Automatic Validation and Design Generation of Modular Microfluidic Devices Based on ISO 22916

Philipp Ebner, Tim Scholl, and Robert Wille *Senior Member, IEEE*

***Abstract*—Microfluidics is an emerging technology for manipulating small fluid volumes for analytic purposes. *Labs-on-Chip* (LoC) are microfluidic devices that have found widespread applications in various fields, including medicine, biology, and chemistry, due to their ability to manipulate small fluid volumes effectively. The lack of common standards, however, has impeded the development of microfluidic chip verification and design automation methods. Fortunately, there have been new developments with the introduction of the ISO 22916:2022 standard, which contains specifications for a platform based on exchangeable modules connected via channels in a larger main chip. Various other constraints have to be considered as well, e.g., the attachment of modules requires clamps and pins, optical experiments and other applications may invoke routing exclusion zones, etc. In this work, we present a methodology that allows, within the framework of this ISO standard, automatic validation of microfluidic chip designs against these constraints. To this end, we utilize solvers for *Satisfiability Modulo Theories* (SMT). The resulting approach does not only validate the ISO compliance of a given design but also provides tool support for the completion of the design. A case study based on real-world use cases illustrates the applicability and viability of the presented solution. An open-source implementation of the resulting tool is available at https://github.com/cda-tum/mmft-iso-designer.**

***Index Terms*—Microfluidics, ISO 22916:2022, Lab-on-a-Chip, Validation, Chip Design**

I. Introduction

MICROFLUIDICS studies the behavior and manipulation of fluids at the micro- to picoliter scale [1]. This miniaturization enables us to minimize, combine, and automate fluidic processes such as mixing, heating, incubating, observing, etc., into compact, automated platforms that can replace bulky and costly lab equipment as well as substantially reduce human labor in many instances. The resulting *Labs-on-a-Chip* (LoCs) have found applications in a wide variety of areas in medicine, biology, and chemistry [2]–[6]. In particular for point-of-care tests and other diagnostics (such as those recently seen in the COVID-19 pandemic), microfluidic devices are the "go-to" technology [7], [8].

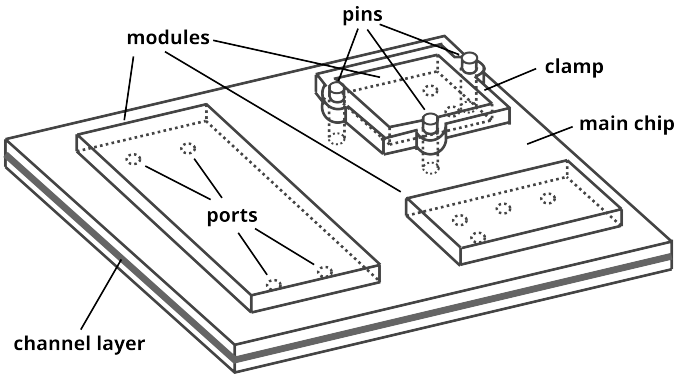
Despite this clear potential, the current state-of-the-art in microfluidic chip design remains far less advanced than we take for granted in, e.g., the design of electrical circuits and systems. Although in the past years, several methods for automating design and validation steps such as microfluidic network generation and dimensioning [9]–[13], optimizing flow-based devices [14]–[17], routing microfluidic channels [18]–[22], design automation for electrowetting [23]–[27], and frameworks such as Columba S [28], OpenDrop [29], and for the design of digital microfluidic biochips [30], etc. have been proposed, most of them have not found their way into the microfluidic community and practice yet. In fact, the established way of designing a new microfluidic device still mostly relies on manual work [31].

This rather unsatisfactory situation may have different reasons; however, a rather weak link between the design automation community and the microfluidics community, as well as a lack of standards that connect both domains, certainly rank among the top. For channel-based devices [32], this situation changed recently. Here, the lack of interoperability when combining microfluidic components has become a pressing issue due to the diversity of possible applications, technologies, and materials. Consequently, ISO 22916:2022 [33] was introduced to address this issue.

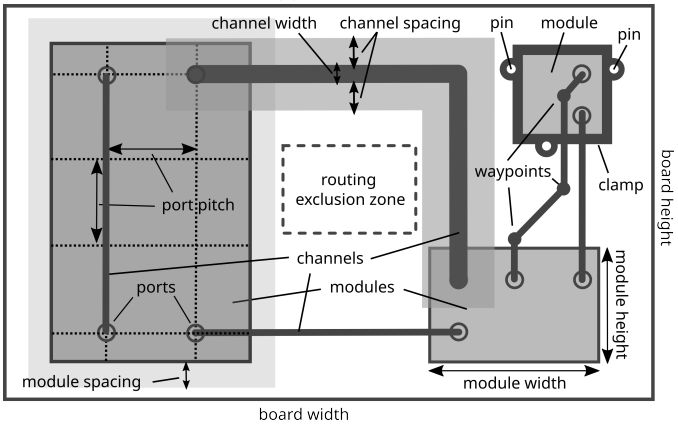
This standard provides a previously missing common description and specification of how microfluidic components, ports, and interconnecting channels should be assembled, constituting the basis for more complex and powerful designs [2]. Essentially, it introduces an architecture analogous to the purpose of manifolds [34] in fluidic applications. Consequently, as it is the first attempt to formalize these concepts for microfluidics, it also enables the development of broader and more generic design automation methods that can be applied to a wider range of applications. Nevertheless, this formalisation also leads to substantially more complex design tasks and, thus, makes design automation methods necessary. Designers must now respect a multitude of geometric constraints, e.g., the placement and dimensions of components [35], in addition to common design tasks (that are not covered by the ISO standard) such as ensuring minimal spacing of microfluidic channels to each other and the edges of the chip, area requirements of the physical attachment of components to the microfluidic chip (i.e., clamps and pins), and certain route avoidance requirements of optical experiments. Manually validating all of these requirements is both cumbersome and error-prone, and tooling support is still lacking due to the novelty of the standard.

In this work, we introduce an approach that automatically validates whether a given chip design not only adheres to relevant aspects of the ISO standard but also whether the placement and routing of modules and channels are valid with respect to their geometric properties[1]. Even further, it considers real-world features such as clamps and pins, which are used to fix single components on a common microfluidic circuit board, and can deal with the requirements of certain optical experiments. To achieve this, we employ an approach using *Satisfiability Modulo Theories* (SMT, [37]), where all relevant geometric properties are modeled as SMT constraints. Consequently, an SMT solver is able to automatically validate

[1]A preliminary version of this work has been presented before in [36].



(a) Overview



(b) Module and channel geometry

Fig. 1: Microfluidic chip with modules

that a given design is indeed compliant with this standard or if this is not the case. Moreover, the resulting approach can easily be extended to a tool that assists the designer in the completion of a chip design from a partial specification, automating parts of the placement and routing tasks. The implementations are available as part of the *Munich Microfluidics Toolkit* (MMFT) [38]. To assess and demonstrate the feasibility of the suggested methodology, we tested the resulting tool in scenarios that were inspired from real-world use cases.

The remainder of this paper is structured as follows: In Sec. II, we review the ISO 22916 standard as well as the resulting design task and outline the general idea of the proposed approach. In Sec. III, the necessary SMT encodings are discussed. Then, Sec. IV summarizes the implementation of the resulting tool. In Sec. V, we showcase the application of the resulting approach and discuss how this improves the design flow. Finally, we summarize our work in Sec. VI.

II. Motivation and General Idea

This section first reviews the ISO 22916 standard and briefly discusses the consequences and tasks that result from that for the design of microfluidic devices. Afterwards, motivated by that, we propose the general idea of a validation and design automation solution that aids the designer in these tasks.

A. The ISO 22916 Standard and Resulting Design Task

The ISO 22916:2022 standard has been defined by the *International Organization for Standardization* and is supposed to provide a standardized description for microfluidic devices consisting of a main chip board and multiple modules that are placed on top or on the bottom of the board, as illustrated in Fig. 1. The ISO standard primarily governs the geometry of the chip, modules, and ports. Additionally, in this work, we consider entities such as channels, routing exclusion zones, pins, and clamps, whose precise implementations are *not* governed by the standard, but are frequently used in such microfluidic systems [2]. More precisely:

a) Main Chip Board: The board is the main chip, where modules are placed and channels are routed inside. Its dimensions, i.e., its board width and board height (cf. Fig. 1b), are predetermined according to the ISO 22916 standard in order to ensure compatibility with certain laboratory equipment, such as microscopes.

b) Modules: Modules have a predetermined width and height (again, restricted by the ISO standard), as indicated in Fig. 1b, and are placed on the top or on the bottom of the main chip board. Furthermore, each module can be oriented along the axes in four possible directions. Additionally, each module must obey minimal spacing with respect to the board's boundaries and other modules (illustrated in Fig. 1b).

c) Clamps and Pins: A common way to attach modules to the main chip board is using clamps and pins, as shown for the upper-right module in Fig. 1b. Clamps are frames that encase modules, having multiple screw holes for securing them to the main board using pins. Pins are inserted through the main chip board, requiring the drilling of holes at their designated positions. These entities and their physical form are not part of the ISO 22916 standard; however, they are a necessity for the assembly of such a microfluidic device and need to be considered in the design process.

d) Ports: Each module may have several ports, which need to have a cleared perimeter without obstacles in order to be cleanly connected to the main board. Generally, this requires that a square with a predefined area be set around the port opening where no other ports or modules occur. The positions of these ports are spread across the module on a uniform grid (depicted in Fig. 1b). The distance between two ports on the grid is called the *port pitch*.

e) Channels: Ultimately, the ports of modules must be connected with microfluidic channels that are located in the channel layer inside the main chip board (cf. Fig. 1a). More precisely, each channel connects a predetermined pair of ports. Although the shape and dimensions of channels are *not* covered by the ISO standard (i.e., channels are not restricted to a certain cross-section or layout, e.g., rectilinear or octilinear routing), some practical constraints are commonly employed and used as a baseline for this work: Channels have a certain width and required minimum spacing (illustrated in Fig. 1b), i.e., the area around each channel where no other channels may be routed and the minimum distance to the chip boundaries is restricted. The path of a channel has a octilinear shape, i.e., only horizontal,vertical, and diagonal segments are allowed. Each channel consists of multiple waypoints,

i.e., intermediate points of channel segments (cf. Fig. 1b). Furthermore, channels on the same layer cannot cross each other. Finally, each channel may have an upper bound for its length.

f) Routing Exclusion Zones: For certain applications, it is necessary that specific areas of the main chip remain free from channels (*routing exclusion zones*). For example, when conducting optical experiments on a module, the area on the main chip behind the module may be cut out for better visibility and, therefore, cannot contain any channels. Such routing exclusion zones may be defined both globally on the main chip as well as relative to a module (*module-bound exlusion zone*), i.e., the exclusion zone moves and rotates along with the module.

