Automatic Validation and Design of Microfluidic Devices Following the ISO 22916 Standard

Philipp Ebner* and Robert Wille†‡
*Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
†Chair for Design Automation, Technical University of Munich, Germany
‡Software Competence Center Hagenberg GmbH, Austria
philipp.ebner@jku.at, robert.wille@tum.de
https://www.cda.cit.tum.de/research/microfluidics

Abstract—**Microfluidics is an emerging technology for manipulating small amounts of fluids for analytic purposes. Microfluidic devices, so-called *Labs-on-Chip* (LoC), have found widespread applications in medicine, biology, and chemistry. In contrast to other fields, verification and design automation methods for microfluidic chips are not yet as developed due to a lack of common standards. However, recently, there have been new developments with the introduction of the ISO 22916:2022 standard, which contains specifications for microfluidic components and interfaces. In this work, we propose a methodology that, for the first time, enables automatic validation of microfluidic chip designs against place and route constraints in the context of this ISO standard. 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. The applicability and feasibility of the proposed solution are demonstrated in a case study inspired by real-world use cases. 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

The field of microfluidics explores fluid behavior and manipulation at micro- to picoliter sizes [1]. The resulting miniaturization enables us to minimize, combine, and automate fluidic processes such as mixing, heating, incubating, observing, etc., which 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]–[5]. 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 [6], [7].

Despite these successes, the methods to properly design those devices are still not as sophisticated as we take for granted, e.g., in 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 [8]–[10], optimizing flow-based devices [11]–[13], routing microfluidic channels [14]–[18], design automation for electrowetting [19]–[23], and frameworks such as Columba S [24], OpenDrop [25], and for the design of digital microfluidic biochips [26], 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 [27].

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 [28], 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. Motivated by that, an ISO standard (namely ISO 22916:2022 [29]) was introduced to address this issue.

For the first time, this standard provides a common understanding and specifications of how microfluidic components are put together, constituting the basis for more complex and powerful designs. At the same time, it also leads to substantially more complex design tasks and, hence, makes design automation methods inevitable. However, this standard encapsulates many concepts, entities, and constraints that need to be considered when designing such devices, for example, with regard to the dimensions of components [30]. Moreover, the placement of modules and routing of microfluidic channels impose further geometrical constraints, such as minimal distances. It is not trivial and cumbersome to manually validate that all of these requirements are met, and, due to the novelty of the standard, tooling support is still lacking.

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. To this end, we employ an approach using *Satisfiability Modulo Theories* (SMT, [31]), where all relevant geometric properties are modeled as SMT constraints. Consequently, an SMT solver is able to automatically validate that a given design is indeed compliant with this standard or, if this is not the case, pinpoint the designer to the violating components. 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. 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. Then, Sec. III provides detailed descriptions of the necessary encodings, while 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.

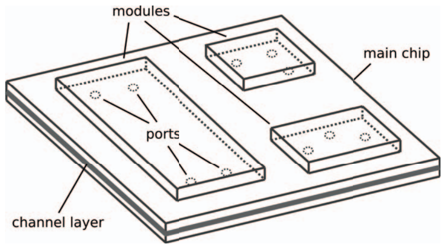
II. Motivation and General Idea

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.

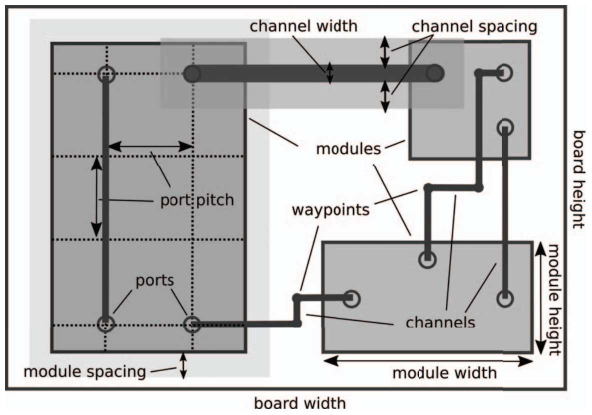
A. The ISO 22916 Standard and Resulting Design Task

The ISO 22916:2022 standard has been defined by the *International Organization for Standardization* and is supposed to provide a standardized description for microfludic devices consisting of a main chip board and multiple modules that are placed on top of the board, as illustrated in Fig. 1. 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.



