Close-to-Optimal Placement and Routing for Continuous-Flow Microfluidic Biochips

Andreas Grimmer[1] Qin Wang[2] Hailong Yao[2] Tsung-Yi Ho[3] Robert Wille[1]
[1]Institute for Integrated Circuits, Johannes Kepler University, Linz, Austria
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]National Tsing Hua University, Taiwan
andreas.grimmer@jku.at woodythu@163.com hailongyao@mail.tsinghua.edu.cn
tyho@cs.nthu.edu.tw robert.wille@jku.at

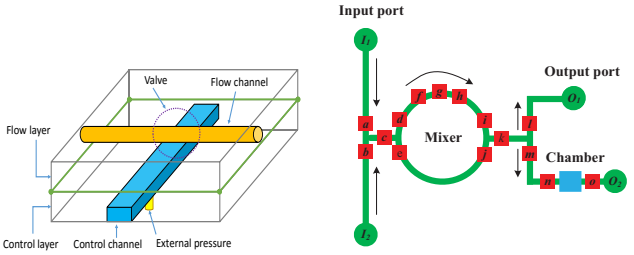
Abstract—Continuous-flow microfluidics rapidly evolved in the last decades as a solution to automate laboratory procedures in molecular biology and biochemistry. Therefore, the physical design of the corresponding chips, i.e., the placement and routing of the involved components and channels, received significant attention. Recently, several physical design solutions for this task have been presented. However, they often rely on general heuristics which traverse the search space in a rather arbitrary fashion and, additionally, consider placement and routing independently from each other. Consequently, the obtained results are often far from being optimal. In this work, a methodology is proposed which aims for determining close-to-optimal physical designs for continuous-flow microfluidic biochips. To this end, we consider all – or, at least, as much as possible – of the valid solutions. As this obviously yields a significant complexity, solving engines are utilized to efficiently traverse the search space and pruning schemes are proposed to reduce the search space without discarding too many promising solutions. Evaluations show that the proposed methodology is capable of determining optimal results for small experiments to be realized. For larger experiments, close-to-optimal results can efficiently be derived. Moreover, compared to the current state-of-the-art, improvements of up to 1-2 orders of magnitude can be observed.

I. Introduction

Flow-based microfluidic biochips rapidly evolved as they revolutionized the traditional biology and biochemistry by automating laboratory tasks and reducing sample volumes [1–3]. These flow-based biochips are especially suited for high-throughput applications and are e.g., successfully used for the extraction of nucleic aids [4], protein crystallization [5], immunoassays [6], or DNA synthesizing [7].

A flow-based microfluidic biochip consists of hundreds or even thousands of integrated *microvalves* [3,8], which are used to control the flow of the liquids. Such biochips are made of an elastomer material (polydimethylsiloxane, PDMS) and the micro structures are produced in a multilayer soft-lithography process [9,10]. Fig. 1a shows the respective schematic of these chips: Each flow-based biochip consists of a two-layer channel circuitry, where the *control layer* contains logic to trigger the microvalves in order to either close or open a channel in the *flow layer*. If an externally produced, pneumatic pressure to the control channel is applied, the elastomer material pinches the flow layer and, hence, blocks the fluid flow. After releasing this external pressure, the elastomer material of the valve restores back and the fluid flow resumes.

By combining and controlling the closing and opening of multiple valves, more complex operations, such as merging, splitting, dispensing, and mixing can be built [11]. Fig. 1b shows a schematic view of a biochip containing a mixer component. In order to mix two fluids, they have to be dispensed from the input ports I_1 and I_2. Then, the valves c as well as k are closed and the mixing is started. The mixing is conducted by actuating the valves f, g, and h in a peristaltic sequence at a high frequency ($\sim 100Hz$). This forces the liquids in the mixer to circulate around and mix together [8].



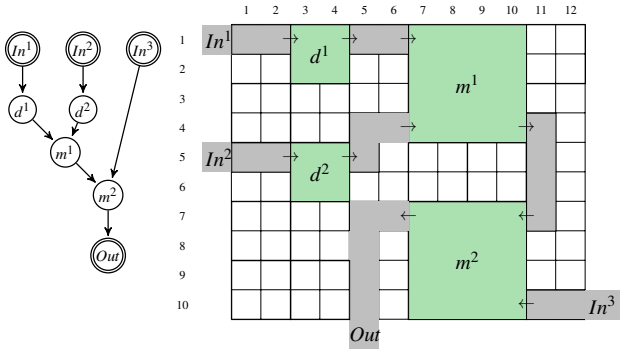
(a) Schematic view (b) A mixer component

Fig. 1. Continuous-flow microfluidic biochip

In the physical design of such biochips, a *placement* and *routing* (P&R) of all those *components* and their *flow channels*, respectively, has to be determined. The quality of the physical design is thereby measured by (1) the amount of flow channel intersections, (2) the flow channel length, and (3) the resulting flow layer size. In particular, a low amount of flow channel intersections is desirable, as each flow channel intersection requires a multiplexer component [12]. Such a multiplexer consists of four valves, which are used to separate the two channels and, in turn, avoid an unexpected mixing of the fluids. A short flow channel length is desirable as it increases the structural reliability and, by this, the fabrication success [13] while, at the same time, reducing the fluid transportation latency [14].

