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

Philipp Ebner* and Robert Wille†‡

I. INTRODUCTION

The field of microfluidics is an emerging technology for manipulating small amounts of fluids for analytic purposes. Microfluidic devices, so-called Labs-on-Chip (LoCs), 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 and design tooling support for the creation of 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 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

Abstract—Microfluidics is an emerging technology for manipulating small amounts of fluids for analytic purposes. Microfluidic devices, so-called Labs-on-Chip (LoCs), 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 and design tooling support for the creation of 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 tool is available at https://github.com/cda-tum/mmft-iso-designer

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 remaining 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 microfluidic 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:

1) 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. 1), are predetermined according to the ISO 22916 standard in order to ensure compatibility with certain laboratory equipment, such as microscopes.
ports on the grid is called the uniform grid (depicted in Fig. 1b). The distance between two positions of these ports are spread across the module on a the port opening where no other ports or modules occur. The requires that a square with a predefined area be set around to be cleanly connected to the main board. Generally, this need to have a cleared perimeter without obstacles in order doing that, they frequently work with partial designs, need tries to connect them with the corresponding channels. While example, the designer first places the modules and, afterwards, above. This is typically done in an iterative fashion. For 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, ensures that indeed all design specifications are satisfied and the resulting chip 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. 2c), of the main chip board are imposed by ISO 22916, leading to a series of constraints in the form of

\[
\sqrt{b_w - v_1 \land b_h - v_2}, 
\]

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 characteriza- and the resulting constraints are explored in the following, all of which are also illustrated in Fig. 2.
modules, i.e., \( m_1 \) must be a multiple of \( m \) has a set of ports distributed on a uniform grid. The distance orientations are depicted for \( m \) point in Fig. 2).

As a shorthand notation, we write \( x \) spanx and \( y \) spany 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 i.e.,

\[
m^i_{spanx} = \begin{cases} m^i_w, & \text{if } m^j_o = U \mid D \\ m^i_h, & \text{if } m^j_o = R \mid L \\ \end{cases} \quad (m^i_{spany} = \begin{cases} m^i_w, & \text{if } m^j_o = U \mid D \\ m^i_h, & \text{if } m^j_o = R \mid L \\ \end{cases})
\]

Spacing Constraints: Modules must keep a certain minimum distance \( m^i_{sp} \) 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

d) Port Pitch: As introduced in Sec. 1, each module has a set of ports distributed on a uniform grid. The distance between ports, the port pitch \( m^i_p \) (illustrated in gray in Fig. 2), must be a multiple of 1.5 mm, i.e.,

\[
\exists n > 0 : m^i_p = 1500n. \quad (4)
\]

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

\[
m^i_{px} = \left\lfloor \frac{m^i_w}{m^i_p} \right\rfloor - 1, \quad m^i_{py} = \left\lfloor \frac{m^i_h}{m^i_p} \right\rfloor - 1. \quad (5)
\]

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

\[
m^i_{offx} = \left\lfloor \frac{m^i_w - m^i_o \cdot (m^i_p - 1)}{2} \right\rfloor, \quad m^i_{offy} = \left\lfloor \frac{m^i_h - m^i_o \cdot (m^i_p - 1)}{2} \right\rfloor. \quad (6)
\]

Finally, the port position coordinates \( (p^i_{xj}, p^i_{yj}) \) depend on the position and orientation of the module, where \( j, k \) with \( 0 \leq j, k < m^i_{px} \) and \( 0 \leq j, k < m^i_{py} \) are the indices of the port on the grid (as an example, \( p^i_{1+1,1} \) is depicted in cyan in Fig. 2), i.e.,

\[
p^i_{xj} = \begin{cases} m^i_x + m^i_{offx} + j \cdot m^i_p, & \text{if } m^j_o = U \\ m^i_x + m^i_{offx} + k \cdot m^i_p, & \text{if } m^j_o = R \\ m^i_x + m^i_{offx} - j \cdot m^i_p, & \text{if } m^j_o = D \\ m^i_x + m^i_{offx} - k \cdot m^i_p, & \text{if } m^j_o = L \\ \end{cases} \quad (7)
\]

