Exact One-pass Synthesis of Digital Microfluidic Biochips

Oliver Keszocze[1,2] Robert Wille[1,2] Tsung-Yi Ho[3] Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
[3]Dept. of CSIE, National Cheng Kung University, Tainan, Taiwan
{keszocze,rwille,drechsle}@informatik.uni-bremen.de tyho@csie.ncku.edu.tw

ABSTRACT

With the advances of the microfluidic technology, the design of digital microfluidic biochips recently received significant attention. But thus far, the corresponding design tasks such as binding, scheduling, placement, and routing have usually been considered separately. Furthermore, often just heuristic results have been obtained. In this work, we present a one-pass synthesis scheme which directly realizes the desired functionality onto the chip and, at the same time, guarantees minimality with respect to area and/or timing. For this purpose, the deductive power of solvers for Boolean satisfiability is exploited. Experiments show how the approach leverages the design of the respective devices.

1. INTRODUCTION

Advances in droplet-based *Digital Microfluidic Biochips* (DMFBs) have led to the emergence of biochips for automating laboratory procedures in biochemistry and molecular biology. Biochemical assays, such as the dilution of samples and reagents, crystallization of protein molecules, on-chip chemistry for DNA sequencing, multiplexed real-time *Polymerase Chain Reaction* (PCR), protein crystallization for drug discovery, and glucose measurement for blood serum have successfully been implemented on such biochips [9].

In general, a DMFB consists of a two-dimensional electrode grid and peripheral devices such as optical detectors and dispensing ports (as schematically shown in Figure 1 [9]). The sample carriers, so called *droplets*, are miniaturized and discretized liquids which are controlled by underlying electrodes using electrical actuations (i. e. a principle called *electrowetting-on-dielectric* (EWOD) [11]). By assigning time-varying voltage values to turn electrodes on and off, droplets can be moved around the entire grid to perform fundamental operations such as dispensing and mixing [13]. These operations are carried out in a reconfigurable manner. Therefore, DMFBs offer various advantages including a more flexible control mechanism, a higher throughput and sensitivity, as well as a lower sample/reagent volume consumption.

Due to these advantages, DMFBs have attracted significant attention being devoted to the need in marketplace from healthcare, environmental, and point-of-care-testing applications. According to a report released by Research and Markets in June 2013, the global biochip market will grow from 1.4 billon in 2013 to 5.7 billion by 2018 [1]. Driven by

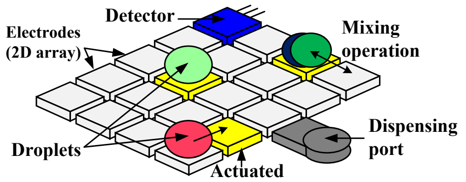


Figure 1: The schematic view of a DMFB

this growth, also the respective applications and, by this, the design of corresponding DMFBs have become more complex. As a consequence, a *manual* design of these devices is not suitable anymore – particularly under the current time-to-market constraints. Instead, high quality design and synthesis methodologies are required which relieve the design burden of manual optimizations of bioassays, time-consuming chip designs, as well as costly testing and maintenance procedures.

Consequently, researchers have developed a design flow for DMFBs composed of several steps such as binding, scheduling, placement, and routing (this is reviewed in more detail in Section 2.1). For each of these steps, a number of corresponding design methods have been separately developed [4, 7, 10, 15, 12]. However, design gaps among these four stages do restrict the effectiveness and feasibility of the entire DMFB realization, which reveals a demand for design convergence. These design gap problems will become even more critical with a rapid escalation in the number of assay operations incorporated into a single large-scale DMFB. Furthermore, most of the previous methods are based on heuristic strategies that sacrifice the optimality requirement of the problem. To the best of our knowledge, there is no work in the literature that provides an integrated solution to deal with all these concerns.

In this paper, we present a one-pass synthesis scheme which directly realizes the desired functionality onto the chip and, at the same time, guarantees minimality with respect to the grid-area and/or the overall completion time. In order to tackle this computationally hard problem, we exploit the deductive power of solvers for Boolean satisfiability. Experimental results confirm that, despite the complexity, *minimal* results for a given set of benchmarks can be generated. While these minimal realizations are beneficial on its own, they also enable more sophisticated studies, e. g. on the relation between the minimal grid-size and the minimal computation time of the considered devices.

In the remainder of this paper, the proposed contributions as well as results are presented as follows: Section 2 provides a brief review on the background needed to make this work self-contained. Afterwards, Section 3 motivates the envisioned one-pass synthesis and provides the general idea of our solution (namely exploiting Boolean satisfiability). Based on that, a detailed description of the proposed SAT formulation is provided in Section 4. Finally, a summary of the obtained experimental results is provided in Section 5, while conclusions are drawn in Section 6.