In the recent past, automated design methods for the physical design of continuous-flow microfluidic biochips have been presented [12,15,16]. However, all of them suffer from the fact that they rely on heuristics, which traverse the search space in an arbitrary fashion and, therefore, cannot guarantee the determination of a good design with respect to the different quality criteria. Furthermore, existing design methods consider the placement of the components and the routing of the channels as two independent steps – further contributing to the fact that the obtained results are far from being optimal.

In this work, we present a methodology which tackles these drawbacks and is capable of producing close-to-optimal designs. Determining close-to-optimal or even optimal designs requires that all possible designs are considered, which in turn is only possible when the placement and routing is conducted in a single step. This results in a significant complexity, which is handled by using powerful satisfiability solvers as well as dedicated pruning methods. Experimental evaluations show that the proposed methodology indeed determines



(a) Experiment (b) Design solution

Fig. 2. Running example

designs, which are close-to-optimal and, by this, outperforms the current state-of-the-art by up to 1-2 orders of magnitude.

The remainder of the paper is organized as follows: The next section introduces the physical design task and reviews the related work. Section III motivates the proposed methodology and how to deal with the resulting complexity. In Section IV and Section V, details on the optimal placement and routing method as well as the proposed pruning schemes are provided, respectively. The results of our evaluation are summarized in Section VI and, finally, the paper is concluded in Section VII.

II. Background

This section describes the considered physical design task and introduces the notion used in this work. Besides that, this section reviews and discusses recently proposed design solutions.

A. Placement & Routing for Continuous-Flow Microfluidics

In a design context, the flow layer of a continuous-flow microfluidic biochip is discretized as a grid. Then, physical design has to place the components and channels specified in an experiment onto this grid. These components are bounded by rectangular boxes where the heights and widths of these boxes are specified as a number of grid cells. Similarly, the channel has a width of one grid cell. More formally, the input for a physical design method is given as

- a discretized grid of size $w \times h$, where w is the width and h is the height of the grid,
- a set of components V, which have to be placed on the grid (each component $v \in V$ has a given size $w_v \times h_v$ and corresponding input/output ports), and
- a set of channels E (with $E \subseteq V \times V$), which connect different components.

This input can be represented as a graph $G = (V,E)$, where a node is either a component, an input, or an output and the edges represent channels. Fig. 2a shows an experiment with two detectors (d^1 and d^2) and two mixers (m^1 and m^2), which is used as running example in this work. In addition to this graph, information on the component sizes (e.g., that component d^1 has a size of $(w_{d^1} \times h_{d^1}) = (2,2)$) as well as their input/output ports (e.g., that component d^1 has its input port on the western side in the top and its output port on the eastern side in the top) are provided.

Using this input, a placement and routing method (P&R method) has to determine a design in which all components and all channels are placed and routed onto a grid. Grid cells can be occupied by a single component (or a part of a component), occupied by one or two channels (the latter leading to a channel intersection), or can be empty. The quality of a design is measured by

- its number of flow channel intersections,
- its overall flow channel length, and
- its grid size.

Fig. 2b shows a possible design of the running example on a 12×10 grid with no channel intersections and a channel length of 20.

B. Related Work

The design of continuous-flow microfluidic biochips can be divided into several main steps: Methods exist for architectural synthesis and resource binding (e.g., [13, 17, 18]). The latter yields a precise specification serving as input for the physical design of the flow layer. This physical design is then the basis for the design of the control layer, for which automated design methods (e.g., [8, 19–21]) aim for e.g., minimizing the amount of control pins and the pressure propagation delay.

In this work, we focus on the physical design of the flow layer. Here, various solutions have been presented. For example, the work presented in [16] introduces a placement algorithm based on simulated annealing which takes "routability-issues" into consideration. The work in [15] proposes a routing algorithm for flow channels but assumes a fixed placement. Hence, both works [15, 16] cover a single part of the physical design only. The approach presented in [12] covers both aspects, placement and routing, in a single work by proposing a sequence-pair-based placement and a negotiation-based routing algorithm. The work aims for iteratively eliminating over-congested regions (which may cause problems for routing) by increasing the spacing of components. But also here, a separate method for placement and a separate method for routing is eventually employed.