It should be noted that ISO 22916 does not consider additional constraints that occur in some microfluidic applications. For example, it makes no distinction between flow and control channels [28] and does not consider their interaction, as such a distinction does not necessarily exist in some applications that employ this standard [2]. Furthermore, it does not consider timing constraints [39].

Following this standard and the resulting practical constraints, the objective of the designer is now to fit all modules as well as all channels onto the chip while, at the same time, abiding by all of the specifications and restrictions reviewed above. This is typically done in an iterative fashion. For example, the designer first places the modules and, afterwards, tries to connect them with the corresponding channels. While doing that, they frequently work with partial designs, need to move components around, and try to "wiggle" everything together until a complete design results. While that alone is a tedious task, constantly checking for compliance with the ISO standard adds an additional burden. Tool support that automatically covers these checks, i.e., validates whether the current (partial) design still is ISO-compliant (and maybe even assists in completing partial designs, e.g., by automatically realizing missing channel connections), would constitute a substantial improvement of this process. In this work, we propose an approach that provides such tool support.

B. General Idea

The main idea behind the proposed approach is based on solvers for *Satisfiability Modulo Theories* (SMT, [37]). More precisely, all entities, specifications, and constraints introduced in the previous section are encoded as SMT constraints. To that end, geometrical quantities such as coordinates, dimensions, and distances are represented as integers on a µm scale, whereas variants of entities such as module orientation or channel direction are encoded as enums, i.e., bitvectors. Consequently, the aforementioned constraints result in a large equation system, factoring in module placements, channel routings, etc.

The most important advantage of using an SMT solver is that the encoded variables can be assigned to an arbitrary degree. This can be used to provide tool support for the following scenarios:

1) *Design validation (complete assignment)*: All modules are already placed and the channels are explicitly routed with fixed waypoints, i.e., all variables are assigned a definitive value. Then, the solver determines whether the imposed design introduces any violations of the design constraints. In other words, it validates that indeed all design specifications are satisfied and the resulting chip is compliant to the ISO standard.
2) *Design completion (partial assignment)*: Only a subset of variables is assigned, while others are left unassigned (e.g., channel waypoints, and thus the channels' paths are not predetermined). Then, the solver attempts to find a model that satisfies all constraints. If this succeeds, it effectively completes the design according to the partial specification while, at the same time, ensures that the resulting solution satisfies all specifications and is compliant to the ISO standard.

In both cases, if such a design exists, one ends up with a fully specified microfluidic chip design, with the certainty that indeed all relevant placement, routing, and ISO constraints are satisfied. If not, it is shown that a violation exists. Overall, this provide great assistance during the design process.

III. SMT Encoding

Obviously, the core of the proposed idea is the encoding of all ISO- and design-constraints in terms of an SMT formulation. In this section, we introduce the necessary constraints and computations for the main chip board, the modules, and the channels. This eventually results in an implementation that can be used for validation and design completion as proposed above.

A. Chip Board Constraints

The basic dimensions, i.e., the width b_w and the height b_h (depicted in Fig. 2), of the main chip board are imposed by ISO 22916, leading to a series of constraints in the form of

$$\bigvee b_w = v_1 \wedge b_h = v_2, \tag{1}$$

where v_1 and v_2 denote admissible board width and board height values allowed defined the standard [33].

B. Module Constraints

On the chip board, there are modules m^i with $0 \leq i < m_n$ where m_n is the total number of modules. Their characterization and the resulting constraints are explored in the following, all of which are also illustrated in Fig. 2.

a) Position: Every module m^i's position is defined by its lower-left corner point (m_x^i, m_y^i) (denoted by the purple point in Fig. 2).

b) Orientation: Every module m^i has a certain orientation m_o^i, i.e., either *up* (m_o^i = U, the default), *right* (R), *down* (D), or *left* (L). As an example, m_o^j = U and m_o^j = R orientations are depicted for m^j in red and blue in Fig. 2.

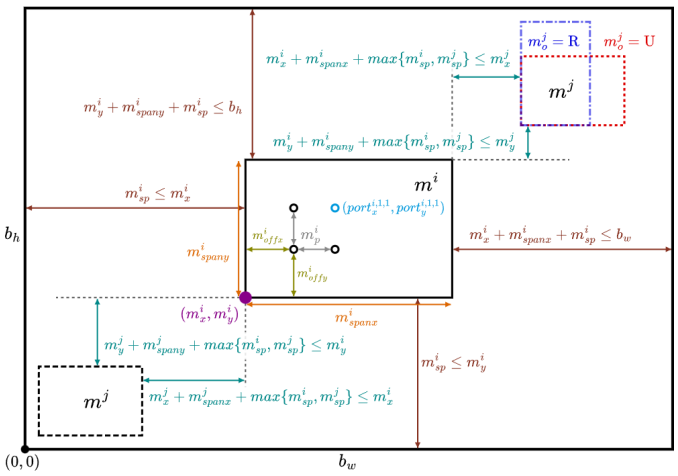


Fig. 2: Module constraints

c) Dimensions: A module has a fixed width m_w^i and height m_h^i, both of which must be a multiple of 15 mm, i.e.,

$$\exists n > 0, m > 0 : m_w^i = 15000n \wedge m_h^i = 15000m. \quad (2)$$

Without changing orientation, this defined the module's span (m_{spanx}^i and m_{spany}^i, as indicated in orange for m^i in Fig. 2) on the board. However, if the module is rotated to the side, its width and height swap (as previously illustrated by the orientation example)[2], i.e.,

$$m_{spanx}^i = \begin{cases} m_w^i, & \text{if } m_o^i = \mathrm{U} \mid \mathrm{D} \\ m_h^i, & \text{if } m_o^i = \mathrm{R} \mid \mathrm{L} \end{cases}$$
$$m_{spany}^i = \begin{cases} m_h^i, & \text{if } m_o^i = \mathrm{U} \mid \mathrm{D} \\ m_w^i, & \text{if } m_o^i = \mathrm{R} \mid \mathrm{L} \end{cases} . \quad (3)$$

d) Top and Bottom Modules: A module m^i may be placed either on the top ($m_s^i = \mathrm{T}$) or on the bottom ($m_s^i = \mathrm{B}$) *side* of the main chip.

e) Spacing Constraints Towards Boundaries: Modules must keep a certain minimum distance to the board edges. Since constraints of this type are needed throughout the rest of this section, we define a *box-contains-box* predicate *bcb* (illustrated in Fig. 3) that yields true if a rectangular box b is fully contained in another rectangular box a with an additional margin of at least d:

$$\begin{aligned} bcb\,(a_x, a_y, a_w, a_h,\ b_x, b_y, b_w, b_h,\ d) := \\ a_x + d \leq b_x \wedge b_x + b_w + d \leq a_x + a_w \wedge \\ a_y + d \leq b_y \wedge b_y + b_h + d \leq a_y + a_h, \end{aligned} \quad (4)$$

where (a_x, a_y) and (b_x, b_y) are the lower-left corner points, a_w and b_w are the widths, and a_h and b_h are the heights of the boxes a and b, respectively.

Subsequently, every module m^i, including its spacing m_{sp}^i, must be fully contained inside the main chip, i.e., the following condition must hold:

$$bcb\left(0, 0, b_w, b_h, m_x^i, m_y^i, m_{spanx}^i, m_{spany}^i, m_{sp}^i\right) \quad (5)$$

[2]As a shorthand notation, we write $x = \mathrm{U} \mid \mathrm{D}$ for $x = \mathrm{U} \vee x = \mathrm{D}$.

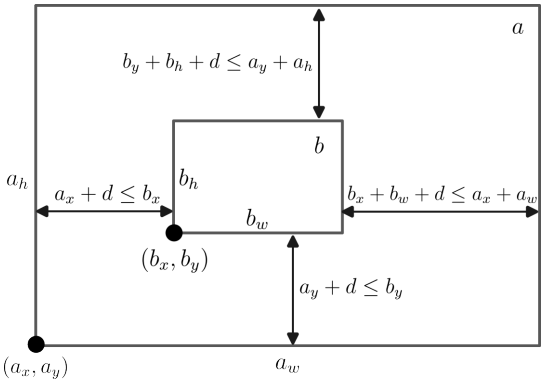


Fig. 3: Box-contains-box (*bcb*, a contains b)

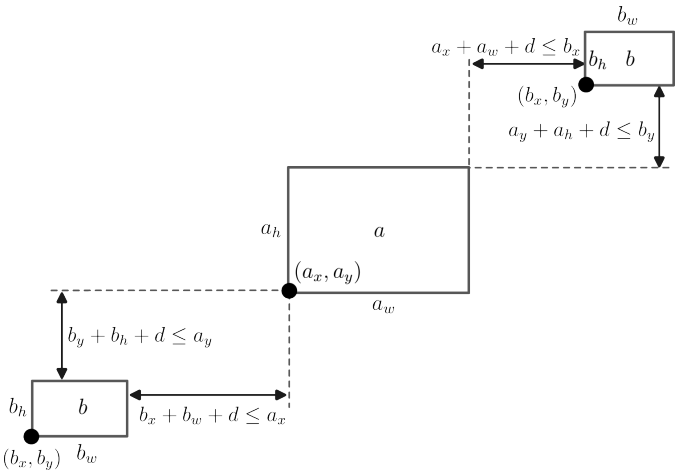


Fig. 4: Bounding-box-separation (*bbs*, between a and b)

In other words, *all* of the depicted boundary constraints (in brown) in Fig. 2 must be satisfied.

f) Spacing Constraints Towards Other Modules: Similarly, modules must keep a certain minimum distance to other modules if they are to be placed on the same side of the main chip. Analogously, we define a *bounding-box-separation* predicate *bbs* (illustrated in Fig. 4) that yields true if two rectangular boxes a and b have a minimum axis-aligned separation of d:

$$\begin{aligned} bbs\,(a_x, a_y, a_w, a_h,\ b_x, b_y, b_w, b_h,\ d) := \\ a_x + a_w + d \leq b_x \vee b_x + b_w + d \leq a_x \vee \\ a_y + a_h + d \leq b_y \vee b_y + b_h + d \leq a_y, \end{aligned} \quad (6)$$

where, again, (a_x, a_y) and (b_x, b_y) are the lower-left corner points, a_w and b_w are the widths, and a_h and b_h are the heights of the boxes a and b, respectively.