DAC '14, June 01 - 05 2014, San Francisco, CA, USA
Copyright 2014 ACM 978-1-4503-2730-5/14/06...$15.00.
http://dx.doi.org/10.1145/2593069.2593135

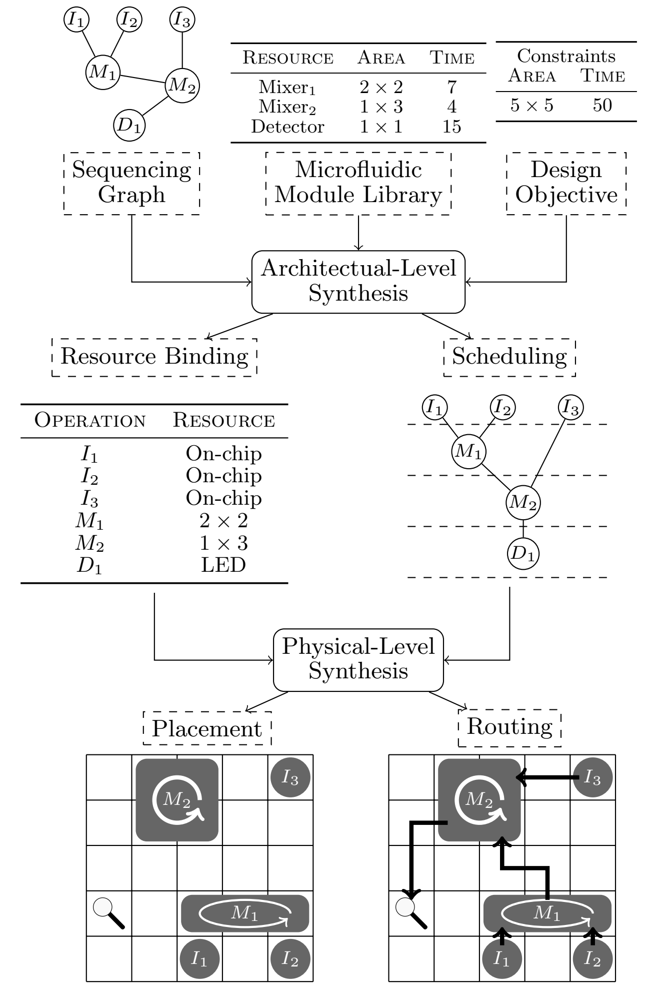


Figure 2: Conventional synthesis flow of DMFBs

2. BACKGROUND

To keep this work self-contained, this section briefly reviews the basics of DMFB design as well as Boolean satisfiability.

2.1 Design of Digital Microfluidic Biochips

Thus far, the design of DMFBs is conducted by means of a two-stage design flow composed of an *architecture-level synthesis* and a *physical-level synthesis* as illustrated in Figure 2. The input of this design flow is (a) a *sequencing graph* which specifies the desired functionality to be implemented, (b) a *module library* providing realizations such as mixing, detecting, etc. as well as their respectively needed grid-size and timing requirements, and (c) a design objective, i. e. the available maximal grid-size and the desired maximal completion time. The objective is to realize the given functionality onto the given grid and within the maximal completion time utilizing the available operations from the module library.

During architecture-level synthesis, *binding* will map the allocated modules from the module library to the biochemical operations in the application. *Scheduling* determines the order of operations based on the binding result. Then, in physical-level synthesis, *placement* determines the actual locations of the different operations corresponding to different time intervals. Finally, *routing* schedules the movement of each droplet in a time-multiplexed manner.

As a result, a cascade of grid-configurations is obtained which specify the placement of all entities such as droplets, dispensers, detectors, and sinks as well as the precise settings of operations such as mixing for each time step t with $1 \leq t \leq T$ and T being the maximal completion time.

Here, a droplet always is of a certain *type* l (e. g. blood or urine) and may only appear onto the grid if it is generated either by a dispenser of the same type or by a corresponding mixing operation. Vice versa, a droplet can only disappear from the grid if it is mixed with another droplet or dumped by using a sink. Both, the number of available dispensers and the number of available sinks may be limited by the designer (respective limits are denoted by $n_{dispensers,l}$ and n_{sinks}). Operations can arbitrarily be placed onto the grid as long as the respective cells do not overlap. Finally, in order to keep the remainder of this work simple, in the following we assume mixing and detecting operations only. Further operations can be incorporated in a similar fashion.

2.2 Boolean Satisfiability

This work exploits solving engines for Boolean satisfiability in order to tackle the considered design problem. The *Boolean Satisfiability* (SAT) problem is defined as follows: Let f be a Boolean function. Then the SAT problem is to determine an assignment for the variables of f so that f evaluates to true or to prove that no such assignment exists.

Example 1. *Let $f = (x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_3)(\overline{x}_2 + x_3)$. Then, $x_1 = 1, x_2 = 1$ and $x_3 = 1$ is a satisfying assignment for f. The values of x_1 and x_2 ensure that the first clause becomes satisfied while x_3 ensures this for the remaining two clauses.*

In the past efficient solving algorithms (so called *SAT solvers*) have been proposed (see e.g. [6]) allowing for coping with problem instances composed of hundreds of thousands of variables.

