# Design of Application-specific Architectures for Networked Labs-on-Chips

Andreas Grimmer *Student Member, IEEE,* Werner Haselmayr *Member, IEEE,* Andreas Springer *Member, IEEE,* and Robert Wille *Senior Member, IEEE*

***Abstract*—*Labs-on-Chips* implement laboratory procedures on a single chip and are successfully used for chemical and biomedical applications. A promising and emerging realization of such chips are *Networked Labs-on-Chips* (NLoCs) in which small volumes of fluids, so-called droplets, flow in closed channels of sub-millimeter diameters. NLoCs allow for an incubation and storage of assays over a long period of time and, hence, avoid evaporation and unwanted reactions. To increase the flexibility, effectiveness, and re-usability, network functionalities allow to passively route droplets in channels and, hence, to dynamically select operations depending on the executed experiment.**

**However, only manually designed architectures are considered for NLoCs thus far. They frequently suffer from large execution times and/or a high contamination of channels. To overcome these drawbacks, we propose the consideration of *application-specific architectures* for NLoCs. To this end, an automatic design method is proposed which, for a given set of experiments as well as constraints and objectives from the designer, is able to generate an optimized NLoC architecture realizing these experiments. Evaluations and case studies demonstrate the potential of the proposed solution for design exploration. Moreover, we are able to show that application-specific architectures are capable of realizing experiments in just a fraction of the time needed by architectures used thus far as well as with a substantially reduced contamination.**

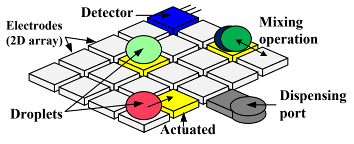
***Index Terms*—Networked Labs-on-Chips, biochips, architecture, design automation, satisfiability solvers.**

## I. Introduction

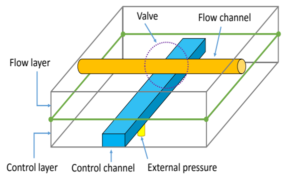
Advances in the microfluidic technologies have led to the emergence of so-called *Labs-on-Chips* (LoCs) in order to automate laboratory procedures in chemistry and molecular biology [1]. An LoC is a microfluidic system that realizes one or multiple experiments which are usually performed in a laboratory (examples include PCR [2], protein crystallization [3], nanoparticle synthesis [4], or cell encapsulation [5], [6]). By this, LoCs allow for an automation of many chemical and biological experiments and, hence, a faster analysis as well as larger throughput while, at the same time, a significantly lower fluid consumption is required.

In recent years, different technologies for LoCs have been considered [7], [8]. The following two technologies are well established:

- LoCs based on *electrowetting on dielectric* (EWOD-based LoCs; often also referred to as *Digital Microfluidic Biochips*, i.e. DMFBs; [9]) comprise a two-dimensional electrical *grid* controlled by underlying electrodes. This is illustrated in Fig. 1a. An activated electrode generates an electric field, which allows to "hold" discretized portions of liquids, so-called *droplets*, on a particular cell within the grid (as illustrated by means of the light green droplet in Fig. 1a). By assigning time-varying voltage values to turn electrodes on and off, droplets can be moved around the grid (as illustrated by the red droplet in Fig. 1a). By moving two droplets onto the same cell, mixing operations can be conducted (as illustrated by the green droplet and dark blue droplet in Fig. 1a). This eventually provides a platform on which *operations* such as mixing, heating, splitting, or analyzing can be conducted and, hence, complete experiments can be realized.



(a) EWOD-based LoC [14]



(b) Flow-based LoC [15]

Fig. 1: Established LoC technologies

- A *flow-based LoC* (often also referred as *microfluidic Very Large-Scale Integration*, i.e. mVLSI [10]) consists of hundreds or even thousands of integrated *microvalves* [8], [11], which are used to control the flow of liquids. Fig. 1b illustrates 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*. By combining and controlling the closing and opening of multiple valves, complex operations, such as merging, splitting, dispensing, and mixing [12], [13] can be realized.

However, both technologies inherit significant disadvantages: EWOD-based LoCs suffer from the evaporation of liquids, the fast degradation of surface coatings, and its lacking biocompatibility [16]–[20]. Flow-based LoCs require a complex and costly multilayer fabrication process [20], [21]. Hence, an alternative to these two technologies has been proposed in terms of *Networked Labs-on-Chips* (NLoCs, [22]–[24]). Fig. 2 shows a schematic of this promising and emerging realization. The droplets flow in microchannels of sub-millimeter diameters (triggered by pressure which is produced by an external pump). The closed channels allow for an incubation and storage of liquid assays over a long period of time and, hence, avoid evaporation and unwanted reac-

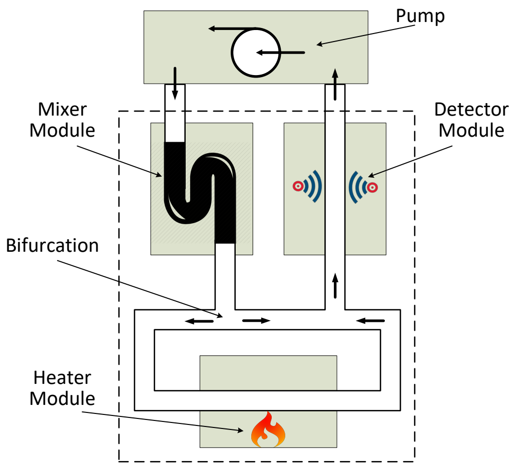


Fig. 2: Networked Labs-on-Chip

tions [7], [8]. Additional network functionalities enable the designer to dynamically select the operations to be conducted and, hence, to dynamically route a droplet to the operations involved in an experiment. For example, the NLoC shown in Fig. 2 enables the designer to decide whether droplets should be heated or not, before they get analyzed by the detector module.

This *passive* routing of droplets increases the flexibility, effectiveness, as well as re-usability of NLoCs and is controlled by exploiting hydrodynamic forces. Experiments are realized by defining multiple paths through which the droplets can flow. These paths are built by *bifurcations* of channels, i.e. the splitting of a channel in two or more successor channels. When a droplet arrives at a bifurcation, it will flow along the successor channel with the lowest *hydraulic resistance* (defined by the channel's geometry, e.g. the smaller the section and the longer the channel, the higher the resistance; see [18], [21], [23], [25]).

More precisely, the architecture of Fig. 2 contains a bifurcation after the mixer module. This bifurcation splits the channel after the mixer module into two successor channels. Assume that the hydraulic resistance of the successor channel to the right is lower than those of the channel to the left. When the pump now injects a droplet, the droplet gets mixed by the mixer module, flows to the right at the bifurcation, flows into the detector module, and eventually flows back to the pump. However, a droplet itself increases the channel's hydraulic resistance and, therefore, temporarily "blocks" this channel for following droplets. Consider again the bifurcation from the architecture of Fig. 2. Additionally, assume that a droplet already took the channel to the right and, therefore, temporarily blocks this channel during its flow. A second closely following droplet will flow into the channel to the left because the channel to the right is blocked. Hence, the second droplet additionally gets heated before it flows into the detector module.

This principle of selectively blocking channels is used to route droplets through the NLoC and, by this, to realize different experiments. We demonstrate this blocking principle in videos at http://www.jku.at/iic/eda/nloc using a physical realization. Furthermore, a more detailed consideration of determining a droplet sequence realizing the desired experiment and especially checking whether an NLoC architecture indeed allows for the realization of all experiments can be found in [26], which is based on a discrete model presented in [27].

