. The proposed procedure ensures that routing block designs can be generated in real-time. Additionally, we provide a user-friendly online tool (available at https://github.com/cda-tum/mmft-routing-block-channel-router) that allows users to design routing blocks in an intuitive, push-button fashion as part of the *Munich Microfluidic Toolkit* (MMFT) [36]. To bridge the gap to the community, we provide options to import routing block configurations from external file formats that are predominantly used in the microfluidic community. In the end, we also provide exports to common CAD formats such as DXF that can then be directly used in fabrication or further refined.



(a) STARTER platform (b) Multi-Organ-on-Chip cultures (c) Routing block

Fig. 1: STARTER platform and routing blocks

We evaluate the applicability and performance of the resulting method and tool by both performance benchmarks as well as an experimental case study. To show that the proposed tool can indeed generate designs for the introduced task in near real-time, we provide benchmarks for 5000 randomly generated, generic routing blocks. Finally, we demonstrate feasibility for real-world applications by testing fabricated routing blocks that have been designed using the proposed tool.

The remainder of this work is structured as follows: First, we review the context of the given routing problem in Sec. II. In Sec. III, we present the proposed method. Fabricated examples (designed using the method and tool), performance benchmarks, and the resulting tool are discussed in Sec. IV, before the paper is concluded in Sec. V.

II. Background and Design Task

In order to introduce the context of the proposed method, we first review the characteristics of the STARTER platform routing blocks. Afterwards, we sketch the resulting design task and main idea behind the proposed solution.

A. STARTER and Routing Blocks

In this work, we consider the STARTER platform [15] as a representative implementation of the ISO 22916 standard and *Translational Organ-on-chip Platform* (TOP) *Design Rules* (TDRs). STARTER consists of an *Fluidic Circuit Board* (FCB) that houses key components for organ-on-chip experiments. As an example, Fig. 1a depicts a STARTER system that accommodates a pump block, 3 reservoirs, a sensor, 3 organ-on-chip cultures, and a routing block that is mounted on the bottom of the FCB. Fig. 1b shows an active STARTER system where red loops are recirculating fluid and the green path is a solution of blue and yellow perfused through all 3 organ cultures. All these components interface fluidically with the FCB using silicone o-rings and are held in place with mechanical clamps.

In prior work [37], [38], a new FCB was almost always required for each new use case, as each FCB contains a fixed set of fluidic circuits. Thanks to the ISO standard, the routing block component of STARTER allows the same FCB to realize many different fluidic circuits. Here, the fluidic channels simply connect the ports of other microfluidic building blocks, e.g., pumps, sensors, or organ cultures, to ports in a single central interfacing area on the bottom face of the FCB, highlighted in Fig. 1a, where a single, compact routing block is located. Fluidic connections established within this routing block, therefore, fully determine the fluidic circuits present in the STARTER system. By simply manufacturing a new routing block, experiments can be fully (re)designed while using the same set of microfluidic building blocks and organs-on-chips.

The properties of routing blocks are derived from the TDRs [13] as well as the ISO 22916 standard [12]. These standards govern various geometric aspects, which are illustrated in Fig. 1c. More precisely:

a) Size: Routing blocks have a rectangular shape with a certain *width* and *height*, as depicted in Fig. 1c, where both width and height are restricted to multiples of $15\,\mathrm{mm}$. Currently, routing blocks of two different sizes are used in the STARTER platform: $30\,\mathrm{mm} \times 15\,\mathrm{mm}$ and $105\,\mathrm{mm} \times 15\,\mathrm{mm}$ [15]. However, in principle, other sizes are possible as well. The upper-left corner is called the *reference point* with coordinate $(0, 0)$.

b) Ports: Routing blocks interoperate with other components of microfluidic chips via circular ports, i.e., openings on top or at the bottom that allow for fluid exchange with other components. The permitted locations of ports are aligned in rows and columns on a grid, as illustrated in Fig. 1c. Rows of ports are identified by letters $(A, B, C, \ldots)$, whereas columns are identified by numbers $(1, 2, 3, \ldots)$. As a result, each port has a unique *port identifier*, e.g., the first column in Fig. 1c consists of the three port locations $A1$, $B1$, and $C1$. While Fig. 1c illustrates all possible port locations for the given routing block, ports that are not end points of connections (e.g., all except $A2$, $A5$, $B4$, $C3$, and $C5$) have no function and, therefore, would not actually be fabricated. The uniform distance between rows and columns of ports is called *port pitch*. The grid itself has a certain offset relative to the reference point in both directions, i.e., the upper-left port $A1$ has coordinates of *pitch offset X* and *pitch offset Y*. The size of ports is given by their *diameter*.

c) Connections: Connections between ports are realized by channels of either rectilinear (i.e., vertical and horizontal segments) or octilinear (i.e., also diagonal segments) *layout*. These channels have a certain *width* as well as a required minimum *spacing* towards other channels in order to allow for defect-free fabrication of the resulting design. A single connection can encompass between two and four ports. 2-port connections are simple point-to-point connections via a single channel, e.g., the 2-port connection in Fig. 1c connects ports $A2$ and $B4$ using a rectilinear layout. For 3-port and 4-port connections, the channels leading to each port join at a common *branch point*, e.g., the 3-port connection in Fig. 1c connects ports $A5$, $C3$, and $C5$ with three branches (i.e., channel sections) that meet at the indicated branch point using an octilinear layout. Since unused ports (i.e., ports that are not part of a connection) are not actually present on the fabricated routing block, channels may be routed through their locations (e.g., the 2-port connection crosses the location of unused port $B3$).