3. MOTIVATION & GENERAL IDEA

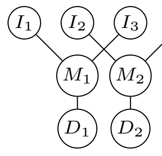
In this section, first the concept of one-pass synthesis is introduced and motivated. Afterwards, the general idea of the proposed solution is sketched. This provides the basis for the detailed description which follows in Section 4.

3.1 Exact One-pass Synthesis

As reviewed in Section 2.1, the design of digital microfluidic biochips is usually split up into several – more or less independent – phases such as binding, scheduling, placement, and routing. For most of these steps several – heuristic and exact – approaches have been proposed which determine intermediate results (see e. g. [12] for scheduling or [15] for place and route). As a consequence, the interaction of the different algorithms is hardly considered. This led to a quality loss or even impractical results, e. g. where highly optimized schedulings do not or just badly fit onto the anticipated target architecture.

Example 2. *Consider the sequencing graph shown in Figure 3(a). The design objective is to realize the given functionality (i. e. mixing four fluids and, afterwards, performing two detecting operations) onto a 3×3 grid with mixers of size 2×2 in the minimal completion time. Solely performing scheduling would lead to an intermediate result assuming that M_1 and M_2 can be scheduled in parallel. This is because the combined size of the mixers is 8 and the total size of the grid is 9. However, the succeeding placement step unveils that such a schedule leads to a* placement failure. *It is not possible to have two mixers on the grid in the same time step (see Fig. 3(b)).*

Placement failures (as well as routing failures) are a major obstacle for the existing design flow. Moreover, the eventually resulting designs are often far from being optimal – even if the intermediate results have been generated by exact



(a) Sequencing graph

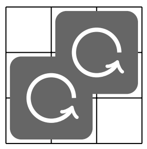

(b) Placement failure

Figure 3: Motivating one-pass synthesis

approaches and, hence, might be optimal individually. Although ensuring optimality is usually computationally expensive, the corresponding exact synthesis approaches are of great interest as (1) they allow determining smaller realizations than the currently best known, (2) they allow for the evaluation of the quality of heuristic approaches, and (3) they allow for the computation of minimal realizations to be used as basic blocks for larger functionality.

Motivated by this, we propose an alternative design method for microfluidic biochips which explicitly addresses these shortcomings. More precisely, a methodology is presented which

- employs a one-pass synthesis scheme and considers the important design steps at once (i. e. for a given sequencing graph a suitable placing and routing is determined directly) and, at the same time,
- generates an exact solution in the sense that the obtained result is optimal with respect to area or completion time.

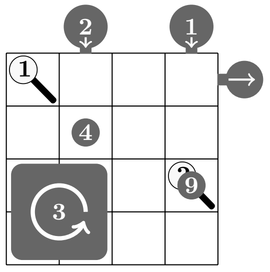
3.2 General Idea

Performing a one-pass synthesis and, at the same time, guaranteeing minimality is a computationally hard problem. For this purpose, all possible placements of entities such as dispensers, detectors, and sinks as well as the precise settings of operations such as mixing need to be considered. The same holds for all possible routing paths of the available droplets over the entire time span. All these combinations easily multiply together to a seriously large search space to be explored.

To cope with this complexity, we propose to exploit the deductive power of solvers for *Boolean satisfiability* (SAT). Their intelligent decision heuristics, powerful learning schemes, and fast implication methods (see e. g. [6]) enable to efficiently traverse large search spaces and have been proven to be very effective for many practically relevant CAD problems such as formal verification and automatic test pattern generation [3]. As shown in this work, this deductive power can also be utilized in order to determine a valid solution from all the possible combinations discussed above. Minimality with respect to the area or the completion time is then ensured by formulating the optimization problem considered here as a sequence of decision problems.

More precisely, the general idea of the proposed approach is as follows: For a given sequencing graph, a grid of size $w \times h$, and a completion time T, the decision problem "Is there a valid placement and routing onto a $w \times h$ grid in T time steps realizing the functionality of the sequencing graph?" is formulated as a SAT instance and passed to the corresponding solving engine. If there is no positive answer, i. e. if the decision problem is unsatisfiable, the grid size and/or the completion time is incremented and the process is repeated. This iteration is stopped when one of the resulting decision problems is satisfiable. Then, a valid placement and routing can be derived from the corresponding solution of the SAT instance. By initially setting the area and/or the completion time to 1 (or any available lower bound) and by iteratively increasing them, minimality is ensured[1].

[1]Note that instead of an incremental scheme also similar approximations such as a binary search can be applied to determine the minimum. This is discussed in more detail in Section 5.