\[
p^i_{yj} = \begin{cases} m^i_y + m^i_{offy} + j \cdot m^i_p, & \text{if } m^j_o = U \\ m^i_y + m^i_{offy} + k \cdot m^i_p, & \text{if } m^j_o = R \\ m^i_y + m^i_{offy} - j \cdot m^i_p, & \text{if } m^j_o = D \\ m^i_y + m^i_{offy} - k \cdot m^i_p, & \text{if } m^j_o = L \\ \end{cases}
\]

C. Channel Constraints

In this section, the characterization of channels and the resulting constraints are explained, and also illustrated in Fig. 3 Each channel \( C \) consists of a sequence of segments \( s^{ij} \) with \( 0 \leq j < s^i \) (where \( s^i \) is the total number of segments of \( C \)) and waypoints \( w^{ik} \) with \( 0 \leq k \leq s^i \). 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 desirable that a channel uses all of its \( s^i \) 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 \( s^i \). Each segment can be active (\( s^{ik} = \text{true} \)) or inactive (\( s^{ik} = \text{false} \)). If a segment is inactive, both of its endpoints collapse into one, i.e.,

\[
\neg s^{ik} \Leftrightarrow w^{i,k} = w^{i,k+1} \land w^{i,k} = w^{i,k+1}.
\]

For example, in Fig. 3 \( w^{i,3} \) and \( w^{i,5} \) (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.,

\[
\forall k > 0 : \neg s^{i,k-1} \Rightarrow \neg s^{i,k}.
\]

b) Segment Types: Additionally, each channel segment has a type \( s^{ik} \) 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.,

\[
\begin{align*}
s^{ik}_R &\Leftrightarrow \begin{cases} s^{ik} = U \Leftrightarrow w^{i,k} = w^{i,k+1} \land w^{i,k} < w^{i,k+1} \\
&\Leftrightarrow w^{i,k} < w^{i,k+1} \land w^{i,k} = w^{i,k+1} \\
&\Leftrightarrow w^{i,k+1} > w^{i,k} \land w^{i,k} = w^{i,k+1} \end{cases} \\
&\Leftrightarrow w^{i,k} = w^{i,k+1} \land w^{i,k} > w^{i,k+1} \end{cases} \\
&\Leftrightarrow w^{i,k} = w^{i,k+1} \land w^{i,k} > w^{i,k+1} \end{cases} \\
&\Leftrightarrow w^{i,k} = w^{i,k+1} \land w^{i,k} > w^{i,k+1} \end{cases} \\
&\Leftrightarrow w^{i,k} = w^{i,k+1} \land w^{i,k} > w^{i,k+1} \end{cases}
\]

As a shorthand notation, we write \( x = U \mid D \) for \( x = U \lor x = D \).
to all pairs of segments that actually crossings occur between segments in orthogonal position, it is ensured that no segment vertical segment with a horizontal segment with

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.,

\[
\begin{align*}
    s^a_i \Rightarrow (s^a_i = \text{U} \mid D) \Rightarrow \left( s^{i,k-1} = \text{U} \mid D \right) \land \\
    (s^i_k = \text{R} \mid L) \Rightarrow \left( s^{i,k-1} = \text{R} \mid L \right).
\end{align*}
\]

(11)

\[\text{with } d_{ij} := \left[ \frac{c_w^i + c_y^j}{2} \right] + \max\{c^j_{sp}, c^i_{sp}\}.\]

(17)

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

\[
-\text{cr} (a_{x_1}, a_{x_2}, a_y, b_x, b_{y_1}, b_{y_2}) := \left( a_{x_2} \leq b_x \land a_y \leq b_{y_2} \land b_{y_1} \geq a_y \right) \land \left( b_{y_1} \leq a_y \right).
\]