TABLE I: Input Parameters

Symbol		Description
b_x, b_y	...	block width and height [mm]
ch_w, ch_{sp}	...	channel width and channel spacing [mm]
p	...	pitch [mm]
po_x, po_y	...	pitch offsets [mm], $po_x \geq p$, $po_y \geq p$
pd	...	port diameter [mm]
l	...	layout (rectilinear or octilinear)
co_n	...	number of connections
co^i	...	i-th connection, $0 \leq i < co_n$
↳ cop^i_n	...	number of ports of co^i, $2 \leq cop^i_n \leq 4$
↳ $cop^{i,k}$	...	k-th port of co^i, $0 \leq k < cop^i_n$
↳ $cop^{i,k}_x$, $cop^{i,k}_y$	...	column and row of the port[1]
↳ brp^i	...	connection branch port (optional)
↳ brp^i_x, brp^i_y	...	column and row of the branch port[1] (optional)

B. Design Task and Main Idea

The design task that naturally emerges from designing a routing block is to find and draw the channel connections introduced above while, at the same time, following the ISO standard, i.e., respecting the given geometric requirements. Until now, this task was exclusively done manually, resulting in a time-consuming and repetitive workflow. In this work, we propose a design automation method that can solve this task not only in a fully automated fashion but is also fast enough to deliver near-instant results.

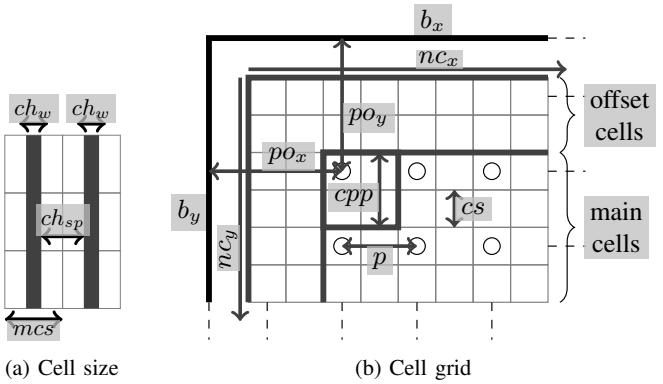
The expected input parameters largely correspond to the aspects introduced in Sec. II-A and Fig. 1c. Table I provides a detailed list of these parameters. More precisely, routing block size (width b_x and height b_y) and channel size (width ch_w and required spacing ch_{sp}) must be provided, as well as pitch size p and offsets $po_{x/y}$. Additionally, port diameters pd are part of the input as well as the layout l that is used for all channels, either rectilinear or octilinear. Finally, a list of connections to be routed has to be provided, where each connection co^i consists of 2 to 4 connection ports $cop^{i,k}$, and an optional branch point brp^i in the form of another, unused port location[1].

Given this input, we propose a design automation method that can route channel connections between ports automatically and efficiently. More precisely, the proposed procedure consists of five major steps:

1) *Discretization of Search Space*: The area of the routing block is partitioned into a uniform cell grid.
2) *Area Reservation*: Cells around ports are reserved for the corresponding connection.
3) *Branch Point Computation*: The cells where channels of 3-port and 4-port connections should meet are computed.
4) *Reordering of Connections*: Connections to be routed are ordered in a specific way to reduce mutual obstruction.
5) *Sequential Routing*: Connections are routed based on A*.

The following sections explain these steps in detail and demonstrate that the proposed solution indeed provides fast and reliable results.

[1]We assume that port identifiers are given as indexed columns and rows here (e.g., $cop^{i,k}_{x/y} = (3, 0)$ for $A4$, $cop^{i,k}_{x/y} = (0, 3)$ for $D1$, etc).



(a) Cell size (b) Cell grid

Fig. 2: Discretization of search space

III. Proposed Method

In this section, we cover the details of the proposed design automation solution, consisting of the five steps *Discretization of Search Space*, *Area Reservation*, *Branch Point Computation*, *Reordering of Connections*, and *Sequential Routing*.