(a) Placement

Droplet (with identifier 4): $c^t_{2,2,4} = 1$
Droplet (with identifier 9): $c^t_{4,3,9} = 1$
Mixer (with identifier 3):
$c^t_{1,4,3} = c^t_{2,4,3} = c^t_{1,3,3} = c^t_{2,3,3} = 1$
Sink: $sink_5 = 1$
Dispenser (for type 1): $dispenser_{4,1} = 1$
Dispenser (for type 2): $dispenser_{2,2} = 1$
Detector (for type 1): $detector_{1,1,1} = 1$
Detector (for type 2): $detector_{4,3,2} = 1$
In this time step t, droplet 9 is detected, i.e. $detecting^t_9 = 1$

(b) Variables

Figure 4: Encoding of a possible grid-placement

Following this scheme, a minimal solution can be determined much faster than by solely enumerating all possible combinations. In contrast, this scheme requires the respective decision problem to be formulated as a SAT instance which can be processed by the mentioned solving engines. How this SAT instance formulated is described in the next section.

4. PROPOSED SAT ENCODING

This section describes the proposed SAT encoding. For a given sequencing graph, a corresponding SAT instance is formulated which is satisfiable if there exists a valid placement and routing realizing the functionality specified by the sequencing graph onto a grid of size $w \times h$ within T time steps. This is encoded by means of constraints over variables representing the position of all droplets, operations, and entities for all time steps. The SAT instance is composed of the following variables:

Definition 1. *Consider an $n \times n$-grid for which a placement and routing over T time steps shall be derived*[2].

- *The variables $c^t_{x,y,i}$ with $1 \leq x, y \leq n$ and $1 \leq t \leq T$ represent whether ($c^t_{x,y,i} = 1$) or not ($c^t_{x,y,i} = 0$) the (x, y)-cell of the grid is occupied by a droplet or mixer with the unique identifier i at time step t.*
- *The variables $sink_p$ and $dispenser_{p,l}$ with $1 \leq p \leq 4n$ represent whether or not a sink or a dispenser (dispensing a droplet of a given type l) is assumed at position p. As sinks and dispensers are positioned outside of the grid, p is defined by a clockwise enumeration of all possible positions starting above the $(1, 1)$-cell.*
- *The variables $detector_{x,y,l}$ with $1 \leq x, y \leq n$ represent whether or not a detector for droplets of given type l is placed below the (x, y)-cell of the grid.*
- *The variables $detecting^t_i$ with $1 \leq t \leq T$ represent whether or not a droplet with the identifier i is being detected in time step t.*

Example 3. *Consider the 4×4-grid shown in Figure 4(a) and representing a possible placement of droplets, operations, and entities for a time step t. This placement is represented by setting all variables listed in Figure 4(b) to 1 while all remaining variables are set to 0.*

Having all these variables, it is up to the solving engine to assign values for each time step t. By doing so, the SAT solver creates a placement and, considering all time steps $1 \leq t \leq T$ together, a routing for the given grid and its droplets, operations, and entities. For this purpose, it obviously has to be ensured that (1) only assignments representing valid placements/routings are chosen and (2) the

[2]For brevity, in the remainder of this paper, an $n \times n$-grid is assumed. However, the proposed encoding works on arbitrary $w \times h$-grids.

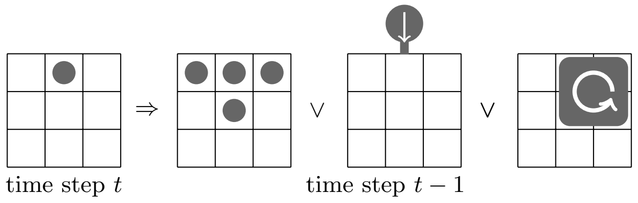


Figure 5: Illustration of the movement constraint

resulting placements/routings realize the given functionality specified in the sequencing graph. This is achieved by applying constraints to these variables as described next.

Encoding Consistency, Placement, and Movement

First, it is constrained that all droplets and operations may not be placed arbitrarily onto the grid, but obey certain consistency properties. This is accomplished by the following *consistency*-constraints.

- A cell may not be occupied by more than one droplet or mixer i per time step, i.e. $\bigwedge_{\substack{1\le t\le T\\1\le x,y\le n}} \left(\sum_i c^t_{x,y,i} \le 1\right)$,
- each droplet i may occur in at most one cell per time step, i. e. $\bigwedge_{1\le t\le T}\left(\sum_{1\le x,y\le n} c^t_{x,y,i}\le 1\right)$,
- in each position p outside of the grid, there may be at most one dispenser (this applies for all types l) or sink, i. e. $\left(sink_p + \left(\sum_l dispenser_{p,l}\right)\right) \le 1$, and
- each cell may be occupied by at most one detector, i. e. $\bigwedge_{1\le x,y\le n}\left(\sum_l detector_{x,y,l}\le 1\right)$.