(a) Overview



(b) Module and channel geometry

Fig. 1: Microfluidic chip with modules

b) Modules: Modules have a predetermined width and height (again, restricted by the ISO standard), as indicated in Fig. 1b, and are placed on top 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) 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*.

d) Channels: Obviously, the ports of modules are supposed to 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, some practical constraints are imposed by them as well: 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 rectilinear shape, i.e., only horizontal and vertical segments are allowed. Each channel consists of multiple waypoints, i.e., intermediate points of channel segments (cf. Fig. 1b). Since there is only one routing layer, channels cannot cross each other. Finally, each channel may have an upper bound for its length.

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, [31]). 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 $\mu\mathrm{m}$ 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 (to which the designer is explicitly pinpointed to). 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 are placeholders for possible values allowed within the standard, which can be found in [29].

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.

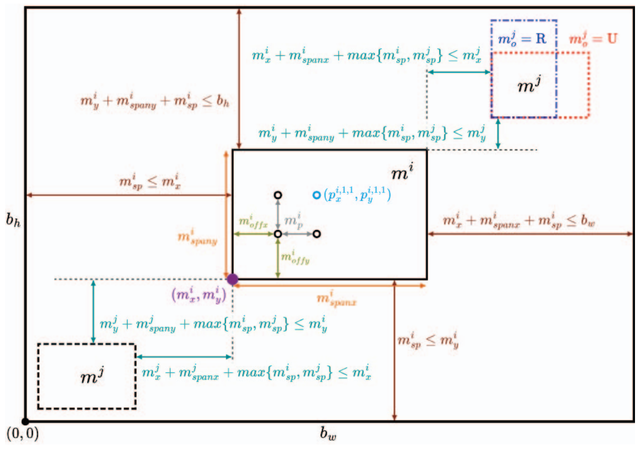

$m_o^j = \mathrm{R}$
$m_o^j = \mathrm{U}$
$m_x^i + m_{spanx}^i + max\{m_{sp}^i, m_{sp}^j\} \leq m_x^j$
m^j
$m_y^i + m_{spany}^i + m_{sp}^i \leq b_h$
$m_y^i + m_{spany}^i + max\{m_{sp}^i, m_{sp}^j\} \leq m_y^j$
m^i
$m_{sp}^i \leq m_x^i$
$(p_x^{i,1,1}, p_y^{i,1,1})$
b_h
m_{offx}^i
m_p^i
m_{spany}^i
$m_x^i + m_{spanx}^i + m_{sp}^i \leq b_w$
m_{offy}^i
(m_x^i, m_y^i)
m_{spanx}^i
$m_y^j + m_{spany}^j + max\{m_{sp}^i, m_{sp}^j\} \leq m_y^i$
$m_{sp}^i \leq m_y^i$
m^j
$m_x^j + m_{spanx}^j + max\{m_{sp}^i, m_{sp}^j\} \leq m_x^i$
$(0, 0)$
b_w

Fig. 2: Module constraints

a) Position: Every module m^i's position is defined by its left-lower 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* ($m_o^i = \mathrm{U}$, the default), *right* (R), *down* (D), or *left* (L). As an example, $m_o^j = \mathrm{U}$ and $m_o^j = \mathrm{R}$ orientations are depicted for m^j in red and blue in Fig. 2.

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

$$\exists n > 0, m > 0 : m_w^i = 15000n \wedge m_h^i = 15000m. \tag{2}$$

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)[1], i.e.,

$$\begin{aligned} m_{spanx}^i &= \begin{cases} m_w^i, & \text{if } m_o^i = \mathrm{U} \,|\, \mathrm{D} \\ m_h^i, & \text{if } m_o^i = \mathrm{R} \,|\, \mathrm{L} \end{cases} \\ m_{spany}^i &= \begin{cases} m_h^i, & \text{if } m_o^i = \mathrm{U} \,|\, \mathrm{D} \\ m_w^i, & \text{if } m_o^i = \mathrm{R} \,|\, \mathrm{L} \end{cases} \end{aligned} \;. \tag{3}$$

d) Spacing Constraints: Modules must keep a certain minimum distance m_{sp}^i to the board edges and to other modules, i.e.,