Overall, all solutions for the physical design of continuous-flow microfluidic biochips consider placement and routing as two – more or less independent – steps. This often leads to inappropriate designs since the total number of channel intersections and the channel length can only be estimated when conducting the placement (e.g., by approximating the channels with lines between components and, then, calculating the intersection points or by considering the Manhattan distances between components). Moreover, in order to avoid routing failures, placement methods usually allocate extra space which often is not needed but, of course, increases the size of the resulting design. On top of that, all solutions proposed thus far rely on general heuristics which traverse the search space in a rather arbitrary fashion. Eventually, this yields solutions which are far from being optimal.

III. Close-to-Optimal Placement & Routing

In this work, we propose an alternative design approach, which conducts the placement and routing for continuous-flow microfluidic biochips in a single step. The main premise is not to heuristically and incompletely traverse the search space in order to determine a rather arbitrary solution, but to determine a realization which is as close as possible to the optimal solution with respect to the needs of the designer. In the ideal case, this would include the consideration of all possible solutions – obviously a computationally very expensive task. In order to cope with this complexity, a methodology is proposed which relies on

- powerful solving engines that can cope with large search spaces and the underlying complexity as well as
- pruning schemes which reduce the search space without, at the same time, discarding promising solutions.

In this section, we briefly sketch the anticipated core concepts of the proposed methodology. Details on their implementation are, afterwards, provided in Section IV and Section V.

A. Using Satisfiability Solvers for Optimal P&R

In order to guarantee the best possible physical design, *all* possibilities have to be considered. In a naive fashion, this could be conducted by enumeratively generating and evaluating all possible designs. More precisely, for a given grid size, a given list of components, as well as all required channels between components, all possible placements of components and all possible routings between components are iteratively considered. From all these realizations, those are eventually discarded which do not realize the respectively given experiment. Afterwards, from the remaining solutions, the design is picked which fits best to the designer's needs, e.g., has the smallest number of intersections.

Moreover, an approach like that would eventually yield an *optimal Placement and Routing method* (optimal P&R method) which, for a given input (i.e., a given experiment to be realized, a given grid size, etc.), determines a solution satisfying the given constraints (e.g., a restricted number of intersections). If no design satisfying the given constraints is possible, the optimal P&R method would prove the non-existence of a solution (since all possibilities have been considered). This can then also be used to determine optimal results by e.g., checking for a solution with none intersections and iteratively increasing the number of intersections until a solution is found (which, due to the iterative increases, has to be optimal)[1].

However, a naive approach based on enumeration would not be capable of determining designs of appropriate size – the sheer number of possibilities would be too huge. Hence, we are proposing to utilize the computational power of solving engines such as satisfiability solvers (see e.g., [22, 23]). They are heavily optimized and additionally employ highly sophisticated deduction and learning schemes which allow them to automatically prune large parts of the search space without discarding any valid solution. Applying satisfiability solvers allows for the consideration of all possible solutions without the need to explicitly enumerate each and everyone of it. For the design of other biochip technologies, this concept has already successfully been employed (see e.g., [24, 25]).

B. Search Space Pruning

Despite the efficiency of satisfiability solvers, the complexity is still significant when *all* possible solutions shall be considered. Already with a grid size of 4×4, just two components to be placed, and four channels to be routed, approx. $(2+4+1)^{4 \cdot 4} > 10^{13}$ combinations have to be considered (more precisely, $4 \cdot 4$ cells which each can be assumed to be occupied by one of the two components, one of the four channels, or by neither). This yields a complexity which, even with satisfiability solvers, can only be handled for small experiments to be realized.

In order to address that, we additionally propose to further prune the search space – in contrast to the pruning of satisfiability solvers, by additionally accepting a discard of valid solutions. While this obviously would lead to solutions for which no optimality is guaranteed anymore, we aim to prune the search space in a fashion which does not discard promising solutions. To this end, two schemes are proposed:

- *Downscaling*
  We apply a simplification which reduces the search space by scaling down the size of all components. This significantly reduces the search space and will make the resulting problem more likely solvable by the optimal P&R method. If a result for this instance is determined, the resulting design is scaled up again. This keeps the number of channel intersections optimal while optimality of the grid size and channel length is only harmed by a bounded factor (namely the scaling factor).
- *Partitioning*
  We partition the initial experiment into sub-problems. To this end, the experiment is split into equally large subsets and, then, the optimal P&R method is applied to solve these sub-problems (which are now significantly less complex). The resulting sub-designs can again be seen as components which, applying the optimal P&R method again, are arranged in an optimal fashion. This way, several local optima are combined in the best possible fashion.