All non-reconfigurable parts such as sinks, dispensers, and detectors also have to be placed properly. This is accomplished by the following *placement*-constraints:

- For detectors, we ensure that, over all possible (x,y)-cells, for every type l of fluids a detector is placed, i. e. $\bigwedge_l \left(\sum_{1\le x,y\le n} detector_{x,y,l} = 1\right)$.
- For dispensers and sinks, we proceed analogously: For every possible outside position p of the grid and every type of fluid l, we ensure that the desired amount of entities ($n_{dispensers,l}$ and n_{sinks}) is placed, i. e.

$$\bigwedge_l\left(\sum_p dispenser_{p,i} = n_{dispensers,l}\right)\wedge\left(\sum_p sink_p = n_{sinks}\right).$$

The occurrences of the droplets on the grid (and, therefore, their movement) is encoded by a *movement*-constraint. This states that, if a droplet i is present at a certain cell (x,y) at a time step t (i. e. if $c^t_{x,y,i}=1$), it has to be ensured that either

a) in the previous time step $t-1$ the droplet was already present at the same cell or one of the neighboring cells (denoted by N),
b) the droplet is next to a dispenser which is creating this droplet in the current time step, i. e. a dispenser for the corresponding type l is at a position p next to (x,y), or
c) the droplet is the result of a mixing operation, i. e. in the previous time step, the cell (x,y) was occupied by the corresponding mixer m and the droplet i was not present in the neighborhood N.

Altogether, the constraint

$$c^t_{x,y,i}\Rightarrow \underbrace{\left(\bigvee_{(x',y')\in N} c^{t-1}_{x',y',i}\right)}_{\text{a)}} \vee \underbrace{\left(\bigvee_p dispenser_{p,l}\right)}_{\text{b)}}$$
$$\vee\underbrace{\left(c^{t-1}_{x,y,m}\wedge\neg\left(\bigvee_{(x',y')\in N} c^{t-1}_{x',y',i}\right)\right)}_{\text{c)}}$$

has to be satisfied. An illustration of this constraint is provided in Figure 5.

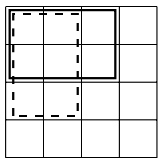


Figure 6: Placements for a 2×3 mixing operation

Note that this constraint implicitly encodes dispensing operations, i. e. no explicit encodings are required for them. Furthermore, all droplets available in the first time step may only origin from dispensers. Hence, only part b) of the movement constraint is applied for this time step.

Encoding Operations

In a second series of constraints, the correct execution of the actual operations, i. e. mixing, detecting, and sinking of droplets, is encoded. We start with the consideration of the mixing operation which requires the most complex constraints.

A mixing operation m takes two droplets $i_{\text{in},1}, i_{\text{in},2}$ and, after a given duration d of time steps, generates a new droplet i. For this purpose, a fixed number of cells (either a $w\times h$-subgrid or a $w\times w$-subgrid) is occupied during this time. Depending on the cell where the new droplet appears (i. e. the "output"-cell of the mixing operation), several options on the precise placement of these subgrids exist. Those are summarized in the set $M(x,y,m)$ with (x,y) being the "output"-cell.

Example 4. *Consider a* 2×3 *mixing operation* m *and the "output"-cell* $(1,1)$. *As illustrated in Figure 6, there are two subgrids (i. e. sets of cells)* g_1 *and* g_2 *that could be occupied by the mixing operation:* $g_1=\{(1,1),(1,2),(2,1),(2,2),(3,1),(3,2)\}$ *and* $g_2=\{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3)\}$. *Hence,* $M(1,1,m)=\{g_1,g_2\}$.

The choice of the subgrid is implicitly encoded, i. e. if the new droplet i appears on cell (x,y) at a time step t (with $t\ge d+2$), then it is only ensured that

a) the two "input"-droplets $i_{\text{in},1}, i_{\text{in},2}$ were present in the neighborhood N of (x,y) right before the mixing operation started at time step $t-(d+1)$,
b) the two "input"-droplets $i_{\text{in},1}, i_{\text{in},2}$ disappeared right after the start of the mixing operation at time step $t-d$, and
c) one of the possible subgrids stored in $M(x,y,m)$ becomes occupied for the total duration of the mixing operation.

Combining the points above gives the following constraint, for which an illustration is provided in Figure 7[3].

$$\bigwedge_{d+2\le t\le T}\bigwedge_{1\le x,y\le n} c^t_{x,y,i}\wedge\left(\sum_{1\le x',y'\le n} c^{t-1}_{x',y',i}=0\right)\Rightarrow$$
$$\bigwedge_{j\in\{i_{\text{in},1},i_{\text{in},2}\}}\left(\underbrace{\sum_{(x',y')\in N} c^{t-(d+1)}_{x',y',j}=1}_{\text{a)}}\wedge\underbrace{\sum_{1\le x',y'\le n} c^{t-d}_{x',y',j}=0}_{\text{b)}}\right)$$
$$\wedge\underbrace{\left(\bigvee_{G\in M(x,y,m)}\left(\bigwedge_{t-d\le t'<t}\bigwedge_{(x',y')\in G} c^{t'}_{x',y',m}\right)\right)}_{\text{c)}}$$

Next, the constraints ensuring the correct execution of detecting operations are provided. For this purpose, it has to

[3]Note that some further consistency constraints for the mixing operation are used, e. g. to enforce that no cells beyond this subgrid are inconsistently set to be occupied by the mixer m. However, those are rather straightforward and do not provide further insights into the proposed encoding. Hence, they have been omitted for reasons of brevity.