A. Discretization of Search Space

In order to simplify the design process, the area of the routing block is split into a uniform grid of square cells, as illustrated in Fig. 2b. To achieve this, the size of these cells is determined first. The *minimum cell size* mcs (depicted in Fig. 2a) is derived from the channel width ch_w and spacing ch_{sp} such that two adjacent cells may be occupied by parallel channels without violating spacing constraints, i.e.,

$$mcs = ch_w + ch_{sp}. \tag{1}$$

Consequently, the number of cells *between* ports on the port grid along each axis, i.e., the *cells per port pitch cpp* (depicted in Fig. 2b), can be computed by

$$cpp = \left\lfloor \frac{p}{mcs} \right\rfloor. \tag{2}$$

However, the previously computed cell size mcs is merely an ideal value that is implementable only if the cell grid aligns perfectly with the port grid, i.e., each port must be located at the center of a cell. If this is not yet the case, the actual *cell size* cs is computed by

$$cs = \frac{p}{cpp}. \tag{3}$$

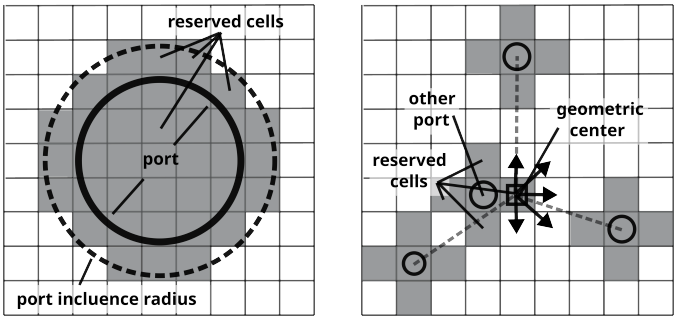
The resulting cell grid with cells of size cs now aligns perfectly with the port grid.

Subsequently, it needs to be determined how many cells are actually needed to discretize the entire routing block. As depicted in Fig. 2b, there are cells that cover the port grid, the *main cells*. However, there may still be unused space left towards the boundaries of the routing block, especially if the pitch offsets $po_{x/y}$ are significantly larger than the pitch p itself. Hence, the number of additional *offset cells* is computed as well.

To this end, the number of rows and columns of port slots are determined. More precisely, the number of columns of available port slots np_x and rows of port slots np_y (cf. Fig. 1c) depend on the dimensions of the routing block, b_x and b_y, as well as on the offsets of the port grid, po_x and po_y[2], i.e.,

$$np_{x/y} = \left\lfloor \frac{b_{x/y} - 2po_{x/y}}{p} \right\rfloor + 1. \tag{4}$$

[2]By $v_{x/y}$ we denote that both values v_x and v_y have to be computed separately.



(a) Area reservation (b) Branch point computation

Fig. 3: Preparatory steps

Consequently, the number of main cells $mcells_{x/y}$ (along each axis) that cover the main area around the ports (illustrated in Fig. 2b) is then determined by[2]

$$mcells_{x/y} = np_{x/y} \cdot cpp + 1 - cpp \bmod 2. \tag{5}$$

Next, the number of additional offset cells $ocells_{x/y}$ towards each boundary (also illustrated in Fig. 2b) with columns $ocells_x$ and rows $ocells_y$ is determined by[2]

$$ocells_{x/y} = \max\left\{\left\lfloor \frac{po_{x/y} - \left\lfloor \frac{cpp}{2} \right\rfloor \cdot cs - \frac{cs+ch_{sp}}{2}}{cpp} \right\rfloor, 0\right\}. \tag{6}$$

Finally, the total number of cells along each axis (nc_x and nc_y, cf. Fig. 2b) is given by[2]

$$nc_{x/y} = mcells_{x/y} + 2 \cdot ocells_{x/y}. \tag{7}$$

In other words, the resulting grid consists of cells $c^{i,j}$ with $0 \leq i < nc_x$ and $0 \leq j < nc_y$. By $c_x^{i,j}$ and $c_y^{i,j}$, we denote the cell's center coordinates.

Ultimately, a port location $cop^{i,k}$ can be mapped to a cell $c^{i,j}$ with a *port-to-cell function* ptc, i.e.,

$$ptc\left(cop^{i,k}\right)_{i/j} = \left\lfloor \frac{cpp}{2} \right\rfloor + cpp \cdot cop^{i,k}_{x/y} + ocells_{x/y}. \tag{8}$$

With the computed cell grid as a result, in principle, the routing task is now reduced to finding paths on the cell grid between cells that represent the positions of connection ports. However, before the actual routing takes place, some additional preparatory steps are necessary in order to minimize mutual interference of connections.

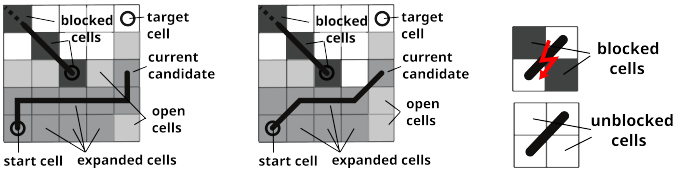
B. Area Reservation

In the previous step, only the cell location of (the center of) ports was considered. However, ports may also be larger than a single cell (depicted in Fig. 3a), depending on the value of the port diameter pd (e.g., when channels are small compared to ports). To ensure that no foreign connections are routed through ports, all cells that are too close to a port of a connection are marked as *reserved* to be usable only by that connection. To this end, we define a port influence radius pir that encompasses all cells that are too close to the port, i.e.,

$$pir = \frac{pd}{2} + \frac{ch_w}{2} + ch_{sp}. \tag{9}$$

All cells with centers within that range of a port center cell $c^{ptc\left(cop^{i,k}\right)}$ are marked as reserved for the corresponding connection (illustrated in Fig. 3a), i.e., all cells c^{jl} that fulfill the condition

$$d\left(c^{ptc\left(cop^{i,k}\right)}, c^{jl}\right) < pir, \tag{10}$$



(a) Rectilinear (b) Octilinear (c) Check

Fig. 4: Routing

where $d\left(c^{ij}, c^{kl}\right)$ is the Euclidean distance between two cells c^{ij} and c^{kl}, i.e.,

$$d\left(c^{ij}, c^{kl}\right) = \sqrt{\left(c_x^{ij} - c_x^{kl}\right)^2 + \left(c_y^{ij} - c_y^{kl}\right)^2}. \tag{11}$$