Having this technology, the problem remains how to design the actual NLoCs so that indeed the desired experiments are realized. For EWOD-based and flow-based LoCs, corresponding automatic design solutions addressing e.g. issues such as scheduling, binding, placement, and routing have already been presented (see e.g. [14], [15], [28]–[34]). However, to the best of our knowledge, no (automatic) design solution exists for NLoCs yet. Up to now only manually designed architectures have been considered and realized (see e.g. [22], [35]). But designing an architecture which allows to execute all experiments, considers the physical constraints, and is optimized for the designer's needs is a cumbersome task. Due to this inherent complexity, the hand-crafted architectures are often unsuited for the given applications and suffer from large execution times (crucial for time-sensitive experiments) as well as a high contamination.

In this work, we aim for overcoming these obstacles by exploring the potential of NLoC architectures which go beyond the currently considered architectures. To this end, we propose the consideration of *application-specific architectures* which are particularly suited for a set of experiments to be realized. In order to generate these application-specific architectures, an automatic method based on *satisfiability solvers* (SAT-solvers, [36]) is introduced. The proposed method automatically generates architectures that are optimized with respect to various physical constraints and/or design objectives such as the number of required entities, maximally allowed contamination, etc.

An evaluation and case studies demonstrate how the proposed design method enables designers to explore various alternative solutions with respect to different criteria so that, eventually, they can choose the one which best fits to their current settings and requirements. We further evaluate the performance of the proposed design method and demonstrate the superiority of the resulting application-specific architectures compared to the currently used architectures. Finally, we show how the proposed method is integrated in the overall design flow.

The remainder of this paper is structured as follows: The next section introduces the state-of-the-art in the design of NLoCs. Afterwards in Sec. III, we motivate application-specific architectures and introduce their notation used in this work. The considered design task and our general idea to solve this task is described in Sec. IV. Next, implementation details are provided in Sec. V. Results of our evaluations and case studies are summarized in Sec. VI. Finally, the paper is concluded in Sec. VII.

## II. State-of-the-Art in NLoC Design

In a *Networked LoC* (NLoC, [22]–[24]), the droplets flow in closed microchannels and are *passively* controlled by hydrodynamic forces. The flow driving the droplets through the

NLoC is generated by a hydrodynamic pressure produced by a *pump*.

In order to realize an experiment, the droplet containing the biological sample has to traverse a sequence of modules, which include elementary operations such as droplet generation $p$ (the pump in combination with junctions is used for generating droplets; details are reviewed in [37]), mixing $m$ (mixes the biological sample within a droplet), splitting $s$ (breakup of droplets), fusion $f$ (the fusion of two or multiple droplets), detecting $d$, or heating $h$[1]. To enable the traversal of a droplet from one module to another, the modules have to be connected by directed channels.

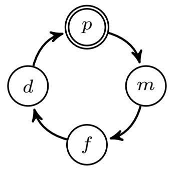
The design of the architecture (i.e. the determination of the required modules and how these are connected) is done manually thus far [22], [35]. However, when designing an architecture, the designer not only has to ensure that the architecture allows to execute all desired experiments but also has to consider further physical constraints, which are motivated by existing technologies. For example, two-way bifurcations as applied in [22] restrict the number of successor channels of a module to 2. Moreover, the architecture may not contain cycles without including the pump, since they would lead to undesired flow conditions. Besides these constraints, further architectural characteristics such as channel depth, number of modules, or contamination exists (since they heavily depend on the considered experiment and/or designer, they are discussed in more detail later in Sec. IV-A). Considering all these constraints and parameters makes the design of NLoC architectures a cumbersome task.

As a consequence, only rather simplistic architectures have been considered thus far. The ring architecture as sketched in Fig. 3a is the most frequent realization [22]. Here, a droplet cyclically traverses the ring in order to execute operations realized by modules. A module can either be executed or skipped by the droplet[2]. In case the operation order defined by the modules of the ring matches the required sequence of operations of the experiment, the droplet has to traverse the ring once. However, for many experiments it is very likely that a droplet may be sent back to the pump before the experiment is completed (e.g. when the order of the modules in the ring is different to the order required for the given experiment). In such a case, the pump re-injects the same droplet again into the architecture and the droplet traverses the ring one more time.

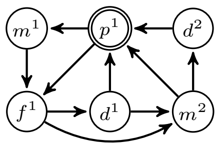
While such an architecture is trivial to design, it inherits severe drawbacks: In fact, for many experiments the droplet has to traverse the complete ring several times for conducting all operations in the required order – significantly increasing the execution time. These execution times might be particularly infeasible for time-sensitive experiments in which reactions depend on a quick execution of operations. Besides that, the respective droplets frequently pass through the same channels – eventually leading to a high contamination which may spoil the result.

[1]More details on the operations, including possible implementations, can be found in [38]–[42].

[2]A possible implementation of such a module is presented in [43].



(a) Ring architecture



(b) Application-specific architecture

Fig. 3: NLoC architectures

**Example 1.** *Consider the ring architecture as shown in Fig. 3a. Nodes in the architecture denote modules and edges denote channels. The directions of the edges indicate the flow direction of the droplets within the channels.*

*In order to execute the operation sequence p, f, m, d, and finally back to the pump p (written as $\phi_1 := (p, f, m, d, p)$), the ring has to be traversed two times. In the first traversal through the ring, only the operation f can be executed. Then, the pump re-injects the same droplet into the ring again. In the second traversal, the operations $m$ and $d$ are executed – completing the experiment. In each traversal, all four modules have to be passed (either the module is executed or skipped) – leading to a total number of eight steps for this experiment. Besides that, after this experiment has been completed, each channel has been used two times. If further experiments shall be realized, e.g. $\phi_2 := (p, m, f, d, p)$ and $\phi_3 := (p, f, d, m, p)$, overall $20$ steps are needed. Furthermore, each channel is used five times – leading to a substantial contamination.*

## III. Application-specific Architectures

In order to overcome the problems of the architectures considered thus far and to always guarantee the best realization for a given set of experiments and constraints, we propose the design of *application-specific* architectures. Here, multiple instances of modules may be applied in order to realize a set of experiments in as less as possible steps and/or with as low as possible contamination.

**Example 2.** *Fig. 3b shows an application-specific architecture, which is supposed to realize the same experiments as defined in Ex. 1. The graphical notation is the same as for ring architectures. In case a module has multiple incoming edges, the corresponding channels merge into a single input channel of the module. Furthermore, a module can have at most two successor edges, which can be physically implemented by a bifurcation of the outgoing channel of the module.*

*Although this architecture includes multiple (but still limited) instances of modules realizing the same operation (for $d$ and $m$, two instances each exist), it allows for conducting each experiment in four steps only (which is the minimal number of steps). Moreover, no edge is used more than once when executing any of the experiments – hence, the contamination is significantly reduced. When executing all experiments, $\phi_1$, $\phi_2$, and $\phi_3$, this architecture requires $12$ steps only (in contrast to the ring architecture which requires $20$ steps) and each edge is traversed at most two times (in contrast to the ring architecture which contaminates each edge five times).*

However, determining such an application-specific architecture is a non-trivial task. In the remainder of this section, we introduce the notion used to describe NLoC architectures and state the underlying design problem. Based on that, the remainder of this work focuses on how to generate the best possible NLoC architecture for a given set of experiments as well as constraints and objectives from the designer.