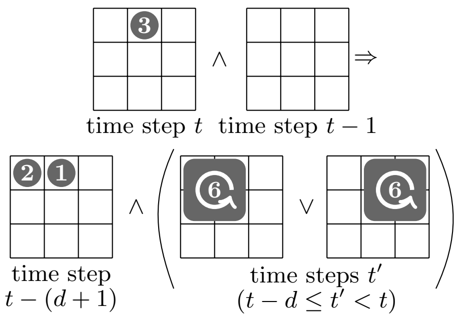


Figure 7: Illustration of the mixing constraint

be ensured that the droplet to be detected and the detector are placed accordingly and that the detection time is considered. More precisely, if there shall be a detecting operation on droplet i (of type l) starting at the time step t (i. e. $detecting_i^t = 1$ and $detecting_i^{t-1} = 0$) and, at the same time, droplet i is on cell (x, y), then

a) a corresponding detector has to be placed at cell (x, y) and
b) for the duration d of the detecting operation, the droplet must not leave the cell and the detection must be continued.

This is encoded by the following constraint.

$$detecting_i^t \wedge \neg detecting_i^{t-1} \wedge c_{x,y,i}^t \Rightarrow$$
$$\underbrace{detector_{x,y,l}}_{\text{a)}} \wedge \underbrace{\bigwedge_{t \leq t' \leq t+d} \left(c_{x,y,i}^{t'} \wedge detecting_i^{t'} \right)}_{\text{b)}}$$

Finally, the disappearance of droplets needs to be encoded. Droplets may disappear either when they leave the grid through a sink or are absorbed by a mixing operation. Hence, if a cell (x, y) was occupied by a droplet i at time step $t-1$, which is not present in the neighborhood N of (x, y) at time step t anymore (a), then there must either be a sink at a reachable position p (b) or a mixing operation m absorbing i was started in the neighborhood N of (x, y) at time step t (c). The corresponding constraint is

$$\underbrace{c_{x,y,i}^{t-1} \wedge \neg \left(\bigvee_{(x',y') \in N} c_{x',y',i}^t \right)}_{\text{a)}} \Rightarrow \underbrace{\left(\bigvee_p sink_p \right)}_{\text{b)}} \vee$$
$$\underbrace{\left(\bigvee_{(x',y') \in N} c_{x',y',m}^t \wedge \neg \left(\bigvee_{(x',y') \in N} c_{x',y',m}^{t-1} \right) \right)}_{\text{c)}}.$$

If the droplet is not to be mixed with another droplet (e. g. it is only used by a detecting operation), the part c) is substituted by 0.

Encoding the Objective and Solving the Problem

All constraints outlined above ensure that the solving engine determines variable assignments representing valid placements and routings. However, additionally it has to be ensured that the actually desired operations (as specified by the sequencing graph) are realized. For this purpose, final constraints enforcing their execution are added:

- It is enforced that all droplets considered in the sequencing graph have to be present for at least one time step. This can easily be accomplished by the constraint $\bigvee_{t=1}^{T} \bigvee_{1 \leq x,y \leq n} c_{x,y,i}^t$. Since this must hold even for droplets generated by mixing, this constraint eventually triggers all desired mixing operations.
- In a similar fashion, all detecting operations are triggered. However, as a single detection may take several time steps, it is enforced that the sum of all $detecting_i^t$ variables is equal to the duration d of the detecting operation, i. e. $\sum_{1 \leq t \leq T} detecting_i^t = d$.

Combining all constraints introduced in this section, a satisfiability instance results which is satisfiable if there exists a valid placement and routing realizing the functionality specified by the sequencing graph. This can efficiently be solved using the solving engines mentioned in Section 3.2. If a satisfiable assignment is returned, the precise placement and routing for the considered problem can be derived from the assignment to the variables defined in Def. 1 as illustrated in Figure 4. Otherwise, it has been proven that the given sequencing graph cannot be realized with the current restrictions on grid size and completion time.

5. EXPERIMENTAL EVALUATION

The proposed approach has been implemented in Ruby which, for a given sequencing graph, a grid size $w \times h$, and a completion time T, generates the satisfiability instance described above using the SMT-LIB2 format [2] extended by a logic for cardinality constraints. The extension has been implemented on top of the open source toolkit *metaSMT* [8]. As solving engine we utilized the SMT solver *Z3* [5].