Given this predicate, no pair (m^i, m^j) of same-side modules may violate the mutual spacing constraints:

$$\begin{aligned} &i \neq j \vee m_s^i \neq m_s^j \vee \\ &bbs\left(m_x^i, m_y^i, m_{spanx}^i, m_{spany}^i, m_x^j, m_y^j, m_{spanx}^j, m_{spany}^j, \max\{m_{sp}^i, m_{sp}^j\}\right). \end{aligned} \quad (7)$$

As a visual example, *at least one* of the mutual separation constraints (in teal) in Fig. 2 must be satisfied.

g) Port Pitch: As introduced in Sec. II, each module has a set of ports distributed on a uniform grid. The distance between ports, the port pitch m_p^i (illustrated in gray in Fig. 2), must be a multiple of 1.5 mm, i.e.,

$$\exists n > 0 : m_p^i = 1500n. \tag{8}$$

h) Port Position: In order to define the positions of the ports, the total number of ports in both directions (m_{px}^i and m_{py}^i) with respect to the grid can be precomputed in advance by

$$m_{px}^i = \left\lfloor \frac{m_w^i}{m_p^i} \right\rfloor - 1, \quad m_{py}^i = \left\lfloor \frac{m_h^i}{m_p^i} \right\rfloor - 1. \tag{9}$$

Then, the offsets in both directions (m_{offx}^i and m_{offy}^i as depicted in olive in Fig. 2) between the module edges and the first port are also precomputed by

$$m_{offx}^i = \left\lfloor \frac{m_w^i - m_p^i(m_{px}^i - 1)}{2} \right\rfloor, \quad m_{offy}^i = \left\lfloor \frac{m_h^i - m_p^i(m_{py}^i - 1)}{2} \right\rfloor. \tag{10}$$

Finally, the port position coordinates $(port_x^{ijk}, port_y^{ijk})$ depend on the position and orientation of the module, where j, k with $0 \leq j < m_{px}^i$ and $0 \leq k < m_{py}^i$ are the indices of the port on the grid (as an example, $port^{i,1,1}$ is depicted in cyan in Fig. 2), i.e.,

$$port_x^{ijk} = \begin{cases} m_x^i + m_{offx}^i + j \cdot m_p^i, & \text{if } m_o^i = \text{U} \\ m_x^i + m_{offy}^i + k \cdot m_p^i, & \text{if } m_o^i = \text{R} \\ m_x^i + m_w^i - m_{offx}^i - j \cdot m_p^i, & \text{if } m_o^i = \text{D} \\ m_x^i + m_h^i - m_{offy}^i - k \cdot m_p^i, & \text{if } m_o^i = \text{L} \end{cases}$$
$$port_y^{ijk} = \begin{cases} m_y^i + m_{offy}^i + k \cdot m_p^i, & \text{if } m_o^i = \text{U} \\ m_y^i + m_w^i - m_{offx}^i - j \cdot m_p^i, & \text{if } m_o^i = \text{R} \\ m_y^i + m_h^i - m_{offy}^i - k \cdot m_p^i, & \text{if } m_o^i = \text{D} \\ m_y^i + m_{offx}^i + j \cdot m_p^i, & \text{if } m_o^i = \text{L} \end{cases}. \tag{11}$$

C. Channel Constraints

In this section, the characterization of channels and the resulting constraints are explained, and also illustrated in Fig. 5. Each channel c^i consists of a sequence of segments s^{ij} with $0 \leq j < s_n^i$ (where s_n^i is the total number of segments of c^i) and waypoints w^{ik} with $0 \leq k \leq s_n^i$. More precisely, each waypoint consists of its two coordinates, i.e., $w^{ik} = (w_x^{ik}, w_y^{ik})$. Each two consecutive waypoints denote the endpoints of a segment, e.g., in Fig. 5 waypoints $w^{i,1}$ and $w^{i,2}$ (in red) are the endpoints of $s^{i,1}$ (in blue).

a) Inactive Segments: In many cases it is not desirable that a channel uses all of its s_n^i segments, as it could be shorter with fewer waypoints. The following mechanism ensures that each channel may be composed of any number of segments between 0 and s_n^i: Each segment can be *active* ($s_a^{ik} = true$) or *inactive* ($s_a^{ik} = false$). If a segment is inactive, both of its endpoints collapse into one, i.e.,

$$\neg s_a^{ik} \Leftrightarrow w_x^{i,k} = w_x^{i,k+1} \wedge w_y^{i,k} = w_y^{i,k+1}. \tag{12}$$

For example, in Fig. 5, $w^{i,3}$ and $w^{i,4}$ (in orange) collapse into the same point since $s^{i,3}$ (in teal) is inactive. Finally, inactive segments should only occur consecutively towards the end of the channel, i.e.,

$$\forall k > 0 : \neg s_a^{i,k-1} \Rightarrow \neg s_a^{i,k}. \tag{13}$$

b) Segment Types: Additionally, each channel segment has a type s_t^{ij} defining its orientation. Not only are they characterized into horizontal, vertical, and diagonal segments, but also by whether they run in positive or negative direction, resulting in eight different types: *right* (R, positive x direction), *up* (U, positive y direction), *left* (L, negative x direction), *down* (D, negative y direction), *up-right* (UR, positive x and y directions), *down-right* (DR, positive x, negative y direction), *down-left* (DL, negative x and y directions), and *up-left* (UL, negative x, positive y direction). As an example, Fig. 5 contains right-oriented segment $s^{i,1}$ (in blue) and upward segment $s^{i,2}$ (in brown). This distinction is in effect only for active segments[3], i.e.,

$$\begin{aligned} s_a^{ik} \Rightarrow &\left(s_t^{ik} = \text{U} \Leftrightarrow w_x^{i,k} = w_x^{i,k+1} \wedge w_y^{i,k} < w_y^{i,k+1}\right) \wedge \\ &\left(s_t^{ik} = \text{R} \Leftrightarrow w_x^{i,k} < w_x^{i,k+1} \wedge w_y^{i,k} = w_y^{i,k+1}\right) \wedge \\ &\left(s_t^{ik} = \text{D} \Leftrightarrow w_x^{i,k} = w_x^{i,k+1} \wedge w_y^{i,k} > w_y^{i,k+1}\right) \wedge \\ &\vdots \\ &\left(s_t^{ik} = \text{UL} \Leftrightarrow w_x^{i,k} > w_x^{i,k+1} \wedge w_y^{i,k} < w_y^{i,k+1}\right). \end{aligned} \tag{14}$$

Additionally, for diagonal segments, the differences in x and y directions must match[2], i.e.,

$$\begin{aligned} &\left(s_t^{ik} = \text{UR} \mid \text{DR} \mid \text{DL} \mid \text{UL}\right) \Rightarrow \\ &\qquad |w_x^{ik} - w_x^{i,k+1}| = |w_y^{ik} - w_y^{i,k+1}|. \end{aligned} \tag{15}$$

Finally, consecutive segments of the same channel should not form sharp, 45° corners (e.g., a R segment followed by a DL segment is not allowed), i.e.,

$$\forall k > 0 : s_a^{ik} \Rightarrow s_t^{ik} = \begin{cases} \text{R} \mid \text{L} \mid \text{UR} \mid \text{UL}, & \text{if } s_t^{i,k-1} = \text{U} \\ \text{D} \mid \text{U} \mid \text{DR} \mid \text{UR}, & \text{if } s_t^{i,k-1} = \text{R} \\ \text{L} \mid \text{R} \mid \text{DL} \mid \text{DR}, & \text{if } s_t^{i,k-1} = \text{D} \\ \text{U} \mid \text{D} \mid \text{UL} \mid \text{DL}, & \text{if } s_t^{i,k-1} = \text{L} \\ \text{DR} \mid \text{UL} \mid \text{R} \mid \text{U}, & \text{if } s_t^{i,k-1} = \text{UR} \\ \text{DL} \mid \text{UR} \mid \text{D} \mid \text{R}, & \text{if } s_t^{i,k-1} = \text{DR} \\ \text{UL} \mid \text{DR} \mid \text{L} \mid \text{D}, & \text{if } s_t^{i,k-1} = \text{DL} \\ \text{UR} \mid \text{DL} \mid \text{U} \mid \text{L}, & \text{if } s_t^{i,k-1} = \text{UL} \end{cases}. \tag{16}$$

c) Soft Corners: While the previous definition allows both 90° and 135° corners at channel waypoints, it may be desirable to allow 135° corners only for certain applications (e.g., due to improved flow properties). To this end, a binary *soft corners flag* sc is introduced that is set to true if only 135°

[3]Some analogous terms are omitted due to lack of space.