An NLoC realizes a set of experiments given by $\Phi$. Each experiment is accomplished by driving a droplet through a sequence of operations from a set $O$.

**Definition 1.** *Let* $\Phi := \{\phi_1, \phi_2, \ldots\}$ *be the* set of experiments *to be realized. Further let* $O := \{p, m, s, f, d, h, \ldots\}$ *be the* set of operations *used in the experiments* $\Phi$*. Then, an* experiment $\phi \in \Phi$ *is a sequence of operations with* $\phi \in O^n$*, where* $n \in \mathbb{N}$ *is a natural number defining the number of operations in the experiment.*

The realization of a set of experiments onto an NLoC requires modules executing the operation. To realize an experiment, the droplet has to traverse a sequence of modules executing the operations defined in the experiment. In order to allow the traversal of a droplet from one module to another, the modules have to be connected by directed channels. The used modules and their channels describe the architecture of an NLoC, which is formally defined as follows:

**Definition 2.** *Each operation* $o \in O$ *can be realized by one or more instances of a* module. *The function* $\mathrm{maxI} : O \rightarrow \mathbb{N}$ *defines the number of available instances of a module realizing an operation* $o$[3]*. Then, the total* set of all available modules *is defined by* $\hat{V} := \{(u^i) : u \in O \text{ and } 1 \leq i \leq \mathrm{maxI}(u)\}$ *where an element* $u^i \in \hat{V}$ *(also called* node *in the following) denotes the* $i^{th}$ instance of a module *realizing the operation* $u \in O$*. Further, the total* set of all possible channels between these modules *is defined by* $\hat{E} := \{(u^i, v^j) : u^i, v^j \in \hat{V}\}$*. An element* $(u^i, v^j) \in \hat{E}$ *(also called* edge *in the following) connects the output of the module* $u^i$ *with the input of the module* $v^j$ *with a channel. Taking* $\hat{V}$ *and* $\hat{E}$*, a* superset architecture $\hat{G} := (\hat{V}, \hat{E})$ *can be formed representing all possible architectures of an NLoC from the available modules. The* application-specific architecture $G = (V, E)$ *of an NLoC is a subset of the modules* $V \subseteq \hat{V}$ *and channels* $E \subseteq \hat{E}$ *from* $\hat{G}$ *which realizes the given set of experiments and, at the same time, satisfies physical constraints as well as further design objectives.*

**Example 3.** *Consider the set* $\Phi := \{\phi_1, \phi_2, \phi_3\}$ *of experiments to be realized with* $\phi_1 := (p, f, m, d, p)$*,* $\phi_2 := (p, m, f, d, p)$*, and* $\phi_3 := (p, f, d, m, p)$*. Additionally assume for the operations* $O := \{p, m, f, d\}$*, that there are* $maxI(p) = 1$*,* $maxI(m) = 2$*,* $maxI(f) = 2$*, and* $maxI(d) = 2$ *modules available. Then, an application-specific architecture of an NLoC realizing* $\Phi$ *can be derived from the superset architecture* $\hat{G} := (\hat{V}, \hat{E})$ *composed of the available modules* $\hat{V} := \{p^1, m^1, m^2, f^1, f^2, d^1, d^2\}$ *and possible channels* $\hat{E} := \{(p^1, p^1), (p^1, m^1), (p^1, m^2), (p^1, f^1), \ldots\}$*. This includes the solution requiring the minimal number of steps shown in Fig. 3b and discussed before in Ex. 2.*

However, the question remains how to obtain an optimized application-specific architecture. A naive realization, which always would allow for a realization of all experiments with minimal steps and minimal contamination, would be an architecture in which, for each operation of an experiment, a dedicated module and channel is built. But obviously, this would result in a too expensive architecture with no reuse. Hence, in the following, we consider the research question:

> *How to efficiently generate an application-specific architecture for a Networked Labs-on-Chip, which realizes the given experiments with minimal steps and/or contamination while, at the same time, keeps the architectural overhead with respect to modules and channels as small as possible?*

## IV. Design of Application-specific Architectures

In this section, we first discuss the task of designing application-specific architectures and the applied quality criteria. In order to handle the complexity of the design task (which is conducted once for a given set of experiments), we finally propose the utilization of powerful *satisfiability solvers* (SAT-solvers, [36]).

### A. Design Task

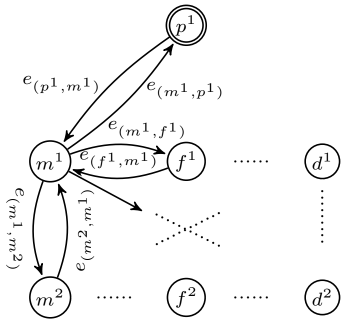
Recall that the main objective is the realization of an application-specific architecture realizing a set of experiments $\Phi := \{\phi_1, \phi_2, \ldots\}$ as discussed in Def. 2. Additionally, the architecture has to satisfy all physical constraints (i.e. maximal allowed successor edges of a node are limited to two and no cycles without including the pump are allowed) and, at the same time, has to satisfy the designer's quality requirements.

In order to define the quality of an architecture, several design objectives can be used. We focus on the following important quality criteria[4]:

- The *channel depth*, i.e. the maximum distance a droplet has to traverse from the pump to the modules and, eventually, back to the pump. Hence, the maximal channel depth is determined by the experiment that requires the longest path in the application-specific architecture. A small channel depth is desirable to decrease the overall execution time of the experiments and also to minimize the required pump pressure.
- The *number of channels*, i.e. $|E|$. This metric represents the complexity of the physical structure to be realized in the microfluidic system.
- The *number of modules* used in the architecture, i.e. $|V|$. This metric obviously is important, since more modules in an architecture cause higher fabrication costs.

[3]Currently, NLoCs mostly have a single instance of a pump producing the continuous flow, i.e. $maxI(p) = 1$.

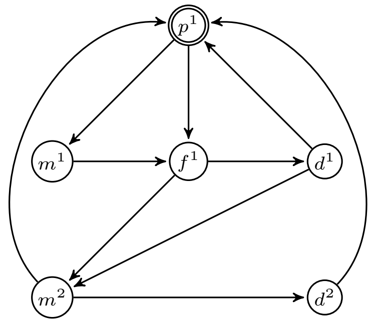
[4]Note that the proposed solution can easily be extended to support further objectives.



(a) $\hat{G}$ and corresponding $e_{(u^i,v^j)}$-vars

$$\begin{array}{ll} e_{(p^1,m^1)} = 1 & e_{(p^1,f^1)} = 1 \\ e_{(p^1,f^1)} = 1 & e_{(f^1,d^1)} = 1 \\ e_{(f^1,m^2)} = 1 & e_{(d^1,m^2)} = 1 \\ e_{(m^2,p^1)} = 1 & e_{(m^2,d^2)} = 1 \\ e_{(d^1,p^1)} = 1 & e_{(d^2,p^1)} = 1 \\ & \\ e_{(u^i,v^j)} = 0 & \text{for all remaining vars.} \end{array}$$

(b) Possible assignment to $e_{(u^i,v^j)}$-vars



(c) $G$ represented by assignment

Fig. 4: Symbolic formulation of all possible architectures

- The *contamination*, i.e. the maximum and/or average number of times a channel is passed by a droplet (assuming all experiments are conducted). The more often a droplet traverses a channel, the higher the probability that the channel is contaminated and spoils an experiment.