In the remainder of this work, the concepts introduced above are described in detail. Afterwards, experimental evaluations demonstrate how the resulting methodology indeed satisfies the promise of an efficient generation of optimal or, at least, close-to-optimal placements and routings for microfluidic biochips.

IV. Optimal Placement & Routing Method

The optimal Placement and Routing method (optimal P&R method) is the "heart" of the proposed methodology and is supposed to determine a placement and a routing realizing the desired experiment with the respectively given constraints. Since an explicit consideration of all possible solutions would be infeasible (as discussed in Section III.A), a symbolic formulation is created instead. This symbolically represents all possible solutions in a much more compact fashion and can be passed to a satisfiability solver in order to determine an explicit solution. In this section, the details on the symbolic formulation and how this eventually realizes the placement and routing is described.

A. Symbolic Formulation of the Grid

To symbolically formulate the placements of the components and the channels on the grid, a formulation of all possible grid cell assignments is created: For each grid cell a *one-hot encoding* for the components is applied, where a single Boolean variable represents whether this cell is occupied by a respective component. Similarly, for each grid cell, a one-hot encoding for the channels is applied, where a single Boolean variable represents whether this cell is occupied by a channel. More formally:

Definition 1 *Consider a grid of size $w \times h$. Each grid cell has a unique position (x,y) with $1 \leq x \leq w$ and $1 \leq y \leq h$. Then,*

- *for each component $v \in V$ to be placed on the grid, a Boolean variable $vp_{(x,y),v}$ is introduced stating whether ($vp_{(x,y),v} = 1$) or not ($vp_{(x,y),v} = 0$) the grid cell at position (x,y) is occupied by the component v and*
- *for each channel between these components $(u,v) \in E$, a Boolean variable $ep_{(x,y),(u,v)}$ is introduced stating whether ($ep_{(x,y),(u,v)} = 1$) or not ($ep_{(x,y),(u,v)} = 0$) the grid cell at position (x,y) is occupied by the channel (u,v).*

Passing the resulting formulation to a satisfiability solver would yield arbitrary assignments to the variables and, hence, arbitrary placements and routings. Hence, as a next step, the possible solutions are restricted so that only valid grid cell assignments (and, hence, placement/routings) are possible. Therefore, constraints are added which ensure that a grid cell can only be occupied

[1] Note that, in a similar fashion, optimality for other constraints such as grid size, channel length, etc. can be guaranteed as well.

- by at most one component, i.e.,

$$\bigwedge_{\substack{1\le x\le w\\ 1\le y\le h}} \left(\sum_{v\in V} vp_{(x,y),v} \le 1 \right), \tag{1}$$

- by at most two channels (if interconnections are allowed), i.e.,

$$\bigwedge_{\substack{1\le x\le w\\ 1\le y\le h}} \left(\sum_{(u,v)\in E} ep_{(x,y),(u,v)} \le 2 \right), \text{ and} \tag{2}$$

- by a component or a channel but not by both simultaneously, i.e.,

$$\bigwedge_{\substack{1\le x\le w\\ 1\le y\le h}} \neg \left(\sum_{v\in V} vp_{(x,y),v} > 0 \wedge \sum_{(u,v)\in E} ep_{(x,y),(u,v)} > 0 \right). \tag{3}$$

B. Enforcing the Placement

Next, constraints are added, which enforce that all components $v \in V$ are placed on the grid. Therefore, we introduce new Boolean variables representing the positions of the components. A position of a component can uniquely be determined by the position of the north west corner. More formally:

Definition 2 *Consider the set of components V which have to be placed on the grid. Then, new Boolean variables $plm_{(x,y),v}$ are introduced stating whether ($plm_{(x,y),v} = 1$) or not ($plm_{(x,y),v} = 0$) the north west corner of a component $v \in V$ is placed at the grid cell at position (x,y).*

Next, a constraint is added which enforces that each component is placed exactly once, i.e.,

$$\bigwedge_{v\in V} \left(\sum_{\substack{1\le x\le w-w_v+1\\ 1\le y\le h-h_v+1}} plm_{(x,y),v} = 1 \right). \tag{4}$$

The variables w_v and h_v specify the size of the component v in terms of grid cells. Since this placement only determines the grid cell with the component's north west corner, another constraint is added which, depending on the size of the respective component, also occupies the corresponding grid cells, i.e.,

$$\bigwedge_{v\in V} \bigwedge_{\substack{1\le x\le w\\ 1\le y\le h}} plm_{(x,y),v} \Longrightarrow \bigwedge_{\substack{x\le x_n<x+w_v\\ y\le y_n<y+h_v}} vp_{(x_n,y_n),v}. \tag{5}$$

More precisely, if a $plm_{(x,y),v}$-variable is set to 1 for a position (x,y) and a component v, it is enforced that the $vp_{(x,y),v}$-variables of the grid cells occupied by the component v are set to 1 as well.