C. Branch Point Computation

As previously introduced in Sec. II-A, *3-port* or *4-port* connections are implemented in such a way that each port connects to a common branch point. Such branch points can be either user-specified as part of the input (cf. Table I), or automatically computed if not provided. With regard to the algorithm, a branch point simply translates to a cell that is reserved for that particular connection.

The following steps are performed for each connection: If the branch point is given as input, it directly maps to a cell (cf. Eq. 8 in Sec. III-A). Otherwise, a suitable branch cell is automatically computed for 3-port and 4-port connections. To this end, a first candidate is the centroid, i.e., the cell that is the geometric center of all of the connection's port cells (illustrated in Fig. 3b). However, this cell may already be reserved for other connections, e.g., if it lies within another port's influence radius. In that case, a breadth-first search is conducted to find the nearest unoccupied cell (indicated by the arrows in Fig. 3b). In either case, the eventually determined cell is marked as reserved for the connection.

D. Reordering of Connections

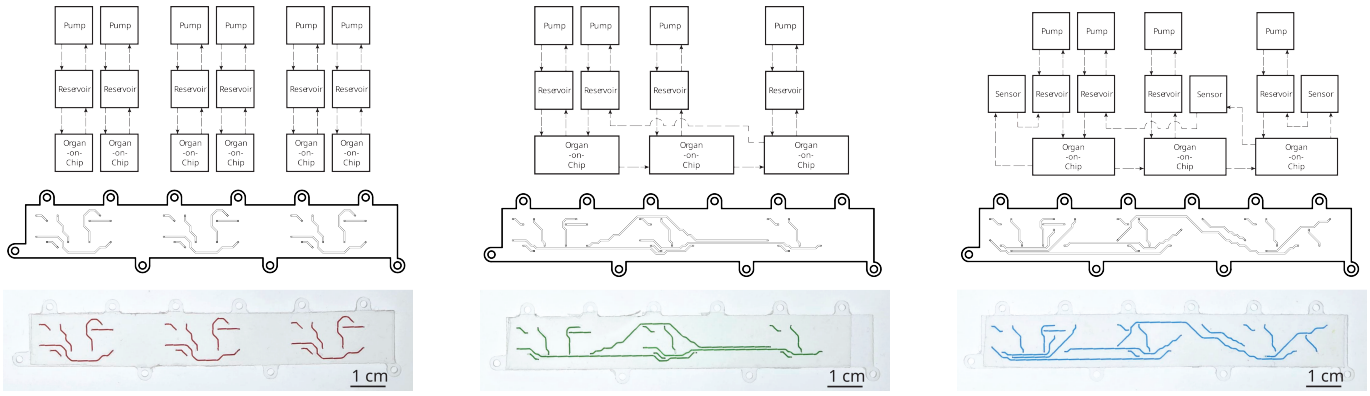
Since connections are routed sequentially, the order in which routing takes place has an impact on the quality of the results. As a reasonable choice in the context of this work, we order the connections to be routed in the following way.

Connections with branch points are routed first. In fact, each branch is treated like a single point-to-point connection. Thus, branches are ordered according to their Euclidean distance between their end points, such that shorter branches are routed first. After branched connections, 2-port connections without branch points are routed in the same order: Connections with shorter direct distances between their end ports are routed first.

E. Sequential Routing

Finally, the actual routing is conducted sequentially for each connection and for each branch of a connection. To this end, the shortest path of cells from a start cell to a target cell (i.e., a port or branch point), is determined by conducting an A* search. To this end, an *open list* of observed but not yet visited cells is maintained, originally containg just the start cell (examples of which are marked as open cells in Fig. 4). Similarly, a *closed list* of all already expanded cells is maintained as well (examples of which are marked as expanded cells in Fig. 4). Fig. 4a and Fig. 4b exemplarily illustrate the progression of the search for rectilinear and octilinear layouts, respectively.

In each step, the cell from the open list that is currently the *most promising* candidate for finding the shortest path to the target cell is selected. In this context, *most promising* is measured as a combination



(a) Parallel cultures (b) MOOC cultures (c) MOOC cultures with sensors

Fig. 5: Fabricated routing blocks

of the already covered path length to that cell and an estimate of the remaining path length (i.e., a heuristic). As a heuristic, the Euclidean distance between two cells $c^{i,j}$ and $c^{k,l}$ (i.e., between the cell on the open list and the target cell) is used (cf. Eq. 11). The thereby selected candidate is *expanded*, meaning that the suitable successor cells are selected.