(13)

where \(a\) is a horizontal segment with \(a_{x_1} < a_{x_2}\) and \(b\) is a vertical segment with \(b_{y_1} < b_{y_2}\). 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, i.e.,

\[
\begin{align*}
    s^i_{ik} \land s^j_{jl} \Rightarrow (s^i_{ik} = \text{U} \land s^j_{jl} = \text{R} \Rightarrow -\text{cr} (w^i_{ik}, w^j_{j1}, w^i_{ik}, w^j_{j1}, w^i_{ik}, w^j_{j1}, \ldots) ) \land \\
    (s^i_{ik} = \text{U} \land s^j_{jl} = \text{L} \Rightarrow -\text{cr} (w^i_{ik}, w^j_{j1}, w^i_{ik}, w^j_{j1}, w^i_{ik}, w^j_{j1}, \ldots) ) \land \\
    \ldots \land (s^j_{jl} = \text{L} \land s^i_{ik} = \text{D} \Rightarrow -\text{cr} (w^i_{ik}, w^j_{j1}, w^i_{ik}, w^j_{j1}, w^i_{ik}, w^j_{j1}, \ldots) ).
\end{align*}
\]

(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.,

\[
psd(p_x, p_y, s_{x1}, s_{x2}, s_{y}, d) := \left( p_x \leq s_{x1} \lor p_x \leq s_{x2} \lor p_y + d \leq s_{y} \lor s_y + d \leq p_y \right)
\]

(15)

where \(s\) is a horizontal segment with \(s_{x1} < s_{x2}\) (a graphic example is illustrated in Fig. 3). 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 \(c^i_{sp}, c^j_{sp}\), i.e.,

\[
s_{ik} \Rightarrow (s^i_{ik} = \text{U} \Rightarrow psd (w^i_{ik}, w^i_{ik}, w^y_{ik}, w^y_{ik}, w^x_{ik}, d_{ij})) \land \\
(s^i_{ik} = \text{D} \Rightarrow psd (w^i_{ik}, w^i_{ik}, w^y_{ik}, w^y_{ik}, w^x_{ik}, d_{ij})) \land \\
(s^i_{ik} = \text{R} \Rightarrow psd (w^i_{ik}, w^i_{ik}, w^y_{ik}, w^y_{ik}, w^x_{ik}, d_{ij})) \land \\
(s^i_{ik} = \text{L} \Rightarrow psd (w^i_{ik}, w^i_{ik}, w^y_{ik}, w^y_{ik}, w^x_{ik}, d_{ij})).
\]

(16)

with \(d_{ij} := \left[ \frac{c_w^i + c_y^j}{2} \right] + \max\{c^j_{sp}, c^i_{sp}\}\).
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. III-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., $s_{k}^{ij}$ and $s_{l}^{ij}$) 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](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 not. An open-source implementation of the resulting tool is available at https://github.com/cda-tum/mmft-iso-designer.

<table>
<thead>
<tr>
<th>Design</th>
<th>Modules</th>
<th>Channels</th>
<th>Valid</th>
<th>Violations</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>10</td>
<td>✓</td>
<td>✗ ✗</td>
<td>0.38</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>15</td>
<td>✓</td>
<td>✗ ✗</td>
<td>1.18</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>24</td>
<td>✓</td>
<td>✗ ✗</td>
<td>6.98</td>
</tr>
</tbody>
</table>

Table I: Validation cases

![Fig. 4: Violation of constraints for chip 3](image)

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 II 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 number of given as well as generated module placements, module orientations, and
TABLE II: Design generation cases

<table>
<thead>
<tr>
<th>Design placement</th>
<th>Modules orientation</th>
<th>Channels given</th>
<th>Channels generated</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>3</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>9</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>9</td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

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 $NP$-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 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 Lossberg-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