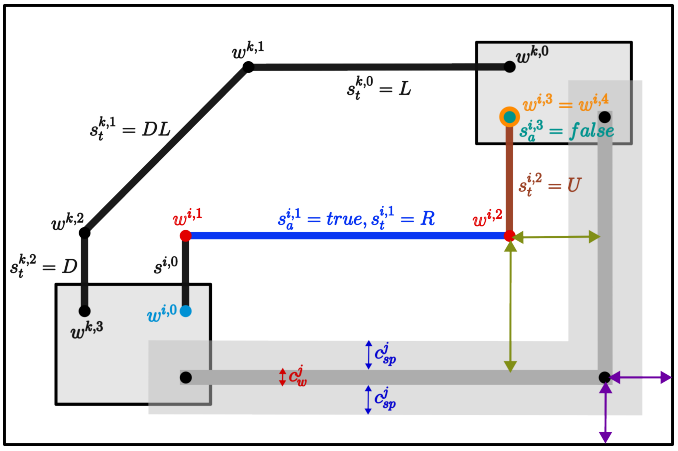


Fig. 5: Channel constraints

corners are allowed. Accordingly, Eq. 16 is further restricted to

$$\forall k > 0 : sc \wedge s_a^{ik} \Rightarrow s_t^{ik} = \begin{cases} \text{UR} \mid \text{UL}, & \text{if } s_t^{i,k-1} = \text{U} \\ \text{DR} \mid \text{UR}, & \text{if } s_t^{i,k-1} = \text{R} \\ \text{DL} \mid \text{DR}, & \text{if } s_t^{i,k-1} = \text{D} \\ \text{UL} \mid \text{DL}, & \text{if } s_t^{i,k-1} = \text{L} \\ \text{R} \mid \text{U}, & \text{if } s_t^{i,k-1} = \text{UR} \\ \text{D} \mid \text{R}, & \text{if } s_t^{i,k-1} = \text{DR} \\ \text{L} \mid \text{D}, & \text{if } s_t^{i,k-1} = \text{DL} \\ \text{U} \mid \text{L}, & \text{if } s_t^{i,k-1} = \text{UL} \end{cases}. \tag{17}$$

d) Waypoint Distance to Boundaries: Each waypoint must be sufficiently far away from the edges of the chip in order to guarantee that the channel's spacing constraint is not violated (illustrated in Fig. 5 by the purple arrows), i.e.,

$$\begin{aligned} \frac{c_w^i}{2} + c_{sp}^i \le w_x^{ik}, \ \frac{c_w^i}{2} + c_{sp}^i \le w_y^{ik}, \\ w_x^{ik} + \frac{c_w^i}{2} + c_{sp}^i \le b_w, \ w_y^{ik} + \frac{c_w^i}{2} + c_{sp}^i \le b_h, \end{aligned} \tag{18}$$

where c_w^i is the channel width and c_{sp}^i is the channel spacing (as depicted in Fig. 5 as c_w^j in red and c_{sp}^j in blue).

e) Segment Crossing: Any two segments of arbitrary channels must never cross each other. To this end, we split the problem in different cases. First, we define an *axis-separation* predicate as that yields true only if the interiors of a horizontal segment s^{ik} ($s_t^{ik} = \text{R} \mid \text{L}$) and a vertical segment s^{jl} ($s_t^{jl} = \text{U} \mid \text{D}$) do not intersect, i.e.,

$$\begin{aligned} &as\left(s^{ik}, s^{jl}\right) := \\ &\quad s_{x2}^{ik} \le s_x^{jl} \vee s_{x1}^{ik} \ge s_x^{jl} \vee s_{y2}^{jl} \le s_y^{ik} \vee s_{y1}^{jl} \ge s_y^{ik} \end{aligned} \tag{19}$$

with the following notation for segment end points:

$$\begin{aligned} s_{x1}^{ik} &:= \min\{w_x^{ik}, w_x^{i,k+1}\} \\ s_{x2}^{ik} &:= \max\{w_x^{ik}, w_x^{i,k+1}\} \\ s_{y1}^{ik} &:= \min\{w_y^{ik}, w_y^{i,k+1}\} \\ s_{y2}^{ik} &:= \max\{w_y^{ik}, w_y^{i,k+1}\} \\ s_x^{ik} &:= s_{x1}^{ik} \Leftrightarrow s_{x1}^{ik} = s_{x2}^{ik} \\ s_y^{ik} &:= s_{y1}^{ik} \Leftrightarrow s_{y1}^{ik} = s_{y2}^{ik} \end{aligned}. \tag{20}$$

.

Next, we transform diagonal segments into horizontal/vertical segments with the following transformation that rotates a segment s^{ik} by 45° in counter-clockwise direction by transforming the segment's end points[4]. The transformed segment is denoted by $\overline{s^{ik}}$, with transformed end points $\overline{w^{ik}}$ and $\overline{w^{i,k+1}}$, i.e.,

$$\begin{aligned} t\left(x, y\right) &:= \left(x - y, x + y\right) \\ \left(\overline{w_x^{ik}}, \overline{w_y^{ik}}\right) &:= t\left(w_x^{ik}, w_y^{ik}\right) \\ \left(\overline{w_x^{i,k+1}}, \overline{w_y^{i,k+1}}\right) &:= t\left(w_x^{i,k+1}, w_y^{i,k+1}\right). \end{aligned} \tag{21}$$

As a summary, the *no-crossing-of-orthogonal-segments* predicate $\neg cr_\perp$ yields true if the interiors of two orthogonal segments s^{ik}, s^{jl} do not intersect (with symmetric cases ignored), i.e.,

$$\neg cr_\perp\left(s^{ik}, s^{jl}\right) := \begin{cases} as(s^{ik}, s^{jl}), & \text{if } s_t^{ik} = \text{R} \mid \text{L} \\ as(\overline{s^{ik}}, \overline{s^{jl}}), & \text{if } s_t^{ik} = \text{UR} \mid \text{DL} \\ true, & \text{otherwise} \end{cases}. \tag{22}$$

The remaining case, non-orthogonal segment pairs (i.e., one segment is axis-aligned while the other one is diagonal), can be dealt with rather easily:

$$\neg cr_\not\perp\left(s^{ik}, s^{jl}\right) := \begin{cases} t1(s^{ik}, s^{jl}), & \text{if } s_t^{ik} = \text{R} \mid \text{L} \\ t2(s^{ik}, s^{jl}), & \text{if } s_t^{ik} = \text{U} \mid \text{D} \\ true, & \text{otherwise} \end{cases}. \tag{23}$$

with

$$\begin{aligned} t1\left(s^{ik}, s^{jl}\right) &:= s_y^{ik} \le s_{y1}^{jl} \vee s_y^{ik} \ge s_{y2}^{jl} \vee x' \le s_{x1}^{ik} \vee x' \ge s_{x2}^{ik} \\ x' &:= s_{x1}^{jl} + \begin{cases} s_y^{ik} - s_{y1}^{jl}, & \text{if } s_t^{jl} = \text{UR} \mid \text{DL} \\ s_{y2}^{jl} - s_y^{ik}, & \text{if } s_t^{jl} = \text{DR} \mid \text{UL} \end{cases}. \end{aligned} \tag{24}$$

$$\neg cr\left(s^{ik}, s^{jl}\right) := \begin{cases} \neg cr_\perp, \text{if } s^{ik} \perp s^{jl} \\ \neg cr_\not\perp, \text{if } s^{ik} \not\perp s^{jl} \end{cases} \tag{25}$$

By employing this predicate to all pairs of segments that actually *can* collide, i.e, active segments in non-parallel position, it is ensured that no segment crossings occur, i.e.,

$$s_a^{ik} \wedge s_a^{jl} \wedge \left(s^{ik} \nparallel s^{jl}\right) \Rightarrow \neg cr\left(s^{ik}, s^{jl}\right). \tag{26}$$

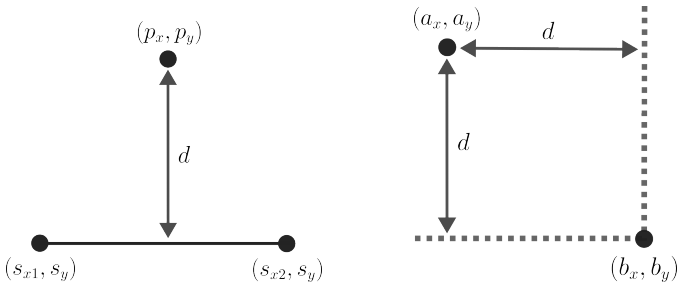
f) Waypoint Spacing Constraints Towards Segments: Waypoints must have a certain distance to any channel segment in order to ensure that channel spacing constraints are not violated. For the case of axis-aligned (horizontal or vertical) segments, we define the *point-segment distance* predicate psd that yields true only if a point p has a minimum axis-aligned distance d from a segment s, i.e.,

$$\begin{aligned} &psd(p_x, p_y, s_{x1}, s_{x2}, s_y, d) := \\ &\quad p_x \le s_{x1} \vee s_{x2} \le p_x \vee p_y + d \le s_y \vee s_y + d \le p_y \end{aligned} \tag{27}$$

where s is a horizontal segment with $s_{x1} < s_{x2}$ (a graphic example is illustrated in Fig. 6a)[5]. Again, this predicate is

[4]The transformation does not retain the segment length, however, this is irrelevant when testing for intersections.

[5]Note that the axes can be swapped here.