Afterwards, the resulting approach has been evaluated on a set of multiplexed in-vitro diagnostic benchmarks taken from [14] in different configurations w.r.t. the number of samples and reagents. As target technology we assumed several grids of different sizes[4]. The operation completion times and areas have been taken from [14]. All experiments have been conducted on an 2.6 GHz Intel Corei5 machine with 8 GB of memory running 64bit Xubuntu 13.10.

This section summarizes and discusses the results of the conducted experiments.

5.1 Determining the Minimum

In a first series of experiments, we evaluated how the desired minimal value of the optimization objective can be determined best. As introduced in Section 3.2, the resulting optimization problem is addressed by solving a sequence of decision problems. Here, options exists on how to approach to the minimum: We proposed an iterative approach, e. g. T is initially set to 1 and iteratively increased until a satisfying solution has been determined. But also alternative schemes (e.g. a binary search additionally exploiting the known problem bounds) can be applied for this purpose. However, our experiments unveil that applying the iterative approach is the most feasible scheme in general.

Table 1 exemplarily shows the results (RES) as well as the run-time (TIME) of the respective steps for the synthesis of the multiplexed in-vitro diagnostic benchmark with 2 samples and 3 reagents on a 3×6-grid. The minimal completion time for this configuration is 17 time steps. Thus, using the iterative approach, 17 checks are performed in total. In contrast, assuming the best case for an alternative approximation (i. e. the minimal depth $T = 17$ is determined at the beginning and additionally one check for $T = 16$ is performed) only two checks are necessary. However, since the runtimes needed for the first checks of the iterative approach are small, the total runtime for both approaches differs only slightly (less than 4%). Similar results have been obtained for other configurations. Hence, even in the best case, alternative schemes for approaching the minimum do not lead to significant improvements. Moreover, since such schemes usually require checks for larger values than the optimum, the run-time might even increase in these cases. For all remaining experiments the iterative approach has been used.

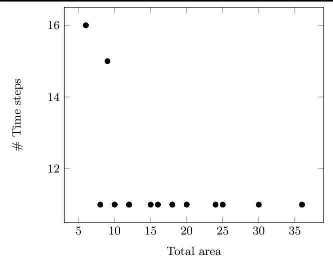
[4]Note that the solutions are symmetric with respect to the width and height of the grid, i.e. a solution for a $w \times h$ grid is a valid solution for a $h \times w$ grid.

Table 1: Determ. the min.

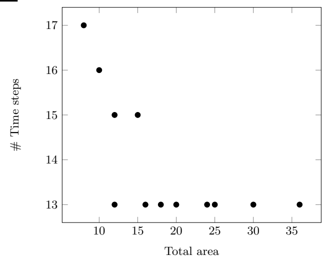
	ITERATIVE		BEST CASE	
T	RES	TIME	RES	TIME
1	UNSAT	0.90s	–	–
2	UNSAT	2.28s	–	–
3	UNSAT	3.71s	–	–
4	UNSAT	5.13s	–	–
5	UNSAT	6.73s	–	–
6	UNSAT	8.37s	–	–
7	UNSAT	10.22s	–	–
8	UNSAT	12.05s	–	–
9	UNSAT	13.89s	–	–
10	UNSAT	16.14s	–	–
11	UNSAT	18.04s	–	–
12	UNSAT	20.61s	–	–
13	UNSAT	23.16s	–	–
14	UNSAT	26.34s	–	–
15	UNSAT	29.29s	–	–
16	UNSAT	63.91s	UNSAT	63.91s
17	SAT	3152.45s	SAT	3152.45s
total		3349.94s		3216.36s

Table 2: Exact results

s	r	$(w \times h)$	T	TIME
2	1	4×6	14	51.05s
2	1	5×5	14	64.18s
2	1	5×6	14	87.52s
2	1	6×6	14	185.22s
2	2	2×5	16	66.96s
2	2	2×6	16	310.38s
2	2	3×4	16	173.48s
2	2	3×5	15	162.86s
2	2	3×6	15	376.0s
2	2	4×4	15	277.8s
2	2	4×5	15	359.32s
2	2	4×6	15	524.48s
2	2	5×5	15	503.28s
2	2	5×6	15	768.24s
2	2	6×6	15	1262.0s
2	3	3×6	17	3349.94s
2	3	4×6	16	1122.91s
2	3	5×6	16	1874.15s
2	3	6×6	16	2147.62s



(a) $s = 1, r = 2$



(b) $s = 1, r = 3$

Figure 8: Area vs. time

5.2 Exact Synthesis Results

Afterwards, the general applicability of the approach has been evaluated. To this end, the minimal completion time has been determined for several configurations of the benchmarks mentioned above as well as different grid sizes. Table 2 exemplarily presents some results. The first two columns provide the number of samples and reagents (denoted by s and r respectively), while the next two columns show the grid size. Afterwards the minimal completion time obtained by the proposed approach as well as the needed (CPU) runtime are reported.