Determining an architecture, which optimizes one or multiple objectives, is a computationally hard problem. To handle this complexity, we propose to exploit the deductive power of SAT-solvers. Our thesis is that SAT-solvers can be used to generate optimized application-specific architectures. However, corresponding SAT-solvers are specialized for solving decision problems. Hence, we next discuss how to re-phrase the design problem considered here as a decision problem.

## B. General Idea

In order to utilize SAT-solvers, we have to re-phrase the design problem considered here as a *decision problem*, i.e.

*"Is there an architecture which*

*(i) realizes all experiments,*

*(ii) satisfies all physical constraints, and*

*(iii) is optimized with respect to one or more of the quality criteria?"*

Furthermore, this decision problem has to be formalized in a fashion that can be handled by the utilized SAT-solvers. To this end, the following steps are conducted:

- A symbolic formulation representing all possible architectures $\hat{G}$ to be considered is created. This is done by encoding these architectures in terms of Boolean variables.
- The possible assignments are restricted (by additional Boolean constraints) so that only assignments result which represent architectures realizing all experiments and satisfying all physical constraints (employing issue *(i)* and *(ii)* of the decision problem).
- An objective function is added which enforces the SAT-solver to determine that particular assignment (out of the remaining ones) which is optimized or even optimal with respect to one or more of the quality criteria (employing issue *(iii)* of the decision problem).

Passing the resulting formulation to the SAT-solver either yields an assignment (which is optimized with respect to the objective function) or proves that no assignment satisfying all constraints exists. In the former case, the desired architecture can be derived from the obtained variable assignments. In the latter case, it has been proven that no architecture exists which realizes all experiments under the given constraints. In the next section, details to all these steps are provided.

## V. Implementation

In this section, the details on the corresponding SAT-formulation are provided. To this end, we explain the implementation of all steps of the decision problem reviewed in Sec. IV-B.

### A. Symbolic Formulation of all Architectures

In order to symbolically formulate all possible architectures to be considered, an encoding of all possible graphs with nodes $V \subseteq \hat{V}$ and edges $E \subseteq \hat{E}$ is created. To this end, a symbolic *one-hot encoding* is applied where a single Boolean variable represents whether an edge possible according to $\hat{E}$ is indeed present in $E$. More formally:

**Definition 3.** *Let $\hat{G} = (\hat{V}, \hat{E})$ be the superset architecture out of which a sub-graph $G$ representing the desired architecture shall be derived. Then, for each edge $(u^i, v^j) \in \hat{E}$, a free Boolean variable $e_{(u^i,v^j)}$ with $u^i, v^j \in \hat{V}$ is introduced*[5]. *These variables represent whether ($e_{(u^i,v^j)} = 1$) or not ($e_{(u^i,v^j)} = 0$) there is an edge from node $u^i$ to node $v^j$.*

Note that, by this, also the presence of nodes in $G$ is implicitly represented: If an assignment is applied representing no incoming edge to a node $v^j \in \hat{V}$ (i.e. $e_{(u^i,v^j)} = 0$ for all $u^i \in \hat{V} \setminus \{v^j\}$), then $v^j$ is not present in $G$.

**Example 4.** *Consider again the setting as given in Ex. 3. Fig. 4a sketches the resulting graph $\hat{G} = (\hat{V}, \hat{E})$ together with the introduced $e_{(u^i,v^j)}$-variables. A possible assignment to these variables is given in Fig. 4b leading to a graph $G$ as shown in Fig. 4c (which is equal to the architecture shown in Fig. 3b).*

[5]Note again that $u^i \in \hat{V}$ denotes a node representing the $i^{th}$ instance ($1 \leq i \leq \max\mathrm{I}(u)$) of a module realizing the operation $u \in O$.

| $\phi_1 :=$ | ( $p$, | $f$, | $m$, | $d$, | $p$ ) |
|---|---|---|---|---|---|
| | $\phi_1[1] = p$ | $\phi_1[2] = f$ | $\phi_1[3] = m$ | $\phi_1[4] = d$ | $\phi_1[5] = p$ |
| Variables: | $ex_{\phi_1[1]^1}$ | $ex_{\phi_1[2]^1}$ $ex_{\phi_1[2]^2}$ | $ex_{\phi_1[3]^1}$ $ex_{\phi_1[3]^2}$ | $ex_{\phi_1[4]^1}$ $ex_{\phi_1[4]^2}$ | $ex_{\phi_1[5]^1}$ |
| Assignments: | $ex_{\phi_1[1]^1} = 1$ | $ex_{\phi_1[2]^1} = 1$ $ex_{\phi_1[2]^2} = 0$ | $ex_{\phi_1[3]^1} = 0$ $ex_{\phi_1[3]^2} = 1$ | $ex_{\phi_1[4]^1} = 0$ $ex_{\phi_1[4]^2} = 1$ | $ex_{\phi_1[5]^1} = 1$ |

Fig. 5: Variables representing realizations of experiment $\phi_1$

Passing this symbolic formulation to a SAT-solver would yield an arbitrary assignment to the $e_{(u^i,v^j)}$-variables and, hence, an arbitrary sub-graph $G$ of $\hat{G}$. Since we are not interested in an arbitrary graph but one that satisfies the experiments and constraints, we now have to restrict the possible assignments to all $e_{(u^i,v^j)}$-variables.

## B. Realization of Experiments

First, restrictions are enforced which only allow for $e_{(u^i,v^j)}$-assignments that represent graphs realizing the desired set of experiments $\Phi$. Recall that an experiment $\phi \in \Phi$ is given as a sequence of operations and these operations have to be executed by respective modules. Hence, we have to make sure that the graph $G$ contains a path of nodes realizing the operation sequence given by $\phi$. To this end, we introduce new Boolean variables representing all possibilities how an experiment can be realized.

**Definition 4.** *Let $\phi \in \Phi$ be an experiment and $\phi[p] \in O$ with $1 \leq p \leq |\phi|$ be the $p^{th}$ operation in it (where $|\phi|$ is the length of the experiment). Further, let $\phi[p]^i \in \hat{V}$ with $1 \leq i \leq \text{maxI}(\phi[p])$ be the $i^{th}$ instance of the module to be employed at position $p$. Then, for all $\phi[p]^i$, new Boolean variables $ex_{\phi[p]^i}$ are introduced. These variables represent whether ($ex_{\phi[p]^i} = 1$) or not ($ex_{\phi[p]^i} = 0$) the module at position $p$ of experiment $\phi$ is realized using the $i^{th}$ instance.*

**Example 5.** *Consider the experiment $\phi_1 := \{p, f, m, d, p\}$ defined in Ex 3. The newly introduced notations and variables are summarized in Fig. 5, i.e. the sequence of modules as specified by $\phi_1$ (first row), the corresponding $\phi[p]$-notation (second row), as well as the accordingly introduced $ex_{\phi[p]^i}$-variables (third row). A possible assignment to these $ex_{\phi[p]^i}$-variables is given in the fourth row. This eventually represents that the experiment $\phi_1$ is realized using the instances $(p^1, f^1, m^2, d^2, p^1)$.*