- *all* of the depicted boundary constraints (brown in Fig. 2) must be satisfied *and*
- *at least one* of the mutual separation constraints (teal in Fig. 2) must be satisfied

e) 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 $1.5\,\mathrm{mm}$, i.e.,

$$\exists n > 0 : m_p^i = 1500n. \tag{4}$$

f) 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

$$m_{px}^i = \left\lfloor \frac{m_w^i}{m_p^i} \right\rfloor - 1, \quad m_{py}^i = \left\lfloor \frac{m_h^i}{m_p^i} \right\rfloor - 1. \tag{5}$$

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

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

$$m_{offx}^i = \left\lfloor \frac{m_w^i - m_p^i\left(m_{px}^i - 1\right)}{2} \right\rfloor, \quad m_{offy}^i = \left\lfloor \frac{m_h^i - m_p^i\left(m_{py}^i - 1\right)}{2} \right\rfloor. \tag{6}$$

Finally, the port position coordinates (p_x^{ijk}, p_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, $p^{i,1,1}$ is depicted in cyan in Fig. 2), i.e.,

$$\begin{aligned} p_x^{ijk} &= \begin{cases} m_x^i + m_{offx}^i + j \cdot m_p^i, & \text{if } m_o^i = \mathrm{U} \\ m_x^i + m_{offy}^i + k \cdot m_p^i, & \text{if } m_o^i = \mathrm{R} \\ m_x^i + m_w^i - m_{offx}^i - j \cdot m_p^i, & \text{if } m_o^i = \mathrm{D} \\ m_x^i + m_h^i - m_{offy}^i - k \cdot m_p^i, & \text{if } m_o^i = \mathrm{L} \end{cases} \\ p_y^{ijk} &= \begin{cases} m_y^i + m_{offy}^i + k \cdot m_p^i, & \text{if } m_o^i = \mathrm{U} \\ m_y^i + m_w^i - m_{offx}^i - j \cdot m_p^i, & \text{if } m_o^i = \mathrm{R} \\ m_y^i + m_h^i - m_{offy}^i - k \cdot m_p^i, & \text{if } m_o^i = \mathrm{D} \\ m_y^i + m_{offx}^i + j \cdot m_p^i, & \text{if } m_o^i = \mathrm{L} \end{cases} \end{aligned} \;. \tag{7}$$

C. Channel Constraints

In this section, the characterization of channels and the resulting constraints are explained, and also illustrated in Fig. 3. 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. 3 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 desireable 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.,

$$\neg s_a^{ik} \Leftrightarrow w_x^{i,k} = w_x^{i,k+1} \wedge w_y^{i,k} = w_y^{i,k+1}. \tag{8}$$

For example, in Fig. 3, $w^{i,3}$ 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.,

$$\forall k > 0 : \neg s_a^{i,k-1} \Rightarrow \neg s_a^{i,k}. \tag{9}$$

b) Segment Types: Additionally, each channel segment has a type s_t^{ij} defining its orientation. Not only are they characterized into horizontal and vertical segments, but also by whether they run in positive or negative direction, resulting in four different types: *right* (R, positive x direction), *up* (U, positive y direction), *left* (L, negative x direction), and *down* (D, negative y direction). As an example, Fig. 3 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, i.e.,

$$\begin{aligned} s_a^{ik} \Rightarrow\; &\left(s_t^{ik} = \mathrm{U} \Leftrightarrow w_x^{i,k} = w_x^{i,k+1} \wedge w_y^{i,k} < w_y^{i,k+1}\right) \wedge \\ &\left(s_t^{ik} = \mathrm{R} \Leftrightarrow w_x^{i,k} < w_x^{i,k+1} \wedge w_y^{i,k} = w_y^{i,k+1}\right) \wedge \\ &\left(s_t^{ik} = \mathrm{D} \Leftrightarrow w_x^{i,k} = w_x^{i,k+1} \wedge w_y^{i,k} > w_y^{i,k+1}\right) \wedge \\ &\left(s_t^{ik} = \mathrm{L} \Leftrightarrow w_x^{i,k} > w_x^{i,k+1} \wedge w_y^{i,k} = w_y^{i,k+1}\right). \end{aligned} \tag{10}$$