For rectilinear layouts (cf. 4a), there are three successor cells: the straight-ahead cell with respect to the previous pathway and the two cells to the left and right that implement a 90° turn. However, blocked cells or cells reserved for other connections are discarded.

For octilinear layouts (cf. 4b), there are also three successor cells: Again, the straight-ahead cell with respect to the previous pathway and the two cells to the diagonal left and diagonal right that implement a 45° turn. Likewise, blocked cells or cells reserved for other connections are discarded. However, there is an additional corner case to be aware of, as illustrated in Fig. 4c: For a diagonal move to be valid, the two adjacent cells must not be blocked or reserved by other channels in order to ensure proper channel spacing.

After this expansion step, the successor cells are placed on the open list while the current candidate is moved to the closed list of already expanded cells and will not be considered again. A valid path is found when the target cell is expanded. Finally, the resulting sequence of cells that realizes the connection is marked as blocked for future searches. If the search ends without a result (i.e., the open list becomes empty), no path can be found for the current connection, and, therefore, the algorithm can only return a partial result. Eventually, carrying out this procedure for all connections results in channels for all connections that conform to the design constraints of routing blocks.

IV. Performance and Resulting Tool

In the following, we analyze the performance of the proposed method on fabricated examples and a large set of benchmarks as well as showcase the applicability of the resulting tool.

A. Fabricated Examples and Performance

Benchmarks evaluating the proposed method were conducted for both fabricated routing blocks and larger generic cases. All benchmarks were conducted natively on an *AMD Ryzen 7 8840HS*.

a) Fabricated Cases: In order to analyze the quality of the generated routing blocks, three designs generated by the proposed method were fabricated and tested. For each routing block, Fig. 5 shows the experimental setup diagram, the corresponding CAD file (generated by the proposed method), and the fabricated routing blocks that were micromilled from poly(methyl methacrylate) and liquid-filled with food dye. Each one of these routing blocks has a size of $105\,\mathrm{mm} \times 15\,\mathrm{mm}$, implements 14-18 connections, and orchestrates complex organ-on-chip cultures. More precisely:

- *Parallel cultures (Fig. 5a)*: Connects 6 different organ-on-chip cultures with 1 pump and 1 reservoir each.
- *MOOC (Multi-Organ-on-Chip) cultures (Fig. 5b)*: Connects 3 different organ-on-chip cultures with 4 pumps and 4 reservoirs.
- *MOOC+Sensors cultures (Fig. 5c)*: Features the same components as the MOOC routing block, with an additional 3 sensors.

These fabrications demonstrate that the designs generated by the proposed method, in terms of quality, do not differ from the previously handcrafted designs.

Moreover, in order to demonstrate that the proposed implementation can efficiently generate routing blocks for these *real-world* applications, the time needed to generate the designs for the fabricated routing blocks above has been measured. To this end, Table II shows the running times needed to generate the three different routing blocks, all of which have low running times of up to $\sim 2\,\mathrm{ms}$ (only octilinear layouts were fabricated). Overall, this impressively shows that the proposed method is capable of *automatically* generating the desired designs in negligible runtime and still providing the required quality for a successful fabrication and execution of experiments.

b) Generic Cases: To further demonstrate the broad applicability of the proposed method, we considered further generic benchmarks. Results of those evaluations are provided in Table III which features 10 groups of generic, larger benchmarks (*G1* through *G10*) for both octilinear and rectilinear layouts. For each group and layout, 250 unique cases with randomly generated connection ports were examined, amounting to a total number of 5000 single benchmark cases. In addition to averaged running times for each group, Table III also provides minimum and maximum running times (i.e., the best and worst test cases), which demonstrates that there are no extreme outliers. All designs can be automatically generated in fractions of a second, i.e., in negligible time.

Attempting to mimic the distribution seen in real-world use cases, the total number of connections for each case is composed of $10\,\%$ 3-port connections and $10\,\%$ 4-port connections, with the remaining $80\,\%$ being 2-port connections. Routing blocks of three different sizes were tested, i.e., $30\,\mathrm{mm} \times 15\,\mathrm{mm}$ (*G1-G3*), $105\,\mathrm{mm} \times 15\,\mathrm{mm}$ (*G4-G7*), and $105\,\mathrm{mm} \times 105\,\mathrm{mm}$ (*G8-G10*).

Generally, the measured running times clearly scale with the size of the cell grid, indirectly proportional to the channel size, as well as with the number of connections. For the smallest routing block size (*G1-G3*) and up to 30 channels, average running times of up to

TABLE II: Fabricated cases performance benchmarks

Case	Routing Block Size		# Connections			Channel Size		Runtime [ms]
	width [mm]	height [mm]	total	3-port	4-port	width [mm]	spacing [mm]	
Parallel	105	15	18	0	0	0.4	0.3	0.20
MOOC	105	15	14	0	0	0.4	0.3	1.74
MOOC+Sensors	105	15	17	0	0	0.4	0.3	2.20