Since the SAT-solver may arbitrarily assign the newly introduced $ex_{\phi[p]^i}$-variables, a constraint has to be added which enforces that each operation in an experiment is realized by exact one module. To this end, for each experiment $\phi \in \Phi$ and each operation $\phi[p]$ of $\phi$, all possible instances (represented by $ex_{\phi[p]^i}$ with $1 \leq i \leq \text{maxI}(\phi[p])$) are considered. Then, it is enforced that only one of them indeed is realizing the operation, i.e. only one of the corresponding $ex_{\phi[p]^i}$-variables is set to 1. This is accomplished by the constraint

$$\bigwedge_{\phi\in\Phi} \bigwedge_{p=1}^{|\phi|} \left( \sum_{i=1}^{\text{maxI}(\phi[\text{p}])} ex_{\phi[p]^i} \right) = 1.$$

Finally, it has to be ensured that the respectively chosen chain of instances realizing the experiment $\phi$ (represented by $ex_{\phi[p]^i}$-variables) is indeed also realized in the graph $G$. To this end, the $e_{(u^i,v^j)}$-variables have to be restricted depending on the respective values of the $ex_{\phi[p]^i}$-variables. This is enforced by the constraint

$$\bigwedge_{\phi\in\Phi} \bigwedge_{p=1}^{|\phi|-1} \bigwedge_{i=1}^{\text{maxI}(\phi[p])} \bigwedge_{j=1}^{\text{maxI}(\phi[p+1])} ex_{\phi[p]^i} \wedge ex_{\phi[p+1]^j} \implies e_{(\phi[p]^i,\phi[p+1]^j)},$$

which checks for each experiment $\phi \in \Phi$ and each sequence of operations $\phi[p]$ and $\phi[p+1]$ whether the instances $\phi[p]^i$ and $\phi[p+1]^j$ are used to realize them, respectively (i.e. whether the variables $ex_{\phi[p]^i}$ and $ex_{\phi[p+1]^j}$ are set to 1). If this is true, then it is enforced that $G$ contains a channel between these modules (i.e. that $e_{(\phi[p]^i,\phi[p+1]^j)}$ is set to 1).

Passing this (extended) formulation to a SAT-solver would yield an assignment to the $e_{(u^i,v^j)}$-variables which will represent a sub-graph $G$ of $\hat{G}$ realizing all given experiments.

## C. Satisfying Physical Constraints

After the realization of the experiments is guaranteed, we address that the physical constraints are satisfied, i.e. that the maximal number of successor edges of a node are limited to two and that resulting architectures do not contain cycles, which do not include the pump.

In order to restrict the maximal number of successor edges of a node to two, we have to ensure for each node except the pump (i.e. $u^i \in \hat{V} \setminus \{p^1\}$) that the number of successor edges (i.e. the number of variables $e_{(u^i,v^j)}$ set to 1) is less than or equal to 2, i.e.

$$\bigwedge_{u^i\in\hat{V}\setminus\{p^1\}} \left( \sum_{v^j\in\hat{V}} e_{(u^i,v^j)} \right) \leq 2.$$

Next, it has to be ensured that the resulting architectures do not contain cycles, which do not include the pump. Therefore, all edges which do not have the pump as source or destination (i.e. $(u^i, v^j)$ where $u^i, v^j \in \hat{V} \setminus \{p^1\}$) must not build a path which allows to start at some node $u^i$ and follow a sequence of edges that eventually loops back to $u^i$ again. To this end, we again introduce new Boolean variables – this time for representing all paths possible in the graph $G$.

**Definition 5.** *Let $(u^i, v^j) \in (\hat{V} \setminus \{p^1\}) \times (\hat{V} \setminus \{p^1\})$ be all tuples of modules excluding the pump. Then, for every tuple a new Boolean variable $pth_{(u^i,v^j)}$ is introduced, which* *represents the existence* ($pth_{(u^i,v^j)} = 1$) *or non-existence* ($pth_{(u^i,v^j)} = 0$) *of a path from node $u^i$ to $v^j$. A path is a sequence of edges starting at node $u^i$ which eventually ends in node $v^j$.*

Obviously, these $pth_{(u^i,v^j)}$-variables depend on the respective assignment of the $e_{(u^i,v^j)}$-variables. If there exists an edge between $u^i \in \hat{V}$ and $v^j \in \hat{V}$ (i.e. if $e_{(u^i,v^j)}$ is set to 1), then there also exists a path between these nodes. This is accomplished by the constraint

$$\bigwedge_{u^i \in \hat{V} \setminus \{p^1\}} \bigwedge_{v^j \in \hat{V} \setminus \{p^1\}} e_{(u^i,v^j)} \implies pth_{(u^i,v^j)}.$$

In a similar fashion, also the transitive relations of paths are enforced, i.e. if there is a path between the nodes $u^i \in \hat{V}$ and $v^j \in \hat{V}$ and a path between the nodes $v^j$ and $w^k \in \hat{V}$, then there is also a path between the nodes $u^i$ and $w^k$. This motivates the constraint

$$\bigwedge_{u^i \in \hat{V} \setminus \{p^1\}} \bigwedge_{v^j \in \hat{V} \setminus \{p^1\}} \bigwedge_{w^k \in \hat{V} \setminus \{p^1\}} pth_{(u^i,v^j)} \wedge pth_{(v^j,w^k)} \implies pth_{(u^i,w^k)}.$$

By this, all paths possible in the graph $G$ are recursively defined and become accessible due to the $pth_{(u^i,v^j)}$-variables. Now, it only has to be ensured that no cyclic paths occur, i.e. if there is a path between the nodes $u^i \in \hat{V}$ and $v^j \in \hat{V}$, there must not be an edge from the nodes $v^j$ to node $u^i$. Hence, enforcing

$$\bigwedge_{u^i \in \hat{V} \setminus \{p^1\}} \bigwedge_{v^j \in \hat{V} \setminus \{p^1\}} pth_{(u^i,v^j)} \implies e_{(v^j,u^i)} = 0$$

prohibits the generation of graphs $G$ with cycles.

Passing the resulting formulation to a SAT-solver now only yields assignments representing architectures realizing the desired experiments and satisfying the physical constraints. If the SAT-solver determines that no assignment is possible satisfying all constraints, it has been proven that the desired experiments cannot be realized given the number of possible instantiations for each module. Then, the designer has to increase the respective values for $\mathrm{maxI}$. If a satisfying assignment is determined, the corresponding graph can easily be derived from the respective values of the $e_{(u^i,v^j)}$-variables (as illustrated in Fig. 4).

### *D. Employing the Quality Criteria*

Usually, the formulation presented above yields a significant number of possible assignments. This is because the considered experiments can usually be realized by a large number of possible architectures (in particular, if the respective values for $\mathrm{maxI}$ are large and, hence, a high degree of freedom exists). However, all these architectures significantly differ in their quality. Hence, designers might be interested in a solution, which is optimized or even optimal with respect to one or more of the quality criteria.

In order to employ that, we exploit the fact that many SAT-solvers do not only allow for determining a satisfying assignment, but an assignment which is additionally optimized with respect to an objective function[6]. More precisely, in order to determine a graph $G$ optimized with respect to

- the *number of channels*, we have to minimize the number of edges in $G$ by using the number of $e_{(u^i,v^j)}$-variables set to 1, i.e.

$$\min: \sum_{(u^i,v^j) \in \hat{E}} e_{(u^i,v^j)},$$

- the *number of modules*, we have to minimize the number of nodes contained in $G$, i.e.

$$\min: \sum_{v^j \in \hat{V}} \left( \bigvee_{u^i \in \hat{V}} e_{(u^i,v^j)} \right), \text{ or}$$

- the *contamination*, we have to minimize the number how often an edge $(u^i,v^j) \in \hat{E}$ is used in all experiments, i.e.

$$\min: \max_{(u^i,v^j) \in \hat{E}} \Bigg( \sum_{\phi \in \Phi} \bigvee_{p=1}^{|\phi|-1} \phi[p] = u \wedge \phi[p+1] = v \wedge ex_{\phi[p]^i} \wedge ex_{\phi[p+1]^j} \Bigg).$$

  This objective function sums up the usages of an edge $(u^i,v^j) \in \hat{E}$ over all experiments $\phi \in \Phi$ and, then, minimizes the most frequently used edge. Whether an edge is actually used in an experiment can be determined using the $ex_{\phi[p]^i}$-variables.