(a) *psd* predicate (b) *ppd* predicate

Fig. 6: Point predicates

applied to all pairings of active axis-aligned segments and waypoints, and the minimum required distance d_{ij} (illustrated by the olive arrows in Fig. 5) is computed from both involved channels' width c^i_w, c^j_w and spacing c^i_{sp}, c^j_{sp}, i.e.,

$$\begin{aligned} s^{ik}_a \Rightarrow &\left(s^{ik}_t = \mathrm{U} \Rightarrow psd\left(w^{jl}_y, w^{jl}_x, w^{ik}_y, w^{i,k+1}_y, w^{ik}_x, d_{ij}\right)\right) \wedge \\ &\left(s^{ik}_t = \mathrm{D} \Rightarrow psd\left(w^{jl}_y, w^{jl}_x, w^{i,k+1}_y, w^{ik}_y, w^{ik}_x, d_{ij}\right)\right) \wedge \\ &\left(s^{ik}_t = \mathrm{R} \Rightarrow psd\left(w^{jl}_x, w^{jl}_y, w^{ik}_x, w^{i,k+1}_x, w^{ik}_y, d_{ij}\right)\right) \wedge \\ &\left(s^{ik}_t = \mathrm{L} \Rightarrow psd\left(w^{jl}_x, w^{jl}_y, w^{i,k+1}_x, w^{ik}_x, w^{ik}_y, d_{ij}\right)\right) \end{aligned} \tag{28}$$

with

$$d_{ij} := \left\lceil \frac{c^i_w + c^j_w}{2} \right\rceil + \max\{c^i_{sp}, c^j_{sp}\}. \tag{29}$$

The constraint for diagonal segments, again, is applied to all pairings of active segments and waypoints. However, the precise formulation is too tedious and lengthy to be presented in this work. Instead, we refer to the implementation that is provided at https://github.com/cda-tum/mmft-iso-designer.

g) Waypoint Spacing Constraints Towards Waypoints: Similarly, waypoints must have a certain minimum distance to other waypoints. The *point-point distance* predicate *ppd* (illustrated in Fig. 6b) guarantees that points a and b have a minimum axis-aligned distance of d, i.e.,

$$\begin{aligned} &ppd(a_x, a_y, b_x, b_y, d) := \\ &a_x + d \leq b_x \vee a_y + d \leq b_y \vee b_x + d \leq a_x \vee b_y + d \leq a_y. \end{aligned} \tag{30}$$

Once again, is applied to all pairs of waypoints. However, for pairs belonging to the same channel, one must exclude cases where waypoints collapse due to inactive segments, i.e., where a waypoint is the second endpoint of an inactive segment. This results in

$$ppd(w^{ik}_x, w^{ik}_y, w^{jl}_x, w^{jl}_y, d_{ij}) \quad \text{if } i \neq j \tag{31}$$

$$s^{i,k-1}_a \Rightarrow ppd(w^{ik}_x, w^{ik}_y, w^{il}_x, w^{il}_y, d_{ij}) \quad \text{if } i = j \wedge k > l \tag{32}$$

with d_{ij} identical to the previously defined Eq. 29.

h) Port Connections: The first waypoint $w^{i,0}$ and last waypoint w^{i,s^i_n} of a channel are each assigned to their corresponding port position (cf. Eq. 11), i.e.,

$$\begin{aligned} w^{i,0}_x &= port^{a_m,a_x,a_y}_x, \quad w^{i,0}_y = port^{a_m,a_x,a_y}_y, \\ w^{i,s^i_n}_x &= port^{b_m,b_x,b_y}_x, \quad w^{i,s^i_n}_y = port^{b_m,b_x,b_y}_y \end{aligned} \tag{33}$$

where a_m and b_m denote the indices of the start and end modules of this channel, respectively. Correspondingly, a_x, a_y, b_x and b_y denote indices of the start and end ports on each module. As an example, endpoints $w^{i,0}$ (in cyan) and $w^{i,4}$ (in orange) are illustrated in Fig. 5.

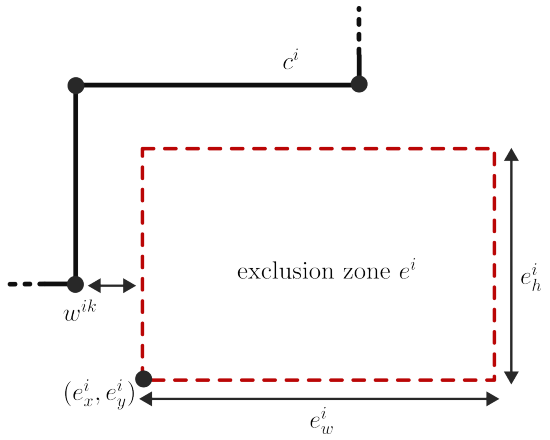


Fig. 7: Routing exclusion zone

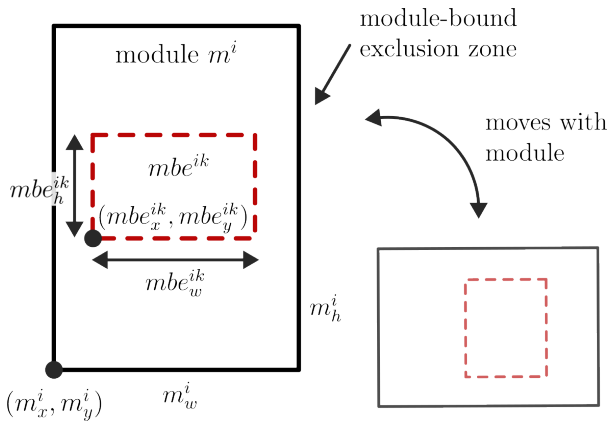


Fig. 8: Module-bound routing exclusion zone

i) Upper Bound for Channel Length: Finally, the channel length, measured in Manhattan distance, i.e., the sum of all of its segments' Manhattan distances, must satisfy an upper bound c^i_l, i.e.,

$$\sum_{j>0} \left(|w^{ij}_x - w^{i,j-1}_x| + |w^{ij}_y - w^{i,j-1}_y|\right) \leq c^i_l. \tag{34}$$

D. Routing Exclusion Zone Constraints

Routing exclusion zones are rectangular areas on the chip board where no channels are allowed to be routed through. We denote exclusion zones by e^i with $0 \leq i < e_n$ where e_n is the total number of exclusion zones. Their implications and the resulting constraints are explored in the following and illustrated in Fig. 7 and Fig. 8.

a) Position and Dimensions: Every exclusion zone e^i's position is defined by its lower-left corner point (e^i_x, e^i_y) (illustrated in Fig. 7). An exclusion zone has a fixed width e^i_w and height e^i_h.

b) Spacing Towards Waypoints of Channels: Since routing exclusion zones are supposed to prevent channels from entering the area, it must be ensured that proper spacing between channels and routing exclusion zones is adhered

to. To this end, a waypoint w^{ik} or channel c_i must have a minimum distance d_i to a routing exclusion zone e^i, i.e.,

$$w_x^{ik} + d_i \leq e_x^i \vee w_y^{ik} + d_i \leq e_y^i \vee w_x^{ik} \geq e_x^i + e_w^i + d_i \vee w_y^{ik} \geq e_y^i + e_h^i + d_i \quad (35)$$

with

$$d_i := \left\lceil \frac{c_w^i}{2} \right\rceil + c_{sp}^i.$$

c) Minimal Spacing to Channel Segments: Consequently, it must be ensured that also channel segments do not intersect or run too close to routing exclusion zones. Such a constraint can be formulated analogously to the no-cross constraint of channel segments with each other (cf. Eq. 26 and preceding), with the additional consideration that an exclusion zone extends into two dimensions (width and height) with the respective spacing, instead of a one-dimensional segment with the respective spacing. To keep this work concise, we omit the detailed formulation of this constraint. However, we refer to the implementation that is provided at https://github.com/cda-tum/mmft-iso-designer.

d) Module-bound Exclusion Zones: Additionally, each module m^i may have multiple module-bound exclusion zones mbe^{ik} (illustrated in Fig. 8) with $0 \leq k < mbe_n^i$ where mbe_n^i is the number of module-bound exclusion zones for module m^i, which prohibit channels in a specified area relative to the module, i.e., it moves and rotates in the same way. Similarly to regular exlusion zones, each module-bound exclusion zone is defined by its width mbe_w^{ik} and height mbe_h^{ik}. The position is given by the lower-left corner point (mbe_x^{ik}, mbe_y^{ik}), which is relative to the module's lower-left corner point (m_x^i, m_y^i) (i.e., if $(mbe_x^{ik}, mbe_y^{ik}) = (0, 0)$, it is identical to the module's position).

Each module-bound exclusion zone mbe^{ik} can be transformed into a regular exclusion zone $e^{mbe^{ik}}$ with a dynamically determined position $(e_x^{mbe^{ik}}, e_y^{mbe^{ik}})$ (an analogous mechanism as already shown in Eq. 11):

$$e_x^{mbe^{ik}} = \begin{cases} m_x^i + mbe_x^{ik}, & \text{if } m_o^i = \text{U} \\ m_x^i + mbe_y^{ik}, & \text{if } m_o^i = \text{R} \\ m_x^i + m_w^i - mbe_x^{ik} - mbe_w^{ik}, & \text{if } m_o^i = \text{D} \\ m_x^i + m_h^i - mbe_y^{ik} - mbe_h^{ik}, & \text{if } m_o^i = \text{L} \end{cases}.$$

$$e_y^{mbe^{ik}} = \begin{cases} m_y^i + mbe_y^{ik}, & \text{if } m_o^i = \text{U} \\ m_y^i + m_w^i - mbe_x^{ik} - mbe_w^{ik}, & \text{if } m_o^i = \text{R} \\ m_y^i + m_h^i - mbe_y^{ik} - mbe_h^{ik}, & \text{if } m_o^i = \text{D} \\ m_y^i + mbe_x^{ik}, & \text{if } m_o^i = \text{L} \end{cases} \quad (36)$$

Additionally, the dimensions $e_w^{mbe^{ik}}$ and $e_h^{mbe^{ik}}$ of the module-bound exclusion zone change depending on the orientation m_o^i of the module.

$$e_w^{mbe^{ik}} = \begin{cases} mbe_w^{ik}, & \text{if } m_o^i = \text{U} \mid \text{D} \\ mbe_h^{ik}, & \text{if } m_o^i = \text{R} \mid \text{L} \end{cases}.$$

$$e_h^{mbe^{ik}} = \begin{cases} mbe_h^{ik}, & \text{if } m_o^i = \text{U} \mid \text{D} \\ mbe_w^{ik}, & \text{if } m_o^i = \text{R} \mid \text{L} \end{cases} \quad (37)$$

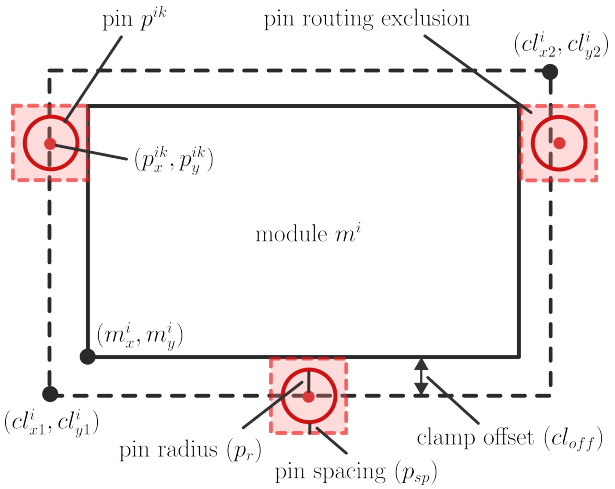


Fig. 9: Clamp and pins

Furthermore, it must be ensured that the module-bound exclusion zone stays within the main chip's boundaries. To achieve this, the *box-contains-box* predicate from Eq. 4 can be used, i.e.,

$$bcb\left(0, 0, b_w, b_h, e_x^{mbe^{ik}}, e_y^{mbe^{ik}}, e_w^{mbe^{ik}}, e_h^{mbe^{ik}}, 0\right). \quad (38)$$

Consequently, all other constraints in Sec. III-D also apply to $e^{mbe^{ik}}$.