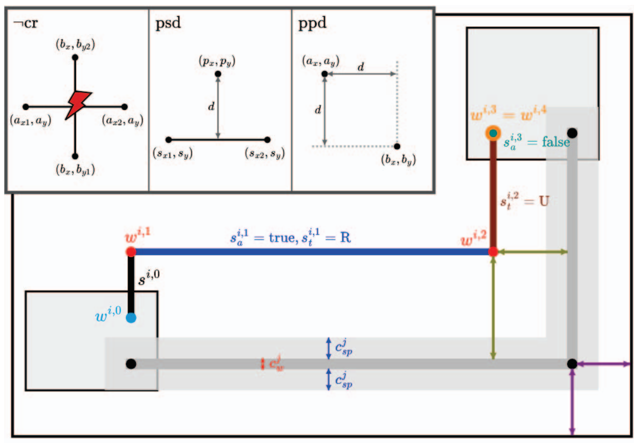


Fig. 3: Channel constraints

Finally, consecutive segments of the same channel should alternate between horizontal and vertical types[1], i.e.,

$$\forall k > 0 : s_a^{ik} \Rightarrow \left(s_t^{i,k} = \text{U} \,|\, \text{D} \Rightarrow \neg\left(s_t^{i,k-1} = \text{U} \,|\, \text{D}\right)\right) \wedge \left(s_t^{i,k} = \text{R} \,|\, \text{L} \Rightarrow \neg\left(s_t^{i,k-1} = \text{R} \,|\, \text{L}\right)\right). \tag{11}$$

c) 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. 3 by the purple arrows), i.e.,

$$\frac{c_w^i}{2} + c_{sp}^i \leq w_x^{ik}, \quad \frac{c_w^i}{2} + c_{sp}^i \leq w_y^{ik}, \quad w_x^{ik} + \frac{c_w^i}{2} + c_{sp}^i \leq b_w, \quad w_y^{ik} + \frac{c_w^i}{2} + c_{sp}^i \leq b_h, \tag{12}$$

where c_w^i is the channel width and c_{sp}^i is the channel spacing (as depicted in Fig. 3 as c_w^j in red and c_{sp}^j in blue).

d) Segment Crossing: Any two segments of arbitrary channels must never cross each other. To avoid that, we first define a *no crossing* predicate $\neg cr$ that yields true only if the interiors of two orthogonal segments do not intersect (cf. Fig. 3), i.e.,

$$\neg cr\left(a_{x1}, a_{x2}, a_y, b_x, b_{y1}, b_{y2}\right) := a_{x2} \leq b_x \vee a_{x1} \geq b_x \vee b_{y2} \leq a_y \vee b_{y1} \geq a_y, \tag{13}$$

where a is a horizontal segment with $a_{x1} < a_{x2}$ and b is a vertical segment with $b_{y1} < b_{y2}$[2]. By employing this predicate to all pairs of segments that actually *can* collide, i.e, active segments in orthogonal position, it is ensured that no segment crossings occur[3], i.e.,

$$\begin{aligned} s_a^{ik} \wedge s_a^{jl} \Rightarrow &\left(s_t^{ik} = \text{U} \wedge s_t^{jl} = \text{R} \Rightarrow \neg cr\left(w_y^{ik}, w_y^{i,k+1}, w_x^{ik}, w_y^{jl}, w_x^{jl}, w_x^{j,l+1}\right)\right) \wedge \\ &\left(s_t^{ik} = \text{U} \wedge s_t^{jl} = \text{L} \Rightarrow \neg cr\left(w_y^{ik}, w_y^{i,k+1}, w_x^{ik}, w_y^{jl}, w_x^{j,l+1}, w_x^{jl}\right)\right) \wedge \\ &\qquad \dots \\ &\wedge \left(s_t^{ik} = \text{L} \wedge s_t^{jl} = \text{D} \Rightarrow \neg cr\left(w_x^{i,k+1}, w_x^{ik}, w_y^{ik}, w_x^{jl}, w_y^{j,l+1}, w_y^{jl}\right)\right). \end{aligned} \tag{14}$$

e) 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. To this end, 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.,

$$\begin{aligned} &psd(p_x, p_y, s_{x1}, s_{x2}, s_y, d) := \\ &\quad p_x \leq s_{x1} \vee p_x \leq s_{x2} \vee p_y + d \leq s_y \vee s_y + d \leq p_y \end{aligned} \tag{15}$$

where s is a horizontal segment with $s_{x1} < s_{x2}$ (a graphic example is illustrated in Fig. 3)[2]. Again, this predicate is applied to all pairings of active segments and waypoints, and the minimum required distance d_{ij} (illustrated by the olive arrows in Fig. 3) is computed from both involved channels' width c_w^i, c_w^j and spacing c_{sp}^i, c_{sp}^j, i.e.,