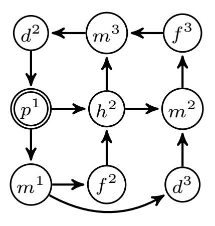
The quality criteria *channel depth* is always optimal because the constraints described in Sec. V-B ensure direct channels between modules realizing consecutive operations defined in the experiments.

## VI. Evaluation and Case Studies

The proposed approach has been implemented in Java resulting in an automatic method which enables designers to efficiently generate application-specific architectures for a given set of experiments and user-defined parameters such as the maximal number of module instances, an objective function employing the desired quality criteria, etc. The resulting architectures are optimized or even optimal with respect to the provided settings and criteria. To efficiently handle the complexity, the solver Z3 [44] in its latest version has been utilized. In order to demonstrate and evaluate the applicability of the proposed solution, several experiments and case studies have been conducted using experiments to be realized on an NLoC. Therefore, we considered for each NLoC a set of experiments which are generated out of sequencing graphs from [45]. More precisely, we considered the operations along each path through the graph as an experiment.

In the following, we focus on a selection of representatives which show the applicability and performance of the proposed method, the improvement of application-specific architectures compared to the current state-of-the-art, as well as their integration in the overall design flow. All experiments have been conducted on a 3.2 GHz Intel Core i5 machine with 8GB of memory running 64-bit Ubuntu 14.04.

[6]Alternatively to objective functions, it is also possible to define constraints restricting the different quality criteria.


(a) Min. modules/channels

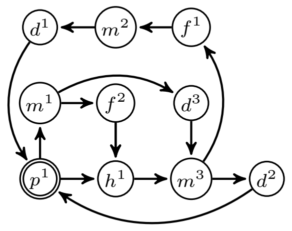

(b) Max. contamination of two

Fig. 6: Architectures for Benchmark 1

### A. Potential for Design Exploration

In a first series of experiments, we evaluated the potential of the proposed solution for design exploration. As discussed above, designers might be interested in architectures which are optimized with respect to different (even contradictory) quality criteria. Using the objective functions, the method can easily be adjusted to generate an architecture which e.g. is optimized for the number of channels or the contamination.

That this can make a difference is exemplarily shown by realizing the set of experiments $\Phi := \{(p, m, f, h, m, d, p), (p, m, d, m, f, m, d, p), (p, h, m, f, m, d, p)\}$[7]. Realizing this set of experiments while aiming first for a minimal channel depth, second for a minimal number of modules, and third for a minimal number of channels yields a solution as shown in Fig. 6a. However, this architecture has a worst-case contamination of three, e.g. the channel represented as edge $(m^3, d^2)$ is used three times. If application-specific constraints require the generation of an architecture where e.g. the worst-case contamination is not greater than two, the addition of a constraint limiting the contamination yields a solution as shown in Fig. 6b. Although this architecture requires one more module and one more channel (the trade-off to be accepted), it guarantees a worst-case contamination of two and, hence, satisfies the additional constraint.

In a similar fashion, alternative (application-specific) design objectives can be employed. This makes the proposed solution flexible and allows for an easy and efficient design exploration. By this, designers are supported by alternatives (including their pros and cons) allowing them to trade-off their respective design needs.

### B. Performance Evaluation

Determining $G$ is obviously a computationally hard problem because for each experiment $\phi \in \Phi$, the SAT-solver has to determine the respective modules (i.e. valid assignments for the $ex_{\phi[p]^i}$-variables). Here a huge number of possibilities exits. For example, let $r$ be the maximal number of allowed instances for all modules and let $|\phi|$ the length of an experiment. Then, for each of the $|\phi|$ positions, $r$ variables are introduced (namely $ex_{\phi[p]^1}, \ldots, ex_{\phi[p]^r}$) where exactly one of them has to be equal to 1 – resulting in $r$ possible combinations. Multiplying these combinations for all $|\phi|$ positions would yield $r^{|\phi|}$ combinations for an experiment to be realized – an exponential complexity. This becomes even more complex when all experiments in $\Phi$ are considered.

In order to evaluate whether the proposed method is capable of dealing with this complexity, we performed evaluations using randomly generated benchmarks, which can be scaled with respect to the number of experiments and their length. This way, we were able to evaluate the performance of the proposed solution with respect to various problem sizes. In this performance evaluation, we increased the problem size until the proposed method did not generate an architecture within 60 minutes.

Using five operations (a reasonable number of basic operations), the method is capable of determining architectures for up to

- 32 experiments with an average channel length of 7.28 (the resulting architecture consists of 30 nodes and 170 edges).
- an average experiment length of 13 for 6 experiments (the resulting architecture consists of 41 nodes and 103 edges).

### C. Comparison to the State-of-the-Art

The ring constitutes the commonly used architecture so far. However, this architecture suffers e.g. from a large execution time (specified by the channel depth) and a high contamination (specified by the usage of the channels). Application-specific architectures as generated by the proposed method yield significantly better results in this regard.

This is confirmed by the results summarized in Table I for experiments generated out of sequencing graphs from [45] (using both the number of channels and the number of modules as optimization criteria). The first five columns provide the name, the number of operations ($|O|$), the number of experiments ($|\Phi|$), the maximal number of considered instances for each module (Max. Instances), and the average length of the experiments (Avg. $\phi$ Length). Afterwards, the respective results are reported if the experiments have been realized by means of a ring[8] (denoted by $R$) or an application-specific architecture (denoted by $G$). More precisely, the number of modules ($|V|$) and channels ($|E|$) specify the size of the resulting architecture. The maximal channel depth (Max. Channel Depth) and the average channel depth (Avg. Channel Depth) state how many channels the droplet has to flow through at most/on average in order to conduct any of the experiments. Furthermore, the maximal contamination (Max. Contamination) and the average contamination (Avg. Contamination) state how often a channel is maximally traversed or how often the channels are in average traversed by the droplet containing the biological sample, respectively. Furthermore, we provide the run-time (in CPU seconds) necessary to generate the architecture.

The results show the benefits of the application-specific architectures. The increase of the NLoC sizes is acceptable as the size is not a limiting factor at the moment. Much

[7]This is one set of experiments generated out of a sequencing graph from [45].

[8]The module order with the lowest channel depth is used for a fair comparison.