TABLE III: Generic cases performance benchmarks

Group	Routing Block Size		# Connections			Channel Size		Runtime Octilinear			Runtime Rectilinear		
	width [mm]	height [mm]	total	3-port	4-port	width [mm]	spacing [mm]	min [ms]	avg [ms]	max [ms]	min [ms]	avg [ms]	max [ms]
G1	30	15	10	1	1	0.4	0.4	0.02	0.04	0.09	0.02	0.05	0.13
G2	30	15	20	2	2	0.2	0.2	0.20	1.68	4.38	0.71	1.89	4.49
G3	30	15	30	3	3	0.1	0.1	11.16	32.80	65.79	10.26	22.62	42.44
G4	105	15	20	2	2	0.4	0.4	0.11	0.19	0.57	0.15	0.28	0.61
G5	105	15	30	3	3	0.2	0.2	4.73	13.61	24.03	5.48	11.45	18.34
G6	105	15	40	4	4	0.1	0.1	77.14	203.53	373.92	70.15	123.09	182.07
G7	105	15	50	5	5	0.1	0.1	89.79	188.98	311.11	70.35	120.57	188.74
G8	105	105	50	5	5	0.4	0.4	5.73	10.79	18.19	6.53	9.91	13.61
G9	105	105	100	10	10	0.2	0.2	152.67	254.68	393.63	123.92	174.96	232.01
G10	105	105	200	20	20	0.2	0.2	134.77	188.90	274.89	105.91	134.88	176.47

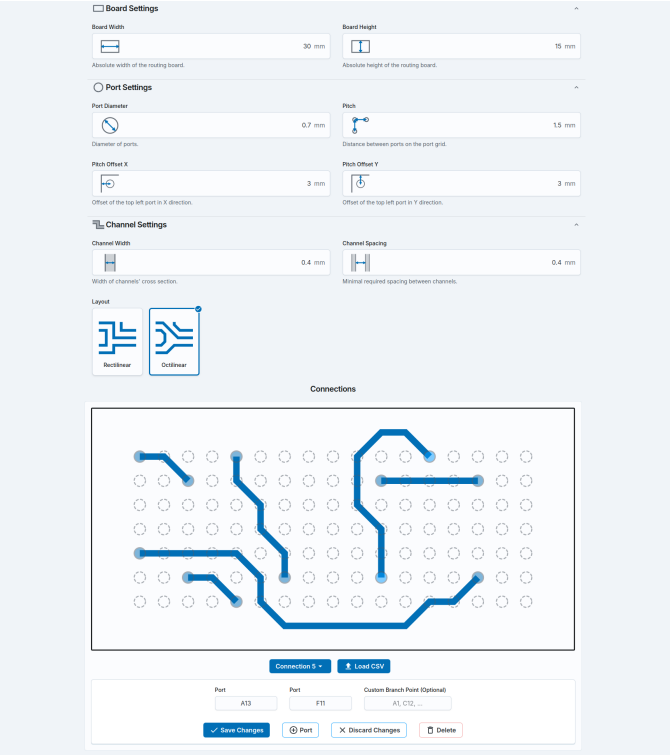


Fig. 6: Interface of the tool

$\sim$33 ms were measured. For the largest real-world routing blocks (*G4*-*G7*) and up to 50 channels, average running times of up to $\sim$200 ms were measured. The largest size (*G8*-*G10*) with up to 200 connections exceeds currently used routing blocks by far but was included to demonstrate the feasibility of the implementation to adapt to future, likely larger use cases. Still, a solution could be obtained for all presented cases within a very reasonable time, i.e., no single case took longer than $\sim$400 ms.

B. Applicability and Resulting Tool

Finally, we demonstrate the applicability of the proposed method. Originally, routing blocks were designed manually in CAD software starting from a routing block template. This design process needed to be repeated for each new application. Furthermore, human errors made in this manual design process were often only discovered after fabrication or, in the worst case, at experiment run time.

The proposed method helps speeding up the design process of routing blocks and reduces the risk of human errors. In order to make the implementations available to the microfluidic community, we provide a user-friendly online tool that is available at https://github.com/cda-tum/mmft-routing-block-channel-router as part of the Munich Microfluidic Toolkit (MMFT). Fig. 6 shows excerpts of the most important functionalities of the tool. It provides inputs that mirror the necessary parameters previously introduced in Table I. Connections can be created in a *click-and-point* fashion by selecting the desired ports in a provided preview of the routing block or by typing the corresponding port identifiers. Additionally, connections can also be imported from a CSV file. Finally, the tool offers the option to export the generated design of the routing block as DXF, a popular CAD format that can be used for fabrication.

Use of the routing tool streamlines the design process for routing block designers. The flexibility of the fluidic routing also enables simple addition of sensing elements external to the organs-on-chips, simplifying experiment iteration and forgoing the requirement of complex fabrications. The designer needs only to specify the desired connections rather than to draw the channel geometry themselves in CAD. By using this tool, a reduction from originally $\sim$2 h to merely $\sim$30 min was observed during the design and fabrication of the examples in Sec. IV-A (of course, depending on the complexity and on the prior experience of the designer). Errors were reduced as well, as designers could focus on simply checking that the correct connections were selected, rather than needing to extensively check the full channel geometry. Last but not least, this shift in focus encourages collaborations with those who may have less design experience.