Passing the resulting formulation to a solving engine yields a valid placement of the components. Note that Eq. 1 ensures that no component overlaps with another component. If the specified grid is too small to place all components, the solving engine will determine the unsatisfiability of the instance and, by this, will prove the non-existence of a valid solution.

C. Enforcing the Routing

After adding constraints enforcing a valid placement of the components, it is left to connect the components with channels according to E. A channel $(u,v) \in E$ connects the output port of the source component u with the input port of the destination component v. Therefore, the grid has to contain a continuous path of grid cells connecting the output port with the input port (diagonal channels are not allowed).

The routing of the channels is accomplished in two steps: A channel starts at a grid cell next to an output port and ends at a grid cell next to an input port. Therefore, in the first step, we add constraints fixing the channel next to the output and input ports. In the second step, we add constraints ensuring that, for a grid cell which is occupied by a channel, its neighboring grid cells continue to be a channel – eventually realizing the start-to-end connection.

More precisely, for fixing a channel $(u,v) \in E$ to the output and input ports of the respective components, we employ the $plm_{(x,y),u}$- and $plm_{(x,y),v}$-variables. They provide information where the respective source and destination component have been placed. Additionally, considering all possible port positions of both components (which are provided by the component's specification and, for sake of clarity, are abstracted by the functions Out and In), the start- and end-cell of the grid can be implied, i.e.,

$$\bigwedge_{(u,v)\in E} \bigwedge_{\substack{1\le x\le w\\ 1\le y\le h}} plm_{(x,y),v} \Longrightarrow \bigvee_{\substack{(x_{in},y_{in})\in\\ \mathrm{In}(x,y,v)}} ep_{(x_{in},y_{in}),(u,v)} \tag{6}$$

$$\bigwedge_{(u,v)\in E} \bigwedge_{\substack{1\le x\le w\\ 1\le y\le h}} plm_{(x,y),u} \Longrightarrow \bigvee_{\substack{(x_{out},y_{out})\in\\ \mathrm{Out}(x,y,u)}} ep_{(x_{out},y_{out}),(u,v)}. \tag{7}$$

Based on these fixed channel-cells, the corresponding start-to-end connections can be realized: To this end, we enforce that a cell (x,y) which realizes a channel (u,v) always has corresponding neighboring cells $(x_n,y_n) \in NB(x,y)$ (with $NB(x,y)$ defining a set of all neighbors of grid cell (x,y)) realizing the same channel as well (and, hence, form a path through several grid cells). In the general case, a channel-cell requires two neighboring cells of the same channel (one incoming cell and one outgoing cell) – leading to the constraint

$$\begin{aligned} \bigwedge_{(u,v)\in E} \bigwedge_{\substack{1\le x\le w\\ 1\le y\le h}} & ep_{(x,y),(u,v)} \wedge (x,y) \notin \mathrm{In}(x,y,v) \wedge (x,y) \notin \mathrm{Out}(x,y,u) \\ & \Longrightarrow \sum_{(x_n,y_n)\in \mathrm{NB}(x,y)} ep_{(x_n,y_n),(u,v)} = 2. \end{aligned} \tag{8}$$

If a cell is next to a port, a *single* cell realizing the same channel is sufficient (as either the incoming or outgoing path leads to the port). In case, the source and destination components have been placed next to each other the ports alone are already sufficient and *no* neighbors are required. For these special cases, Eq. 8 is adjusted accordingly.

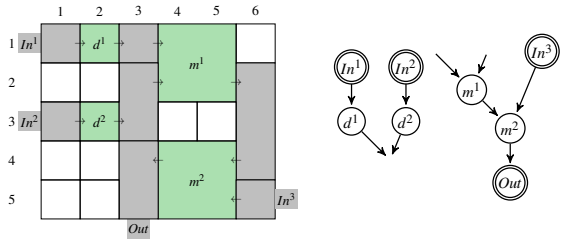
Finally, it is left to add constraints enforcing the routing of the channels in a way that the designer's constraints are satisfied. For example, a maximal allowed amount of channel intersections can be enforced by counting the intersections of all grid cells and restricting the sum to the desired limit (given as *maxIntersections*), i.e.,

$$\sum_{\substack{1\le x\le w\\ 1\le y\le h}} \left(\sum_{(u,v)\in E} ep_{(x,y),(u,v)} = 2 \right) \le maxIntersections. \tag{9}$$

Similarly, the maximal channel length can be restricted by adding the number of all grid cells realizing a channel and restricting this number to the desired limit (given as *maxLength*), i.e.,

$$\sum_{\substack{1\le x\le w\\ 1\le y\le h}} \left(\sum_{(u,v)\in E} ep_{(x,y),(u,v)} \right) \le maxLength. \tag{10}$$

Passing the formulation to a solving engine now either gives an assignment of the variables representing a valid design or proves that the given experiment cannot be realized on the given grid size, with the given maximal number of intersections, and with the given maximal channel length.