$$\begin{aligned} s_a^{ik} \Rightarrow &\left(s_t^{ik} = \text{U} \Rightarrow psd\left(w_y^{jl}, w_x^{jl}, w_y^{ik}, w_y^{i,k+1}, w_x^{ik}, d_{ij}\right)\right) \wedge \\ &\left(s_t^{ik} = \text{D} \Rightarrow psd\left(w_y^{jl}, w_x^{jl}, w_y^{i,k+1}, w_y^{ik}, w_x^{ik}, d_{ij}\right)\right) \wedge \\ &\left(s_t^{ik} = \text{R} \Rightarrow psd\left(w_x^{jl}, w_y^{jl}, w_x^{ik}, w_x^{i,k+1}, w_y^{ik}, d_{ij}\right)\right) \wedge \\ &\left(s_t^{ik} = \text{L} \Rightarrow psd\left(w_x^{jl}, w_y^{jl}, w_x^{i,k+1}, w_x^{ik}, w_y^{ik}, d_{ij}\right)\right) \end{aligned} \tag{16}$$

with

$$d_{ij} := \left\lceil \frac{c_w^i + c_w^j}{2} \right\rceil + \max\{c_{sp}^i, c_{sp}^j\}. \tag{17}$$

f) 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. 3)[2] guarantees that points a and b have a minimum axis-aligned distance of d, i.e.,

$$\begin{aligned} &ppd(a_x, a_y, b_x, b_y, d) := \\ &a_x + d \leq b_x \vee a_y + d \leq b_y \vee b_x + d \leq a_x \vee b_y + d \leq a_y. \end{aligned} \tag{18}$$

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

$$ppd(w_x^{ik}, w_y^{ik}, w_x^{jl}, w_y^{jl}, d_{ij}) \quad \text{if } i \neq j \tag{19}$$

$$s_a^{i,k-1} \Rightarrow ppd(w_x^{ik}, w_y^{ik}, w_x^{il}, w_y^{il}, d_{ij}) \quad \text{if } i = j \wedge k > l \tag{20}$$

with d_{ij} identical to the previously defined Eq. 17.

g) Port Connections: The first waypoint $w^{i,0}$ and last waypoint w^{i,s_n^i} of a channel are each assigned to their corresponding port position (cf. Eq. 7), i.e.,

$$w_x^{i,0} = p_x^{a_m,a_x,a_y}, \quad w_y^{i,0} = p_y^{a_m,a_x,a_y}, \quad w_x^{i,s_n^i} = p_x^{b_m,b_x,b_y}, \quad w_y^{i,s_n^i} = p_y^{b_m,b_x,b_y} \tag{21}$$

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. 3.

h) Upper Bound for Channel Length: Finally, the channel length, i.e., the sum of all of its segment lengths, must satisfy an upper bound c_l^i, i.e.,

$$\sum_{j>0} \left(|w_x^{ij} - w_x^{i,j-1}| + |w_y^{ij} - w_y^{i,j-1}|\right) \leq c_l^i. \tag{22}$$

[2]Note that the axes can be swapped here.
[3]Some analogous terms are omitted due to lack of space.

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 [31] 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. 6). In these cases, only the resulting value is supplied to the solver.

Using this implementation, a tool results 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 respectively considered design. 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; otherwise, the solver determines what constraint(s) have been violated, which can be directly "translated" back to the design and, hence, be used to pinpoint the designer to the design components that are not compliant.
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.

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 [5], [30]—eventually resulting in designs including 3 to 9 modules and 10 to 24 channels, as well as different complexities with respect to channel length, port positions, etc. Those designs are denoted *Design 1-3* in the following.

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

B. Validation

Table I summarizes the (representative subset of) results obtained when considering the validation of ISO-compliance. The first three columns provide the name and characteristics of the considered design. Then, it is denoted whether the corresponding design has been validated as ISO-compliant or

TABLE I: Validation cases

Design	Modules	Channels	Valid	Violations				Time
				V_1	V_2	V_3	V_4	[s]
1	3	10	✓					0.38
				✗				0.11
					✗			0.31
						✗		0.35
							✗	0.38
2	3	15	✓					1.78
				✗				0.19
					✗			1.45
						✗		1.69
							✗	1.56
3	9	24	✓					6.98
				✗				0.68
					✗			3.81
						✗		5.12
					✗		✗	5.43