E. Clamp and Pin Constraints

Finally, this section introduces how clamps and pins are modeled. Each clamp cl^i that encases a module m^i is attached to the board by pins p^{ik} with $0 \leq k < p_n^i$ where p_n^i is the number of pins fixing module m^i. The geometry of pins and the resulting constraints are explained in the following, and illustrated in Fig. 9.

a) Clamp Dimensions: Clamps add a certain offset cl_{off} to a module's dimensions as depicted in Fig. 9. As a result, the lower-left corner point (cl_{x1}^i, cl_{y1}^i) and the upper-right corner point (cl_{x2}^i, cl_{y2}^i) are defined by

$$\begin{aligned} cl_{x1}^i &= m_x^i - cl_{off}, \; cl_{x2}^i = m_x^i + m_{spanx}^i + cl_{off}, \\ cl_{y1}^i &= m_y^i - cl_{off}, \; cl_{y2}^i = m_y^i + m_{spany}^i + cl_{off}. \end{aligned} \quad (39)$$

b) Pin Position and Dimensions: The position of each pin is defined by its center point (illustrated in Fig. 9). All pins are assumed to have a uniform radius p_r with additional spacing p_{sp}.

c) Position on Clamp: Pins are positioned around a module as depicted in Fig. 9. More precisely, the position of each pin is bound to be located on the perimeter of the clamp, i.e.,

$$\begin{aligned} &\left(\left(p_x^{ik} = cl_{x1}^i \vee p_x^{ik} = cl_{x2}^i\right) \wedge p_y^{ik} \geq cl_{y1}^i \wedge p_y^{ik} \leq cl_{y2}^i\right) \vee \\ &\left(\left(p_y^{ik} = cl_{y1}^i \vee p_y^{ik} = cl_{y2}^i\right) \wedge p_x^{ik} \geq cl_{x1}^i \wedge p_x^{ik} \leq cl_{x2}^i\right) \end{aligned} \quad (40)$$

.

d) Routing Exclusion Zone: Since pins are fixed to the chip by drilling holes that prohibit channels at that location, each pin induces a routing exclusion zone $e^{p^{ik}}$ where

$$\begin{aligned} e_x^{p^{ik}} &= p_x^{ik} - p_r - p_{sp} \\ e_y^{p^{ik}} &= p_y^{ik} - p_r - p_{sp}. \\ e_w^{p^{ik}} = e_h^{p^{ik}} &= 2 \cdot (p_r + p_{sp}) \end{aligned} \tag{41}$$

Consequently, all constraints for exclusion zones introduced in Sec. III-D apply as well.

e) Distribution of Pins: When determining the position of pins on the perimeter of the clamp, it is important to ensure that the pins provide good fixing support for the module, i.e., they need to be spread evenly around the clamp and should not, e.g., be placed on one side only. To achieve this, we first define a perimeter separation distance d_i for module m^i:

$$d_i = \begin{cases} \max\{m_w^i, m_h^i\} & \text{if } p_n^i = 2 \\ 2p_r + \left[\frac{m_{cf}^i}{6}\right] & \text{if } p_n^i = 3 \\ p_r + \left[\frac{m_{cf}^i}{9}\right] & \text{if } p_n^i > 3 \end{cases} \tag{42}$$

where $m_{cf}^i = 2(m_w^i + m_h^i)$ is the circumference of module m^i. Then, using this parameter, the following condition has to be fulfilled for all pairs of pins p^{ik} and p^{il} on module m^i:

$$\begin{aligned} &(p_x^{ik} + d_i \leq p_x^{il} \wedge p_x^{il} + d_i \leq p_x^{ik}) \vee (p_y^{ik} + d_i \leq p_y^{il} \wedge p_y^{il} + d_i \leq p_y^{ik}) \vee \\ &\left(p_x^{ik} + \frac{d_i}{2} \leq p_x^{il} \wedge p_x^{il} + \frac{d_i}{2} \leq p_x^{ik} \wedge p_x^{ik} + \frac{d_i}{2} \leq p_x^{il} \wedge p_x^{il} + \frac{d_i}{2} \leq p_x^{ik}\right) \end{aligned} \tag{43}$$

where the first two conjunctions ensure sufficient axis separation, while the latter conjunction ensures sufficient separation around the corners.

f) Spacing to other Pins: Furthermore, pins must have appropriate spacing towards the pins of other modules. Once again, the *point-point distance* predicate *ppd* (cf. Eq. 30) can be employed to prevent intersection with other pins:

$$i \neq j \Rightarrow ppd\left(p_x^{ik}, p_y^{ik}, p_x^{jl}, p_y^{jl}, 2\left(p_r + p_{sp}\right)\right). \tag{44}$$

g) Spacing to Other Modules: In the same way, pins may not intersect other modules and must adhere to their spacing constraints. To achieve this, we can reuse the *bounding-box-separation* predicate *bbs* (cf. Eq. 6) and require that the pin (i.e., the area that is also spanned by the pin's routing exclusion zone $e^{p^{ik}}$) has at least the required spacing m_{sp}^j to each other module, i.e.,

$$\begin{aligned} &i \neq j \Rightarrow \\ &bbs\left(e_x^{p^{ik}}, e_y^{p^{ik}}, e_w^{p^{ik}}, e_h^{p^{ik}}, m_x^j, m_y^j, m_{spanx}^j, m_{spany}^j, m_{sp}^j\right). \end{aligned} \tag{45}$$

h) Spacing to Routing Exclusion Zones: Finally, pins may not overlap with other routing exclusion zones (e.g., pins could obstruct optical experiments, the main motivation for routing exclusion zones). Again, this is achieved by employing the *bounding-box-separation* predicate (cf. Eq.6):

$$i \neq l \Rightarrow bbs\left(e_x^{p^{ik}}, e_y^{p^{ik}}, e_w^{p^{ik}}, e_h^{p^{ik}}, e_x^l, e_y^l, e_w^l, e_h^l, 0\right). \tag{46}$$

IV. Resulting Tool

Following the general idea and the encodings proposed above, we implemented a tool that aids microfluidic designers in the validation and design completion of ISO-compliant chips. To this end, we utilized the Z3 Theorem Prover [37] as an SMT solver. Using the corresponding interfaces, almost all the constraints and equations provided above can be directly supplied to this solver. Exceptions are some variables that need to be computed in advance since their terms are non-linear (e.g., Eq. 10). In these cases, only the resulting value is supplied to the solver.

Using this implementation, we developed a tool that can be used for the two scenarios discussed in Sec. II-B, namely:

1) For validating whether a design is ISO-compliant, *all* variables introduced in Sec. III (with the exception of channel segment parameters that can be derived in a straightforward fashion, e.g., s_a^{ij} and s_t^{ij}) are assigned a precise value based on the specific design under consideration. Then, the solver will determine whether the resulting equation system is indeed satisfiable. If this is the case, the design has been proven to satisfy all ISO constraints.
2) For completing a design, one can also leave variables (representing components that have not been completed yet) unassigned. Then, the solver tries to determine an assignment for those remaining ones (of course, while keeping all constraints satisfied). If this succeeds, an ISO-compliant completion of the design can be obtained from the resulting assignments. If not, the partial or incomplete design already includes some components that render the generation of an ISO-compliant design impossible.

An open-source implementation of the resulting tool is available at https://github.com/cda-tum/mmft-iso-designer as part of the *Munich Microfluidics Toolkit* (MMFT).

V. Application

In order to evaluate (and eventually demonstrate) that the proposed method indeed provides assistance that improves the design process, we applied the tool described above to several use cases inspired by real-world settings, which confirmed the benefits of the endeavor described in the paper. In this section, we provide a summary of a representative subset of the conducted case studies. To this end, we first summarize the characteristics of the considered designs. Afterwards, we cover the application for ISO validation and design completion.

A. Considered Microfluidic Designs

In the following, we present results obtained from our case study of three chip designs that were inspired by corresponding real-world use cases [6], [35]—eventually resulting in designs including 3 to 9 modules with up to 19 pins, 10 to 25 channels, and up to 5 routing exclusion zones, as well as different complexities with respect to channel length, port positions, pins, etc.

Using these designs, validation tasks and design completion tasks have been considered. For the former, violations of

the constraints have been introduced to evaluate whether they are detected and how the designer is supported in this case. For the latter, some module positions, module orientations, and/or channels (i.e., their waypoints) have been removed to evaluate how the designer is supported in the "completion" of the resulting designs. The implementations and all of the presented evaluation cases can be found at https://github.com/cda-tum/mmft-iso-designer as part of the *Munich Microfluidics Toolkit* (MMFT).