(a) Down scaled grid (b) Split experiment

Fig. 3. Pruning steps

V. Search Space Pruning

As discussed in Section III, the optimal P&R method has to tackle a significant complexity and, hence, might only be applicable for small experiments to be realized. Hence, we additionally propose two methods which can be used to prune the search space and, by this, reduce the complexity. The pruning shall thereby be conducted in a fashion which does not discard promising solutions. In the following, details are provided.

A. Downscaling

The first method aims for reducing the complexity by reducing the size of the respective components and the overall grid. To this end, we first scale down all components $v \in V$ as well as the considered grid by a constant factor f. The value of f may depend on the originally given component and grid sizes[2]. For example, the design problem discussed before in Section II.A assumed a grid size of 12×10 as well as component sizes 2×2 and 4×4 – hence, a factor $f = 2$ yielding a grid size 6×5 as well as component sizes 1×1 and 2×2 seems obvious[3].

The resulting (downscaled) design problem is significantly easier to solve and, hence, enables the optimal P&R method to determine a result in significantly less time (for larger designs, downscaling may enable the optimal P&R method to determine a result at all). For the downscaled version of the considered example, this yields a placement and routing as shown in Fig. 3a.

Next, the obtained design is scaled up to the originally desired grid and component sizes. This can easily be conducted for the grid itself as well as all its components. While this already yields a valid placement, an upscaling of the determined routing obviously would lead to disproportional dimensions between components and channels. Hence, we apply the optimal P&R method again in order to determine a routing for the obtained placement. As the placement is assumed to be fix, the complexity of this task is feasible and, for the considered example, yields the design as shown in Fig. 2b.

Overall, this results in a design solution which is close to the optimal result. In fact, although an optimal design is only guaranteed for the downscaled version, many structurally properties are preserved when scaling up the solution to the original dimension. This particularly holds for the important issue of intersections: When e.g., a routing with no intersection is determined at the downscaled level, the optimal P&R method will also determine such a routing in the original dimension.

B. Partitioning

The second method aims for reducing complexity by partitioning the overall problem into sub-problems. As the sub-problems are smaller, also the complexity to be tackled by the optimal P&R method is reduced. At the same time, the quality of the resulting design can be expected to be close-to-optimal since, eventually, local optima are combined[4].

Using the running example discussed before in Section II.A, the proposed method works as follows: First, the initial experiment (shown in Fig. 2a) is partitioned into equally large parts as shown in Fig. 3b. Then, the optimal P&R method determines a sub-design for all sub-problems. To this end, additional constraints are added which allow that skipped channels between sub-designs remain routable, i.e., that a channel from the output port located in one sub-design can still be connected to the input port located in another sub-design.

These sub-designs can be seen as components, which have to be placed and routed on the original grid. Since this leads to a problem of moderate complexity, also this final step can again be conducted by the optimal P&R method.

Overall, partitioning provides a further trade-off between quality and run-time. While quality is obviously harmed due to the local consideration of sub-problems, results of rather good quality still can be ensured.

VI. Experimental Evaluation

We implemented the proposed methodology in Java and used Boolector [23] in its latest version as solving engine for the optimal P&R method. Afterwards, we evaluated the resulting approach using three different configurations, namely

- a sole application of the optimal P&R method as proposed in Section IV (denoted by *Optimal P&R Method*),
- the additional application of the downscaling pruning method as proposed in Section V.A (denoted by *Downscaling*; a scaling factor of $f = 5$ is used), and
- the additional application of the partitioning pruning method as proposed in Section V.B (denoted by *Downscaling + Partitioning*; 4 partitions are used).

These configurations allow for an evaluation of the flexibility of the proposed methodology with respect to optimality and efficiency. Besides that, we compared the obtained results to solutions generated by

- the current state-of-the-art method as proposed in [12] (denoted by *Heuristic presented in [12]*).

As benchmarks, we considered all experiments which have been used before in [12]. Besides that, in order to evaluate the scalability of the proposed approach, we additionally applied another series of experiments in which we assumed smaller component specifications (denoted by the suffix *Small* in the benchmark name). All experiments using the proposed methodology have been conducted on a desktop machine with a 3.2GHz Core i5 and 8GB of main memory running 64-bit Ubuntu 14.04 LTS. The evaluations of the heuristic presented in [12] have been conducted on a 2.6GHz 32-core Intel Xeon Linux workstation with 132GB main memory.