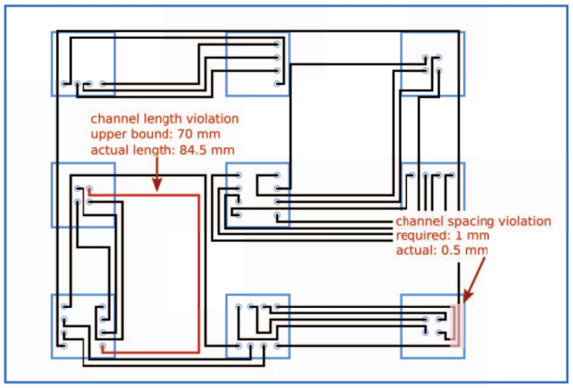


Fig. 4: Violation of constraints for chip 3

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 moderate runtime (i.e., a few seconds). Obviously, the runtime 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), and
- violations of channel lengths (denoted V_4).

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. 4 for *Design 3*. Here, a channel length violation as well as a channel spacing violation exist. Thus far (i.e., without the proposed tool), the designer needed to consider the entire board, including all modules, ports, as well as channels, and manually check for any violations. With the proposed tool, they are directly pinpointed to the violations (highlighted red in Fig. 4)—a substantial improvement of the current design process.

C. Design Completion

Table II summarizes the (representative subset of) results obtained when considering the completion of ISO-compliant designs. To "emulate" intermediate but incomplete designs, which may have resulted during the design phase, we removed some module positions, module orientations, and/or channels (i.e., their waypoints). The first columns of Table II indicate the resulting cases (providing the number of given as well as generated module placements, module orientations, and

TABLE II: Design generation cases

Design	Modules			Channels			Time
	placement given	orientation given	$\sum$	given	generated	$\sum$	[s]
1	3	3	3	0	10	10	3.29
	0	3	3	0	10	10	4.93
	0	0	3	0	10	10	19.06
2	3	3	3	5	10	15	7.22
	3	3	3	0	15	15	17.82
	0	3	3	0	15	15	40.17
	0	0	3	0	15	15	767.95
3	9	9	9	12	12	24	35.10
	9	9	9	0	24	24	613.96
	7	9	9	0	24	24	417.46
	4	9	9	0	24	24	627.51

channel placements). Finally, the last column again provides the runtime (in seconds) of the design completion process.

The results clearly show that design completion is a computationally significantly more complex task. This is no surprise; after all, the overall design task is an $\mathcal{NP}$-hard endeavor [32]. 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 *Design 1*), the *entire* design can be automatically generated in less than a minute (cf. the case in Table II, where no module placements, module orientations, or channels are given). 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, pinpoints the designer to the corresponding problem.

VI. Conclusion

In this work, we presented a method for the validation of microfluidic chip designs that conform to the ISO 22916 standard. In order to achieve this objective, we converted the criteria mandated by the standard and further geometric constraints imposed by them into SMT constraints. Subsequently, an SMT solver was employed to prove that a specific chip design effectively satisfies the given criteria. Additionally, the resulting 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 [5], [30]), we demonstrated how the resulting tool substantially improves the design process for microfluidic chips.

Acknowledgment

The authors thank Joshua Loessberg-Zahl, Eric Safai, and Aniruddha Paul for the fruitful discussions. This work has partially been supported by the FFG project AUTOMATE (project number: 890068) as well as by BMK, BMDW, and the State of Upper Austria in the frame of the COMET Programme managed by FFG.

References

[1] G. M. Whitesides, "The origins and the future of microfluidics," *Nature*, vol. 442, no. 7101, pp. 368–373, 2006.
[2] 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.
[3] 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.
[4] Y. Ding, P. D. Howes, and A. J. deMello, "Recent advances in droplet microfluidics," *Analytical chemistry*, vol. 92, no. 1, pp. 132–149, 2019.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] P. Zhang, C. Chen, F. Guo, J. Philippe, Y. Gu, Z. Tian, H. Bachman, L. Ren, S. Yang, Z. Zhong *et al.*, "Contactless, programmable acoustofluidic manipulation of objects on water," *Lab on a Chip*, vol. 19, no. 20, pp. 3397–3404, 2019.
[20] 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.
[21] 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.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] 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.
[28] 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.
[29] "Microfluidic devices – Interoperability requirements for dimensions, connections and initial device classification," International Organization for Standardization, Geneva, CH, Standard, Mar. 2022.
[30] 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.
[31] 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.
[32] 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.