TABLE I: Comparison to the state-of-the-art

| Benchmark | $\lvert O\rvert$ | $\lvert\Phi\rvert$ | Max. Instances | Avg. $\phi$ Length | $\lvert V\rvert$ | | $\lvert E\rvert$ | | Max. Channel Depth | | Avg. Channel Depth | | Max. Contamination | | Avg. Contamination | | Time [s] | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | $R$ | $G$ | $R$ | $G$ | $R$ | $G$ | $R$ | $G$ | $R$ | $G$ | $R$ | $G$ | $R$ | $G$ |
| B1 | 5 | 3 | 3 | 6.33 | 5 | 9 | 5 | 12 | 15 | 7 | 13.33 | 6.33 | 8 | 3 | 8 | 1.58 | 1 | 3 |
| B2 | 5 | 8 | 6 | 6.00 | 5 | 11 | 5 | 19 | 25 | 6 | 18.75 | 6.00 | 30 | 8 | 30 | 2.52 | 1 | 2820 |
| B3 | 5 | 10 | 6 | 7.70 | 5 | 13 | 5 | 23 | 30 | 8 | 24.50 | 7.70 | 49 | 10 | 49 | 3.35 | 3 | 2642 |
| B4 | 5 | 12 | 6 | 7.83 | 5 | 15 | 5 | 28 | 30 | 8 | 24.17 | 7.83 | 58 | 12 | 58 | 3.36 | 22 | 3996 |
| B5 | 5 | 14 | 7 | 9.21 | 5 | 18 | 5 | 33 | 30 | 10 | 26.07 | 9.21 | 73 | 14 | 73 | 3.91 | 206 | 8982 |

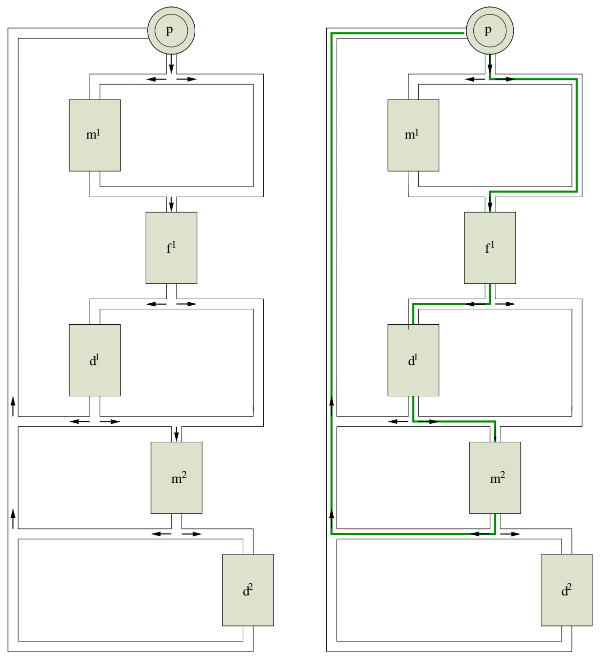
$R$: ring architecture $G$: application-specific architecture $|O|$: number of operations $|\Phi|$: number of experiments Max. Instances: maximal instances of each module Avg. $\phi$ Length: average length of the experiments $|V|$: number of nodes $|E|$: number of edges Max./Avg. Channel Depth: maximal and average channel depth Max./Avg. Contamination: maximal and average contamination Time [s]: required run-time in CPU seconds to generate the architecture

more important is the significant decrease in channel depth and also contamination, which are crucial criteria for many time-sensitive experiments. Therefore, the resulting architectures constitute a significant improvement. Moreover, further improvements are possible if additional quality criteria are employed.

### *D. Integration into the Design Flow*

Finally, we describe how the proposed method is embedded in the design flow and how physical implementations can be obtained from the application-specific architectures. Recall that the proposed solution provides the determined architecture in terms of a graph composed of nodes and edges. In the physical implementation, each node is realized by a corresponding module/pump and each edge is realized by a corresponding channel. The direction of the edges indicates the intended flow direction of the droplets within these channels. If a node has two successor edges, a bifurcation is used in the resulting physical implementation. Similarly, multiple incoming edges to a node are merged into a single channel. Finally, a physical implementation can be fabricated at low costs with 3D printers or engraving machines using polymers as base material [37], [46].

Overall, this yields a first step in the design flow for NLoCs: Given a set of experiments (e.g. $\Phi := \{\phi_1, \phi_2, \phi_3\}$ with $\phi_1 := (p, f, m, d, p)$, $\phi_2 := (p, m, f, d, p)$, and $\phi_3 := (p, f, d, m, p)$ as considered in the example), the method proposed in this work allows for the automatic generation of an application-specific architecture as shown in terms of a graph in Fig. 3b. This architecture can then be mapped to a physical implementation whose blueprint is shown in Fig. 7a. After the realization of this blueprint, e.g. the experiment $\phi_3$ can be conducted by letting the droplet containing the biological sample flow through the path indicated in Fig. 7b.



(a) Physical structure (b) Path for $\phi_3 = (p, f, d, m, p)$

Fig. 7: Blueprint of the architecture from Fig. 3b

## VII. Conclusions

In this work, we considered the design of Networked Labs-on-Chips – a promising and emerging technology which overcomes drawbacks of established LoC technologies. In a first step towards an automatic design flow for NLoCs, we propose the design of application-specific architectures. These application-specific architectures address drawbacks of manually designed architectures such as large execution times or high contaminations. In order to derive the respective architectures, a method has been proposed which tackles the underlying complexity using SAT-solvers. Evaluations and case studies showed that the proposed method enables designers to generate architectures satisfying their needs as well as quality criteria and confirmed the advantages of application-specific architectures compared to the current state-of-the-art.

## References

[1] A. J. Demello, "Control and detection of chemical reactions in microfluidic systems." *Nature*, vol. 442, no. 7101, pp. 394–402, 2006.

[2] Z. Guttenberg, H. Müller, H. Habermüller, A. Geisbauer, J. Pipper, J. Felbel, M. Kielpinski, J. Scriba, and A. Wixforth, "Planar chip device for PCR and hybridization with surface acoustic wave pump," *Journal on Lab on a Chip*, vol. 5, no. 3, pp. 308–317, 2005.

[3] B. Zheng, L. S. Roach, and R. F. Ismagilov, "Screening of protein crystallization conditions on a microfluidic chip using nanoliter-size droplets," *Journal of the American Chemical Society*, vol. 125, no. 37, pp. 11 170–11 171, 2003.

[4] L.-H. Hung, K. M. Choi, W.-Y. Tseng, Y.-C. Tan, K. J. Shea, and A. P. Lee, "Alternating droplet generation and controlled dynamic droplet fusion in microfluidic device for CdS nanoparticle synthesis," *Journal on Lab on a Chip*, vol. 6, no. 2, pp. 174–178, 2006.

[5] M. He, J. S. Edgar, G. D. Jeffries, R. M. Lorenz, J. P. Shelby, and D. T. Chiu, "Selective encapsulation of single cells and subcellular organelles into picoliter-and femtoliter-volume droplets," *Journal of Analytical Chemistry*, vol. 77, no. 6, pp. 1539–1544, 2005.

[6] Y.-C. Tan, K. Hettiarachchi, M. Siu, Y.-R. Pan, and A. P. Lee, "Controlled microfluidic encapsulation of cells, proteins, and microbeads in lipid vesicles," *Journal of the American Chemical Society*, vol. 128, no. 17, pp. 5656–5658, 2006.

[7] S. Haeberle and R. Zengerle, "Microfluidic platforms for Lab-on-a-Chip applications," *Journal on Lab on a Chip*, vol. 7, pp. 1094–1110, 2007.

[8] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle, "Microfluidic Lab-on-a-Chip platforms: requirements, characteristics and applications," *Journal of Chemical Society Reviews*, vol. 39, no. 3, pp. 1153–1182, 2010.