V. Conclusion

In this work, we proposed a design method and corresponding tool for microfluidic routing blocks. This tool makes it possible to *automatically* generate designs of routing blocks with corresponding microfluidic channels that adhere to the associated geometric design constraints with just a few clicks. Real-world applicability and performance have been thoroughly examined through benchmarks and fabricated examples. The developed tool is available at https://github.com/cda-tum/mmft-routing-block-channel-router as part of the Munich Microfluidic Toolkit.

Acknowledgment

This work has partially been supported by BMK, BMDW, the State of Upper Austria in the frame of the COMET Programme managed by FFG, and the NWO-TTW Perspective Programme of the Dutch Research Council (NWO, project number P19-03) as part of the SMART Organ-on-Chip consortium.

References

[1] U. A. Gurkan, D. K. Wood, D. Carranza, L. H. Herbertson, S. L. Diamond, E. Du, S. Guha, J. Di Paola, P. C. Hines, I. Papautsky, S. S. Shevkoplyas, N. J. Sniadecki, V. K. Pamula, P. Sundd, A. Rizwan, P. Qasba, and W. A. Lam, "Next generation microfluidics: fulfilling the promise of lab-on-a-chip technologies," *Lab Chip*, 2024. [Online]. Available: http://dx.doi.org/10.1039/D3LC00796K

[2] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle, "Microfluidic Lab-on-a-Chip platforms: requirements, characteristics and applications," *Chemical Society Reviews*, 2010.

[3] P. S. Dittrich and A. Manz, "Lab-on-a-chip: microfluidics in drug discovery," *Nature Reviews Drug Discovery*, vol. 5, no. 3, p. 210, 2006.

[4] A. Fabozzi, F. Della Sala, M. di Gennaro, M. Barretta, G. Longobardo, N. Solimando, M. Pagliuca, and A. Borzacchiello, "Design of functional nanoparticles by microfluidic platforms as advanced drug delivery systems for cancer therapy," *Lab Chip*, vol. 23, pp. 1389–1409, 2023.

[5] A. E. Ongaro, Z. Ndlovu, E. Sollier, C. Otieno, P. Ondoa, A. Street, and M. Kersaudy-Kerhoas, "Engineering a sustainable future for point-of-care diagnostics and single-use microfluidic devices," *Lab on a Chip*, vol. 22, no. 17, pp. 3122–3137, 2022.

[6] P. Aryal, C. Hefner, B. Martinez, and C. S. Henry, "Microfluidics in environmental analysis: advancements, challenges, and future prospects for rapid and efficient monitoring," *Lab Chip*, 2024.

[7] M. Emmerich, P. Ebner, and R. Wille, "Automated Design for Multi-Organ-on-Chip Geometries," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[8] C. M. Leung, P. De Haan, K. Ronaldson-Bouchard, G.-A. Kim, J. Ko, H. S. Rho, Z. Chen, P. Habibovic, N. L. Jeon, S. Takayama *et al.*, "A Guide to the Organ-on-a-Chip," *Nature Reviews Methods Primers*, 2022.

[9] K. Ronaldson-Bouchard, D. Teles, K. Yeager, D. N. Tavakol, Y. Zhao, A. Chramiec, S. Tagore, M. Summers, S. Stylianos, M. Tamargo, B. M. Lee, S. P. Halligan, E. H. Abaci, Z. Guo, J. Jacków, A. Pappalardo, J. Shih, R. K. Soni, S. Sonar, C. German, A. M. Christiano, A. Califano, K. K. Hirschi, C. S. Chen, A. Przekwas, and G. Vunjak-Novakovic, "A multi-organ chip with matured tissue niches linked by vascular flow," *Nature Biomedical Engineering*, vol. 6, no. 4, pp. 351–371, 2022.

[10] M. B. Esch, H. Ueno, D. R. Applegate, and M. L. Shuler, "Modular, pumpless body-on-a-chip platform for the co-culture of GI tract epithelium and 3d primary liver tissue," *Lab on a Chip*, vol. 16, no. 14, pp. 2719–2729, 2016.

[11] A. Tajeddin and N. Mustafaoglu, "Design and Fabrication of Organ-on-Chips: Promises and Challenges," *Micromachines*, vol. 12, no. 12, p. 1443, 2021.

[12] "Microfluidic devices – Interoperability requirements for dimensions, connections and initial device classification," International Organization for Standardization, Geneva, CH, Standard, Mar. 2022.

[13] E. Safai, "Translational organ-on-chip platform (top) design rules (tdrs)," 2024. [Online]. Available: https://data.4tu.nl/datasets/2558bd4c-d7ad-4e17-bc54-8c335b4c1c01/2

[14] "ANSI SLAS 1-2004 (R2012)," https://www.slas.org/SLAS/assets/File/public/standards/ANSI_SLAS_1-2004_FootprintDimensions.pdf, 2012, accessed: 2025-04-12.