Table I reports the obtained results. The first columns provide the name of the experiment to be realized as well as its number of components ($|V|$) and channels ($|E|$) to be placed. Afterwards, the size of the grid (*Area*), the number of channel intersections (*Int.*), the channel length (*Length*), and the run-time in CPU-seconds (*T[s]*) are provided for all configurations as well as the method proposed in [12]. Benchmarks for which no result could be determined within 4 CPU hours have been aborted (denoted by "—").

[2] In case this leads to fractional numbers for components and/or grid sizes, the resulting values are rounded up.

[3] Also bigger scaling factors can be used (the maximal possible scaling factor would yield 1×1 components).

[4] This has also been confirmed by experimental evaluations which are summarized later in Section VI.

TABLE I
EVALUATION RESULTS

Benchmark			Heuristic presented in [12]				Optimal P&R Method				Downscaling				Downscaling + Partitioning			
Name	$\|V\|$	$\|E\|$	Area	Int.	Length	T [s]	Area	Int.	Length	T [s]	Area	Int.	Length	T [s]	Area	Int.	Length	T [s]
InVitro-1 Small	30	24	84×123	8	1368	80	9×9	0	24	253	9×9	0	24	253	15×15	0	36	3
InVitro-2 Small	45	36	126×123	22	2546	172	11×11	0	36	5474	11×11	0	36	5474	20×20	0	74	9
InVitro-3 Small	60	48	158×156	68	4670	310	13×13	0	62	7456	13×13	0	62	7456	14×14	0	48	10
PCR Small	16	15	63×41	0	609	34	9×9	0	19	801	9×9	0	19	801	23×23	4	97	6
ProteinSplit-1 Small	30	27	78×90	17	1405	117	11×11	0	60	11935	11×11	0	60	11935	20×20	1	72	13
ProteinSplit-2 Small	66	60	58×72	331	2324	379	—	—	—	—	—	—	—	—	27×27	2	145	6005
InVitro-1	30	24	70×73	1	802	84	—	—	—	—	45×45	0	144	253	63×63	0	212	3
InVitro-2	45	36	84×98	8	1485	180	—	—	—	—	55×55	0	228	5474	76×76	0	338	9
InVitro-3	60	48	99×113	6	1846	301	—	—	—	—	65×65	0	314	7457	70×70	0	288	10
PCR	16	15	51×58	1	522	43	—	—	—	—	45×45	0	103	801	79×79	4	389	28
ProteinSplit-1	30	27	63×78	5	1162	114	—	—	—	—	55×55	0	272	11938	76×76	1	328	14
ProteinSplit-2	66	60	131×130	42	3247	528	—	—	—	—	—	—	—	—	99×99	2	625	6018

Name: the name of the benchmark $|V|$: number of components $|E|$: number of channels Area: the size of the grid
Int.: the number of channel intersections Length: the channel length T[s]: run-time in CPU-seconds

The results clearly show the flexibility and strengths of the proposed methodology. For the first time, *optimal* placements and routings can be obtained for continuous-flow microfluidic biochips – although for rather small benchmarks only. However, this lack of scalability can easily be addressed by the proposed pruning methods. While applying *Downscaling* alone already allows for completing the tasks for all benchmarks except one[5], additionally employing *Partitioning* allows to realize *all* experiments in acceptable run-time.

The quality of the obtained results is thereby hardly harmed by the pruning methods. In fact, despite the downscaling, an optimal placement/routing with respect to the number of intersections (as discussed in Section I, the most important criteria) can still be derived. If partitioning is additionally applied, only slight increases for the grid size (avg. increase of 40%), intersections (between 0 and only 4 new intersections), and channel length (avg. increase of 77%) are observed. Hence, close-to-optimal results can be achieved.

Overall, compared to the current state-of-the-art method proposed in [12], substantial improvements are reported for all design criteria. Often differences with 1-2 orders of magnitude are reported. In particular, the improvement with respect to the number of intersection is of importance: While the previous solution generated placements/routings requiring dozens or even hundreds of intersections, the proposed methodology can reduce them to just a few or, oftentimes, none.

VII. CONCLUSIONS

In this work, we presented a placement and routing methodology for continuous-flow microfluidic biochips, which is capable of generating optimal or close-to-optimal results. To this end, we considered placement and routing together and aimed for covering the entire search space. In order to cope with the resulting complexity, efficient solving engines and pruning schemes are utilized. Experimental evaluations showed that the resulting methodology satisfies the promise of being capable to generate close-to-optimal designs. By this, results generated by the current state-of-the-art method could be improved by up to 1-2 orders of magnitude.

REFERENCES

[1] T. Thorsen, S. J. Maerkl, and S. R. Quake. Microfluidic large-scale integration. *Science*, 298(5593):580–584, 2002.