B. Validation

For validation, three base designs of increasing complexity were considered, denoted by S (small), M (medium), and L (large). Table I summarizes the (representative subset of) results obtained when considering the validation of ISO-compliance. The first five columns provide the name and characteristics of the considered design: number of modules, channels, routing exclusion zones, and pins. Then, it is denoted whether the corresponding design has been validated as ISO-compliant or whether a violation has been detected. Finally, the last column provides the runtime (in seconds) of the validation process.

As can easily be seen, the validation of the original ISO-compliant designs can be conducted in reasonable runtime. As expected, the runtime moderately increases with the size of the designs. But for current settings and the complexities of real-word use cases, this does not constitute a limitation. That alone provides a substantial benefit since designers do not have to manually check for ISO-compliance anymore.

Moreover, in the case of ISO violations, substantial support is provided. This has been evaluated by introducing violations of the constraints into the design. More precisely, we considered

- violations of module spacing (denoted V_1),
- violations of channel spacing (denoted V_2),
- intersections of channels (denoted V_3),
- violations of channel lengths (denoted V_4),
- violations of routing exclusion zones (denoted V_5),
- spacing violations between pins (denoted V_6), and
- spacing violations of channels and modules to pins (denoted V_7).

Table I again confirms that those violations can be detected within a few seconds as well.

In addition to that, designers are also explicitly pinpointed to the corresponding violations. This is exemplarily showcased in Fig. 10 for *Design L*. In this example, there are violations of channel spacing (V_2), intersections of channels and routing exclusion zones (V_5), pins that are placed too close to each other and do not fix the module appropriately on both sides (V_6), and intersections of channels and pin-induced routing exclusions (V_7). Thus far (i.e., without the proposed tool), the designer needed to consider the entire board, including all modules, ports, channels, routing exclusions, as well as pins, and manually check for any violations. With the proposed tool, they are directly pinpointed to the violations (highlighted red in Fig. 10)—a substantial improvement of the current design process.

C. Design Completion

As summarized in Table II, 24 designs ($S1-S6$, $M1-M7$, $L1-L11$) of varying complexity were considered for design completion, where the prefixes S, M, and L correspondingly identify small, medium, and large designs. To "emulate" intermediate but incomplete designs, which may have resulted during the design phase, we removed some module positions, module orientations, channels (i.e., their waypoints), and/or routing exclusion zones. The first columns of Table II indicate the resulting cases (providing the number of total modules and the number of bottom-side placed modules. Furthermore, the number of *free* (not predetermined/given) module placements and *free* module orientations are given, i.e., parts of the design that need to be completed. In the same way, the total number of channels, channel segments, and length constraints on channels are given, where the *free* column again denotes the number of channels that have no predetermined placement/routing (i.e., the number of channels that need to be completed). Moreover, the number of custom routing exclusion zones (both module-bound and global) are given, not including the routing exclusions that are induced by other entities such as pins. Total channel length and bend count provide an idea of the optimality of the design. Finally, the last column again provides the runtime (in seconds) of the design completion process.

As an example, Fig. 11 depicts design $L6$ of Table II, consisting of 9 modules, 24 channels with a total of 135 channel segments, and 23 pins, but no custom routing exclusion zones other than those induced by each pin (illustrated as a red dashed box). This design has a total channel length of $1542.43\,\mathrm{mm}$ and 106 bends.

The results clearly show that design completion is a significantly more computationally complex task. This is no surprise; after all, the overall design task is an $\mathcal{NP}$-hard endeavor [40]. Moreover, the increase in runtime does not necessarily correlate with the perceived increase in complexity of a case (meaning the number of modules and channels). As an example, $L11$ of Table II has a much larger runtime than, e.g., $L7$, which has more channels to be placed and has more routing exclusion constraints. This is also due to the fact that SMT solvers frequently suffer from instability [41], i.e., small changes in input can have large effects on both the computed solution and the runtime.

But despite that, the proposed tool offers substantial value (at least for the design sizes currently considered in the microfluidic community). For smaller designs (such as $S1-S6$), the *entire* design can be automatically generated in less than a minute (cf. $S5$ in Table II, where no module placements, module orientations, or channels are given).

To put these results into perspective, equivalent runtime ranges of up to $\sim 3\,\mathrm{h}$ were reported in [19] with slightly more modules and channels (up to 60), employing a similar SAT-based approach. However, the ISO standard and many other geometric constraints of this work are not considered. Moreover, a grid-based encoding as presented in [19] is only feasible for a very coarse model. In contrast, the software presented in [28] reports better runtimes, but also does not

TABLE I: Validation cases

Design	Modules	Channels	Routing Exclusions	Pins	Valid	Violations V_1	V_2	V_3	V_4	V_5	V_6	V_7	Time [s]
S	3	10	0	6	✓								0.98
						✗							0.91
							✗						1.05
								✗					1.35
									✗				0.97
			1							✗			1.12
											✗		0.92
								✗	✗			✗	1.28
M	3	12	0	9	✓								3.32
						✗							2.71
							✗						3.22
								✗					2.92
									✗				3.17
			1							✗			2.75
											✗		2.99
			1			✗				✗		✗	2.85
L	9	25	2	19	✓								63.57
						✗							56.46
							✗						75.26
								✗					59.39
									✗				60.97
										✗			64.69
											✗		52.84
							✗			✗	✗	✗	61.30

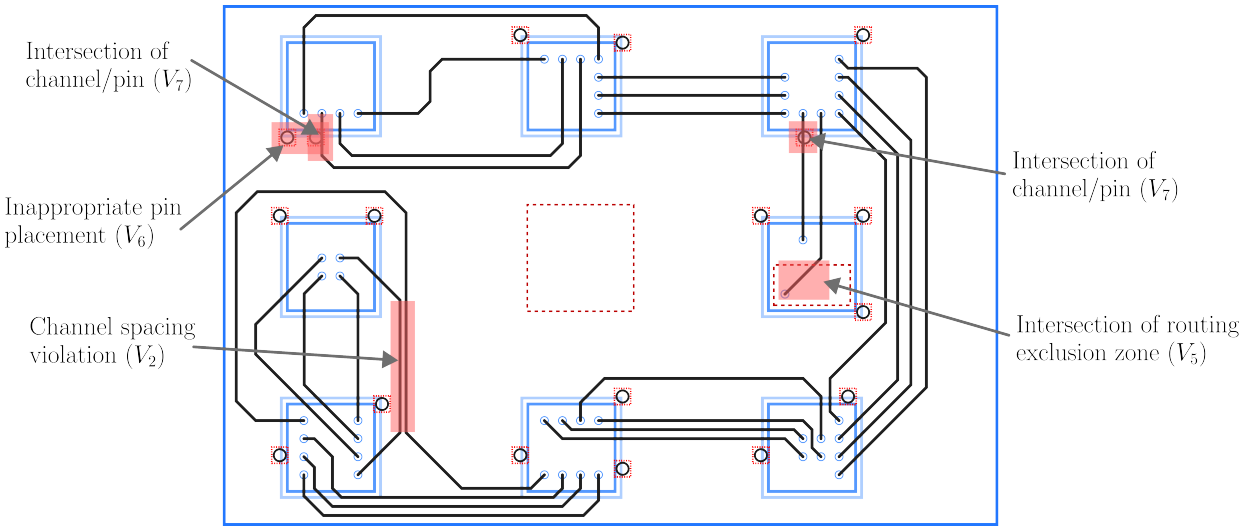


Fig. 10: Constraint violations for chip L

consider ISO-specific constraints.

For larger designs, this might not be feasible at some point. But here, too, the designer still gets substantial assistance. More precisely, following the currently established iterative design process as reviewed in Section II-B, they can make first or obvious design decisions by themselves and, then, let the tool complete the design. In particular, for the final steps in which the designers need to "wiggle" everything together, this provides important support. Of course, all this design completion assistance always guarantees compliance with the ISO standard and, if violations occur, makes the designer aware of those.

VI. Conclusion

In this work, we introduced an approach that allows the validation of microfluidic chip designs that conform to the

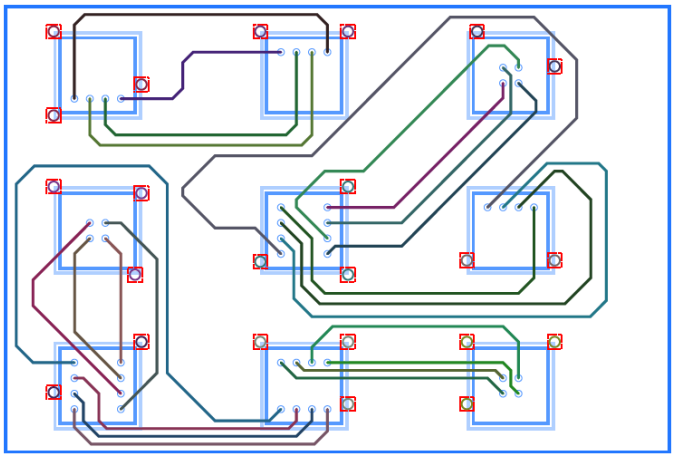


Fig. 11: The generated design for chip L6

TABLE II: Design generation cases

Design	Modules				Channels				Routing Exclusions		Total Channel Length [mm]	Bends	Time [s]
	total #	free orientation	free placement	bottom modules	total #	free	segments	length-constrained	module-bound	static			
S1	3	3	0	1	4	4	6	0	0	0	52.11	2	0.52
S2	3	3	3	1	4	4	6	0	0	0	33.21	2	0.96
S3	3	3	3	0	6	6	12	0	0	0	43.86	2	6.29
S4	3	3	3	0	6	6	12	0	0	0	109.38	6	4.89
S5	3	3	3	0	8	8	22	0	0	0	211.59	13	29.27
S6	3	3	3	0	8	8	22	0	0	0	189.58	14	96.36
M1	3	3	3	0	8	5	24	0	0	0	204.24	7	7.93
M2	3	3	3	0	6	6	16	0	0	0	201.09	9	32.31
M3	3	0	0	0	12	12	34	0	0	0	381.40	21	105.71
M4	3	3	3	0	9	9	24	0	0	0	218.09	15	160.27
M5	3	3	3	0	8	5	24	3	0	0	208.85	11	655.62
M6	3	3	3	0	12	12	34	0	0	0	421.12	20	808.21
M7	3	3	3	0	12	12	34	0	0	0	358.29	20	880.41
L1	3	3	0	0	4	4	12	0	3	2	225.79	6	8.86
L2	6	4	4	1	4	4	20	0	0	0	357.77	15	44.32
L3	9	0	0	0	4	4	18	0	0	0	257.89	13	50.83
L4	3	3	3	1	7	7	21	0	0	0	298.63	10	99.89
L5	3	3	0	0	8	8	24	0	3	2	403.71	18	103.99
L6	9	0	0	0	24	1	135	23	0	0	1542.43	106	332.20
L7	3	0	2	0	10	10	39	0	1	2	536.76	25	577.47
L8	3	0	3	0	10	10	33	0	0	2	608.85	19	743.23
L9	6	0	0	0	8	8	40	0	0	0	517.76	15	808.23
L10	9	6	0	1	8	8	38	0	0	0	526.23	30	902.26
L11	3	0	3	0	10	5	33	5	0	2	641.45	23	3525.73