[15] A. Paul, E. R. Safai, L. E. de Heus, A. Vollertsen, K. Weijgertse, B. de Wagenaar, H. E. Amirabadi, E. van de Steeg, M. Odijk, A. van der Meer, and J. Loessberg-Zahl, "STARTER: A Stand-Alone Reconfigurable and Translational OoC Platform based on Modularity and Open Design Principles," *bioRxiv*, 2025. [Online]. Available: https://dx.doi.org/10.1101/2025.05.13.653800

[16] X. Huang, T.-Y. Ho, W. Guo, B. Li, K. Chakrabarty, and U. Schlichtmann, "Computer-aided design techniques for flow-based microfluidic lab-on-a-chip systems," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–29, 2021.

[17] Q. Wang, H. Zou, H. Yao, T.-Y. Ho, R. Wille, and Y. Cai, "Physical co-design of flow and control layers for flow-based microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1157–1170, 2017.

[18] A. Grimmer, W. Haselmayr, A. Springer, and R. Wille, "Design of Application-Specific Architectures for Networked Labs-on-Chips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[19] A. Grimmer, X. Chen, M. Hamidovic, W. Haselmayr, C. L. Ren, and R. Wille, "Simulation before fabrication: a case study on the utilization of simulators for the design of droplet microfluidic networks," *RSC Advances*, 2018.

[20] A. Grimmer, P. Frank, P. Ebner, S. Häfner, A. Richter, and R. Wille, "Meander Designer: Automatically Generating Meander Channel Designs," *Micromachines*, 2018.

[21] A. Grimmer, M. Hamidović, W. Haselmayr, and R. Wille, "Advanced Simulation of Droplet Microfluidics," *Journal on Emerging Technologies in Computing Systems (JETC)*, 2019.

[22] P. Pop, I. E. Araci, and K. Chakrabarty, "Continuous-flow biochips: Technology, physical-design methods, and testing," *Journal on Design & Test*, vol. 32, no. 6, pp. 8–19, 2015.

[23] P. Ebner, G. Fink, and R. Wille, "Channel Routing for Microfluidic Devices: A Comprehensive and Accessible Design Tool," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD)*, 2022.

[24] G. Fink, T. Mitteramskogler, M. A. Hintermüller, B. Jakoby, and R. Wille, "Automatic design of microfluidic gradient generators," *IEEE Access*, vol. 10, pp. 28 155–28 164, 2022.

[25] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1588–1601, 2017.

[26] A. Grimmer, B. Klepic, T.-Y. Ho, and R. Wille, "Sound valve-control for programmable microfluidic devices," in *Asia and South Pacific Design Automation Conference*, 2018.

[27] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer optimization for flow-based mVLSI microfluidic biochips," in *Int'l Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, 2014, pp. 1–10.

[28] K. Hu, T. Dinh, T. Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, 2016.

[29] J. McDaniel, B. Crites, P. Brisk, and W. H. Grover, "Flow-layer physical design for microchips based on monolithic membrane valves," *Journal on Design & Test*, vol. 32, no. 6, pp. 51–59, 2015.

[30] O. Keszocze, R. Wille, K. Chakrabarty, and R. Drechsler, "A General and Exact Routing Methodology for Digital Microfluidic Biochips," in *Int'l Conf. on Computer-Aided Design*, 2015.

[31] D. Grissom, C. Curtis, S. Windh, C. Phung, N. Kumar, Z. Zimmerman, O. Kenneth, J. McDaniel, N. Liao, and P. Brisk, "An open-source compiler and pcb synthesis tool for digital microfluidic biochips," *INTEGRATION, the VLSI journal*, vol. 51, pp. 169–193, 2015.

[32] M. Alistar and U. Gaudenz, "OpenDrop: An integrated do-it-yourself platform for personal use of biochips," *Bioengineering*, vol. 4, no. 2, p. 45, 2017.

[33] O. Keszocze, Z. Li, A. Grimmer, R. Wille, K. Chakrabarty, and R. Drechsler, "Exact routing for micro-electrode-dot-array digital microfluidic biochips," in *Asia and South Pacific Design Automation Conference*, 2017.

[34] P. Ebner and R. Wille, "Automatic Validation and Design of Microfluidic Devices Following the ISO 22916 Standard," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2024.

[35] J. McDaniel, W. H. Grover, and P. Brisk, "The case for semi-automated design of microfluidic very large scale integration (mvlsi) chips," in *Design, Automation and Test in Europe*, 2017, pp. 1793–1798.

[36] R. Wille, P. Ebner, M. Emmerich, and M. Takken, "The Munich Microfluidic Toolkit: Design Automation and Simulation Tools for Microfluidic Devices," in *Int'l Conf. on Computer-Aided Design*, 2025.

[37] M. N. S. de Graaf, A. Vivas, D. G. Kasi, F. E. van den Hil, A. van den Berg, A. D. van der Meer, C. L. Mummery, and V. V. Orlova, "Multiplexed fluidic circuit board for controlled perfusion of 3d blood vessels-on-a-chip," *Lab Chip*, 2023. [Online]. Available: http://dx.doi.org/10.1039/D2LC00686C

[38] A. Vivas, A. van den Berg, R. Passier, M. Odijk, and A. D. van der Meer, "Fluidic circuit board with modular sensor and valves enables stand-alone, tubeless microfluidic flow control in organs-on-chips," *Lab Chip*, 2022. [Online]. Available: http://dx.doi.org/10.1039/D1LC00999K