[9] M. G. Pollack, A. D. Shenderov, and R. B. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Journal on Lab on a Chip*, vol. 2, no. 2, pp. 96–101, 2002.

[10] I. E. Araci and S. R. Quake, "Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves," *Journal on Lab on a Chip*, vol. 12, no. 16, pp. 2803–2806, 2012.

[11] 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, Architecture and Synthesis for Embedded Systems*, 2014, pp. 1–10.

[12] T. M. Squires and S. R. Quake, "Microfluidics: Fluid physics at the nanoliter scale," *Journal on Reviews of Modern Physics*, vol. 77, no. 3, p. 977, 2005.

[13] V. Studer, G. Hang, A. Pandolfi, M. Ortiz, W. F. Anderson, and S. R. Quake, "Scaling properties of a low-actuation pressure microfluidic valve," *Journal of Applied Physics*, vol. 95, no. 1, pp. 393–398, 2004.

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

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

[16] D. R. Link, E. Grasland-Mongrain, A. Duri, F. Sarrazin, Z. Cheng, G. Cristobal, M. Marquez, and D. A. Weitz, "Electric control of droplets in microfluidic devices," *Angewandte Chemie International Edition*, vol. 45, no. 16, pp. 2556–2560, 2006.

[17] C.-G. Yang, Z.-R. Xu, and J.-H. Wang, "Manipulation of droplets in microfluidic systems," *Trends in Analytical Chemistry*, vol. 29, no. 2, pp. 141–157, 2010.

[18] A. Biral, D. Zordan, and A. Zanella, "Modeling, simulation and experimentation of droplet-based microfluidic networks," *Trans. on Molecular, Biological, and Multi-scale Communications*, vol. 1, no. 2, pp. 122–134, 2015.

[19] E. D. Leo, L. Donvito, L. Galluccio, A. Lombardo, and G. Morabito, "Microfluidic networks: design and test of a pure hydrodynamic switching function," in *Int'l Conf. on Communications*, 2013, pp. 787–791.

[20] L. Donvito, L. Galluccio, A. Lombardo, and G. Morabito, "$\mu$-NET: a network for molecular biology applications in microfluidic chips," *Trans. on Networking*, 2015.

[21] ——, "On the assessment of microfluidic switching capabilities in NLoC networks," in *Int'l Conf. on Nanoscale Computing and Communication*, 2014, p. 19.

[22] E. De Leo, L. Galluccio, A. Lombardo, and G. Morabito, "Networked labs-on-a-chip (NLoC): Introducing networking technologies in microfluidic systems," *Journal of Nano Communication Networks*, vol. 3, no. 4, pp. 217–228, 2012.

[23] E. D. Leo, L. Donvito, L. Galluccio, A. Lombardo, G. Morabito, and L. M. Zanoli, "Communications and switching in microfluidic systems: Pure hydrodynamic control for networking Labs-on-a-Chip," *Trans. on Communications*, vol. 61, no. 11, pp. 4663–4677, 2013.

[24] S.-Y. Teh, R. Lin, L.-H. Hung, and A. P. Lee, "Droplet microfluidics," *Journal on Lab on a Chip*, vol. 8, pp. 198–220, 2008.

[25] E. D. Leo, L. Donvito, L. Galluccio, A. Lombardo, G. Morabito, and L. M. Zanoli, "Design and assessment of a pure hydrodynamic microfluidic switch," in *Int'l Conf. on Communications*, 2013, pp. 3165–3169.

[26] A. Grimmer, W. Haselmayr, A. Springer, and R. Wille, "Verification of Networked Labs-on-Chip architectures," in *Design, Automation and Test in Europe*, 2017, pp. 1679–1684.

[27] ——, "A discrete model for Networked Labs-on-Chips: Linking the physical world to design automation," in *Design Automation Conference*, 2017.

[28] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Design, Automation and Test in Europe*, 2006, pp. 1–6.

[29] 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 3-D deferred decision making technique," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 8, pp. 1151–1162, 2013.

[30] D. Grissom and P. Brisk, "Fast online synthesis of generally programmable digital microfluidic biochips," in *Int'l Conf. on Hardware/Software Codesign and System Synthesis*, 2012, pp. 413–422.

[31] E. Maftei, P. Pop, and J. Madsen, "Tabu search-based synthesis of digital microfluidic biochips with dynamically reconfigurable non-rectangular devices," *Design Automation for Embedded Systems*, vol. 14, no. 3, pp. 287–307, 2010.

[32] A. J. Ricketts, K. Irick, N. Vijaykrishnan, and M. J. Irwin, "Priority scheduling in digital microfluidics-based biochips," in *Design, Automation and Test in Europe*, 2006, pp. 329–334.

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

[34] P. Pop, I. E. Araci, and K. Chakrabarty, "Continuous-flow biochips: Technology, physical-design methods, and testing," *Journal on Design & Test*, vol. 32, no. 6, pp. 8–19, 2015.

[35] E. D. Leo, L. Galluccio, A. Lombardo, and G. Morabito, "On the feasibility of using microfluidic technologies for communications in Labs-on-a-Chip," in *Int'l Conf. on Communications*, 2012, pp. 2526–2530.

[36] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*. Amsterdam, NL: IOS Press, Feb. 2009.

[37] H. Gu, M. H. Duits, and F. Mugele, "Droplets formation and merging in two-phase flow microfluidics," *Journal of Molecular Sciences*, vol. 12, no. 4, pp. 2572–2597, 2011.

[38] Y.-C. Tan, J. S. Fisher, A. I. Lee, V. Cristini, and A. P. Lee, "Design of microfluidic channel geometries for the control of droplet volume, chemical concentration, and sorting," *Journal on Lab on a Chip*, vol. 4, no. 4, pp. 292–298, 2004.

[39] D. Link, S. L. Anna, D. Weitz, and H. Stone, "Geometrically mediated breakup of drops in microfluidic devices," *Physical Review Letters*, vol. 92, no. 5, p. 054503, 2004.

[40] J. Köhler, T. Henkel, A. Grodrian, T. Kirner, M. Roth, K. Martin, and J. Metze, "Digital reaction technology by micro segmented flowcomponents, concepts and applications," *Chemical Engineering Journal*, vol. 101, no. 1, pp. 201–216, 2004.

[41] Y.-C. Tan, Y. L. Ho, and A. P. Lee, "Droplet coalescence by geometrically mediated flow in microfluidic channels," *Journal of Microfluidics and Nanofluidics*, vol. 3, no. 4, pp. 495–499, 2007.

[42] W. Wang, C. Yang, and C. M. Li, "On-demand microfluidic droplet trapping and fusion for on-chip static droplet assays," *Journal on Lab on a Chip*, vol. 9, no. 11, pp. 1504–1506, 2009.

[43] W. Haselmayr, A. Biral, A. Grimmer, A. Zanella, A. Springer, and R. Wille, "Addressing multiple nodes in Networked Labs-on-Chips without payload re-injection," in *Int'l Conf. on Communications*, 2017.

[44] L. M. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for Construction and Analysis of Systems*, 2008, pp. 337–340.

[45] M. F. Schmidt, "Microfluidic flow-based biochips," https://sites.google.com/site/mlsibiochips/, 2012.

[46] D. C. Duffy, J. C. McDonald, O. J. Schueller, and G. M. Whitesides, "Rapid prototyping of microfluidic systems in poly (dimethylsiloxane)," *Journal of Analytical Chemistry*, vol. 70, no. 23, pp. 4974–4984, 1998.