As can clearly be seen, performing exact one-pass synthesis is computationally expensive. Mostly, this is due to the generally large complexity of the considered problem as already discussed in Section 3.2. Nevertheless, it is possible to determine minimal results for many relevant configurations. We derived time-optimal placements and routings for grids with an area of up to 6×6. For comparison, the heuristically generated results from [14] consider grid sizes of merely up to 2×18 – while, at the same time, only addressing the scheduling problem and *not* guaranteeing minimality. In fact, the presented results allow for an evaluation of best case scenarios for the first time. Besides that, they enable more detailed case studies as presented in the following.

5.3 Further Evaluations

In a final series of experiments, we examined how the obtained minimal results can be exploited for further, more in-depth, investigations. Exemplarily, the relation between the grid size and the corresponding minimal completion time has additionally been evaluated. Using the proposed approach, we investigated this issue for several benchmark configurations for different amounts of samples and reagents (s and r respectively). Figure 8 summarizes selected results in terms of plots. The x-axis denotes the total area (determined by multiplying the width and the height of the grid), while the y-axis represents the completion time.

It can be clearly seen from the results that less time steps are needed to realize the desired functionality, the larger the available grid (and vice versa). Although this is an expected result, it is now confirmed by computing the necessary exact realizations. Moreover, these results also allow for a better understanding of how to tackle the multi-objective optimization problem of minimizing both, the total area and the completion time. In fact, it can be observed that the area should be increased before increasing the number of time steps.

6. CONCLUSION

In this work, we proposed a one-pass synthesis scheme for microfluidic biochips which considers several important design steps at once and, at the same time, guarantees minimality with respect to area and/or timing. For this purpose, we utilized the deductive power of solvers for Boolean satisfiability. Experimental results confirmed the applicability of the methodology and provided examples of how this leverages the design of the respective devices. Future work will focus on extending the proposed approach by a more comprehensive support of further operations and their respective grid-size and timing requirements.

7. REFERENCES

[1] Microfluidic applications in the pharmaceutical, life sciences, in-vitro diagnostic and medical device markets report 2013. `http://www.researchandmarkets.com/research/z6nhg7/microfluidic`.

[2] C. Barrett, A. Stump, and C. Tinelli. The SMT-LIB Standard: Version 2.0. In A. Gupta and D. Kroening, editors, *International SMT Workshop (Edinburgh)*, 2010.

[3] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability.* IOS Press, 2009.

[4] Y.-H. Chen, C.-L. Hsu, L.-C. Tsai, T.-W. Huang, and T.-Y. Ho. A reliability-oriented placement algorithm for reconfigurable digital microfluidic biochips using 3D deferred decision making technique. *IEEE TCAD*, 32(8):1151–1162, 2013.

[5] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS*, pages 337–340. Springer, 2008. Z3 is available at http://z3.codeplex.com/.

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

[7] D. Grissom and P. Brisk. Fast online synthesis of generally programmable digital microfluidic biochips. In *ACM CODES+ISSS*, pages 413–422. ACM, 2012.

[8] F. Haedicke, S. Frehse, G. Fey, D. Große, and R. Drechsler. metaSMT: Focus on your application not on solver integration. In *DIFTS*, pages 22–29, 2011. metaSMT is available at https://github.com/agra-uni-bremen/metaSMT.

[9] T.-Y. Ho, J. Zeng, and K. Chakrabarty. Digital microfluidic biochips: A vision for functional diversity and more than moore. In *IEEE/ACM ICCAD*, pages 578–585. IEEE Press, 2010.

[10] E. Maftei, P. Pop, and J. Madsen. Tabu search-based synthesis of digital microfluidic biochips with dynamically reconfigurable non-rectangular devices. *DAES*, 14(3):287–307, 2010.

[11] M. Pollack, A. Shenderov, and R. Fair. Electrowetting-based actuation of droplets for integrated microfluidics. *Lab on a Chip*, 2(2):96–101, 2002.

[12] A. J. Ricketts, K. Irick, N. Vijaykrishnan, and M. J. Irwin. Priority scheduling in digital microfluidics-based biochips. In *DATE*, pages 329–334, 2006.

[13] J. Song, R. Evans, Y.-Y. Lin, B.-N. Hsu, and R. Fair. A scaling model for electrowetting-on-dielectric microfluidic actuators. *Microfluidics and Nanofluidics*, 7(1):75–89, 2009.

[14] F. Su and K. Chakrabarty. High-level synthesis of digital microfluidic biochips. *ACM JETC*, 3(4):1:1–1:32, January 2008.

[15] F. Su, W. Hwang, and K. Chakrabarty. Droplet routing in the synthesis of digital microfluidic biochips. In *DATE*, volume 1, pages 1–6, 2006.