ISO 22916 standard. In order to achieve this objective, we converted the criteria mandated by the standard and further geometric constraints such as channel routing, clamps and pins, and routing exclusion zones into SMT constraints. Subsequently, an SMT solver was employed to prove that a specific chip design effectively satisfies the given criteria. Additionally, the presented implementation can be used to complete a chip design from a partial specification. Through a case study covering microfluidic chip designs inspired by real-world use cases (such as [6], [35]), we demonstrated how the resulting tool substantially improves the design process for microfluidic chips.

ACKNOWLEDGMENTS

The authors used Proton Lumo and Quillbot for language editing. All content was reviewed and edited by the authors, who take full responsibility for the final work.

REFERENCES

[1] G. M. Whitesides, "The origins and the future of microfluidics," *Nature*, vol. 442, no. 7101, pp. 368–373, 2006.

[2] A. Paul, E. Safai, L. de Heus, A. Vollertsen, K. Weijgertse, B. de Wagenaar, H. 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*, pp. 2025–05, 2025.

[3] C. Parent, Y. Zhou, V. Aubert, D. Surdez, D. Olivier, C. Wilhelm, J.-L. Viovy *et al.*, "Simple droplet microfluidics platform for drug screening on cancer spheroids," *Lab on a Chip*, 2023.

[4] S. Fowler, W. L. K. Chen, D. B. Duignan, A. Gupta, N. Hariparsad, J. R. Kenny, W. G. Lai, J. Liras, J. A. Phillips, and J. Gan, "Microphysiological systems for adme-related applications: current status and recommendations for system development and characterization," *Lab on a Chip*, vol. 20, no. 3, pp. 446–467, 2020.

[5] Y. Ding, P. D. Howes, and A. J. deMello, "Recent advances in droplet microfluidics," *Analytical chemistry*, vol. 92, no. 1, pp. 132–149, 2019.

[6] A. Vollertsen, D. De Boer, S. Dekker, B. Wesselink, R. Haverkate, H. Rho, R. Boom, M. Skolimowski, M. Blom, R. Passier *et al.*, "Modular operation of microfluidic chips for highly parallelized cell culture and liquid dosing via a fluidic circuit board," *Microsystems & Nanoengineering*, vol. 6, no. 1, p. 107, 2020.

[7] A. Kumar, A. Parihar, U. Panda, and D. S. Parihar, "Microfluidics-based point-of-care testing (POCT) devices in dealing with waves of COVID-19 pandemic: The emerging solution," *ACS applied bio materials*, vol. 5, no. 5, pp. 2046–2068, 2022.

[8] R. Ramezankhani, R. Solhi, Y. C. Chai, M. Vosough, and C. Verfaillie, "Organoid and microfluidics-based platforms for drug screening in covid-19," *Drug discovery today*, vol. 27, no. 4, pp. 1062–1076, 2022.

[9] W. Ji, T.-Y. Ho, J. Wang, and H. Yao, "Microfluidic design for concentration gradient generation using artificial neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2544–2557, 2019.

[10] A. Grimmer, W. Haselmayr, and R. Wille, "Automated dimensioning of Networked Labs-on-Chip," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[11] 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.

[12] 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.

[13] ——, "Design Automation for Organs-on-Chip," 2024.

[14] X. Huang, T.-Y. Ho, K. Chakrabarty, and W. Guo, "Timing-driven flow-channel network construction for continuous-flow microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 6, pp. 1314–1327, 2019.

[15] M. Calaon, H. N. Hansen, G. Tosello, J. Garnaes, J. Nørregaard, and W. Li, "Microfluidic chip designs process optimization and dimensional quality control," *Microsystem Technologies*, vol. 21, pp. 561–570, 2015.

[16] 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 (ASP-DAC)*, 2018.

[17] M. Takken, M. Emmerich, and R. Wille, "An Abstract Simulator for Species Concentrations in Channel-Based Microfluidic Devices," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[18] P. Ebner, G. Fink, and R. Wille, "Channel routing for microfluidic devices: A comprehensive and accessible design tool," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 533–543, 2022.

[19] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 530–535.

[20] G. Liu, H. Huang, Z. Chen, H. Lin, H. Liu, X. Huang, and W. Guo, "Design automation for continuous-flow microfluidic biochips: A comprehensive review," *Integration*, vol. 82, pp. 48–66, 2022.

[21] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 55–68, 2016.

[22] A. Grimmer, P. Frank, P. Ebner, S. Häfner, A. Richter, and R. Wille, "Meander Designer: Automatically Generating Meander Channel Designs," *Micromachines*, vol. 9, no. 12, 2018.

[23] P. Zhang, C. Chen, F. Guo, J. Philippe, Y. Gu, Z. Tian, H. Bachman, L. Ren, S. Yang, Z. Zhong *et al.*, "Contactless, programmable acoustoflu-

idic manipulation of objects on water," *Lab on a Chip*, vol. 19, no. 20, pp. 3397–3404, 2019.
[24] O. Keszöcze, R. Wille, and R. Drechsler, "Exact Routing for Digital Microfluidic Biochips with Temporary Blockages," in *International Conference on Computer Aided Design (ICCAD)*, 2014, pp. 599–606.
[25] M. Ibrahim, K. Chakrabarty, and K. Scott, "Synthesis of cyberphysical digital-microfluidic biochips for real-time quantitative analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 5, pp. 733–746, 2016.
[26] O. Keszöcze, 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 (ASP-DAC)*, 2017.
[27] O. Keszöcze, R. Wille, K. Chakrabarty, and R. Drechsler, "A General and Exact Routing Methodology for Digital Microfluidic Biochips," in *International Conference on Computer Aided Design (ICCAD)*, 2015.
[28] T.-M. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T.-Y. Ho, and U. Schlichtmann, "Columba S: A scalable co-layout design automation tool for microfluidic large-scale integration," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
[29] 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.
[30] 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.
[31] 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 & Test in Europe Conference & Exhibition, 2017*. IEEE, 2017, pp. 1793–1798.
[32] 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.
[33] "Microfluidic devices – Interoperability Requirements for dimensions, connections and initial device classification," International Organization for Standardization, Geneva, CH, Standard, Mar. 2022.
[34] R. O. McDonald, "Fluidics," *IEEE Transactions on Industry and General Applications*, 2009.
[35] S. Dekker, W. Buesink, M. Blom, M. Alessio, N. Verplanck, M. Hihoud, C. Dehan, W. César, A. Le Nel, A. van den Berg *et al.*, "Standardized and modular microfluidic platform for fast lab on chip system development," *Sensors and Actuators B: Chemical*, vol. 272, pp. 468–478, 2018.
[36] 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.
[37] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.
[38] R. Wille, P. Ebner, M. Emmerich, and M. Takken, "The Munich Microfluidics Toolkit: Design Automation and Simulation Tools for Microfluidic Devices," in *International Conference on Computer Aided Design (ICCAD)*, 2025.
[39] T. Loveless, J. Ott, and P. Brisk, "Time- and resource-constrained scheduling for digital microfluidic biochips," in *ACM/IEEE International Conference on Cyber-Physical Systems*, 2021.
[40] A. Malik, C. Walker, M. O'Sullivan, and O. Sinnen, "Satisfiability modulo theory (SMT) formulation for optimal scheduling of task graphs with communication delay," *Computers & Operations Research*, vol. 89, pp. 113–126, 2018.
[41] C. Cebeci, N. Bjørner, G. Candea, and C. Pit-Claudel, "A conjecture regarding smt instability," in *International Workshop on Satisfiability Modulo Theories*, 2025.

Philipp Ebner received his Master's degree in computer science from the Johannes Kepler University Linz, Austria, in 2021. Currently, he is a Ph.D. student at the Institute for Integrated Circuits and Quantum Computing at the Johannes Kepler University. His main research interest is design automation for microfluidics.

Tim Scholl is a master student with a research interest in microfluidic design automation.

Robert Wille (M'06–SM'15) is a Full and Distinguished Professor at the Technical University of Munich, Germany, and Chief Scientific Officer at the Software Competence Center Hagenberg, Austria. He received the Diploma and Dr.-Ing. degrees in Computer Science from the University of Bremen, Germany, in 2006 and 2009, respectively. Since then, he worked at the University of Bremen, the German Research Center for Artificial Intelligence (DFKI), the University of Applied Science of Bremen, the University of Potsdam, and the Technical University Dresden. From 2015 until 2022, he was Full Professor at the Johannes Kepler University Linz, Austria, until he moved to Munich. His research interests are in the design of circuits and systems for both conventional and emerging technologies. In these areas, he published more than 400 papers and served in editorial boards as well as program committees of numerous journals/conferences such as TCAD, ASP-DAC, DAC, DATE, and ICCAD. For his research, he was awarded, e.g., with Best Paper Awards, e.g., at TCAD and ICCAD, an ERC Consolidator Grant, a Distinguished and a Lighthouse Professor appointment, a Google Research Award, and more.