[2] G. M. Whitesides. The origins and the future of microfluidics. *Nature*, 442(7101):368–373, 2006.

[3] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle. Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications. *Chemical Society Reviews*, 39(3):1153–1182, 2010.

[4] J. W. Hong and S. R. Quake. Integrated nanoliter systems. *Nature Biotechnology*, 21(10):1179–1183, 2003.

[5] M. J. Anderson, C. L. Hansen, and S. R. Quake. Phase knowledge enables rational screens for protein crystallization. *National Academy of Sciences*, 103(45):16746–16751, 2006.

[6] E. P. Kartalov, J. F. Zhong, A. Scherer, S. R. Quake, C. R. Taylor, and W. F. Anderson. High-throughput multi-antigen microfluidic fluorescence immunoassays. *BioTechniques*, 40(1):85, 2006.

[7] Y. Huang, P. Castrataro, C.-C. Lee, and S. R. Quake. Solvent resistant microfluidic DNA synthesizer. *Lab on a Chip*, 7(1):24–26, 2006.

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

[9] J. A. Rogers and R. G. Nuzzo. Recent progress in soft lithography. *Materials Today*, 8(2):50–56, 2005.

[10] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science*, 288(5463):113–116, 2000.

[11] T. M. Squires and S. R. Quake. Microfluidics: Fluid physics at the nanoliter scale. *Reviews of Modern Physics*, 77(3):977, 2005.

[12] Q. Wang, Y. Ru, H. Yao, T.-Y. Ho, and Y. Cai. Sequence-pair-based placement and routing for flow-based microfluidic biochips. In *Asia and South Pacific Design Automation Conference*, pages 587–592, 2016.

[13] W. H. Minhass, P. Pop, J. Madsen, and F. S. Blaga. Architectural synthesis of flow-based microfluidic large-scale integration biochips. In *Int'l Conf. on Compilers, Architectures and Synthesis for Embedded Systems*, pages 181–190, 2012.

[14] K.-H. Tseng, S.-C. You, J.-Y. Liou, and T.-Y. Ho. A top-down synthesis methodology for flow-based microfluidic biochips considering valve-switching minimization. In *Int'l Symp. on Physical Design*, pages 123–129, 2013.

[15] C.-X. Lin, C.-H. Liu, I-C. Chen, D. Lee, and T.-Y. Ho. An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips. In *Design Automation Conference*, pages 1–6, 2014.

[16] J. McDaniel, B. Parker, and P. Brisk. Simulated annealing-based placement for microfluidic large scale integration (mLSI) chips. In *Int'l. Conf. on Very Large Scale Integration of System-on-Chip*, pages 1–6, 2014.

[17] W. H. Minhass, P. Pop, and J. Madsen. System-level modeling and synthesis of flow-based microfluidic biochips. In *Int'l Conf. on Compilers, Architectures and Synthesis for Embedded Systems*, pages 225–233, 2011.

[18] W. H. Minhass, P. Pop, and J. Madsen. Synthesis of biochemical applications on flow-based microfluidic biochips using constraint programming. In *Symp. on Design, Test, Integration and Packaging of MEMS/MOEMS*, pages 37–41, 2012.

[19] N. Amin, W. Thies, and S. Amarasinghe. Computer-aided design for microfluidic chips based on multilayer soft lithography. In *Int'l Conf. on Computer Design*, pages 2–9, 2009.

[20] W. H. Minhass, P. Pop, J. Madsen, and T.-Y. Ho. Control synthesis for the flow-based microfluidic large-scale integration biochips. In *Asia and South Pacific Design Automation Conference*, pages 205–212, 2013.

[21] H. Yao, Q. Wang, Y. Ru, Y. Cai, and T.-Y. Ho. Integrated flow-control codesign methodology for flow-based microfluidic biochips. *Design & Test*, 32(6):60–68, 2015.

[22] N. Eén and N. Sörensson. An extensible SAT solver. In *SAT 2003*, volume 2919 of *LNCS*, pages 502–518, 2004.

[23] R. Brummayer and A. Biere. Boolector: An Efficient SMT Solver for Bit-Vectors and Arrays. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 174–177, March 2009.

[24] O. Keszocze, R. Wille, T.-Y. Ho, and R. Drechsler. Exact one-pass synthesis of digital microfluidic biochips. In *Design Automation Conference*, pages 1–6, 2014.

[25] O. Keszocze, R. Wille, K. Chakrabarty, and R. Drechsler. A general and exact routing methodology for digital microfluidic biochips. In *Int'l Conf. on Computer-Aided Design*, pages 874–881, 2015.

[5]Note that downscaling the "Small"-benchmarks does not reduce the search space further and, therefore, Table I reports the same results for "Optimal P&R Method" and "Downscaling".