BioViz: An Interactive Visualization Engine for the Design of Digital Microfluidic Biochips

Jannis Stoppe[1,2] Oliver Keszöcze[1,2] Maximilian Luenert[2] Robert Wille[3] Rolf Drechsler[1,2]
[1]Cyber-Physical Systems, DFKI GmbH, Bibliothekstr. 5, 28359 Bremen, Germany
[2]Group of Computer Architecture, University of Bremen, Bibliothekstr. 5, 28359 Bremen, Germany
[3]Institute for Integrated Circuits, Johannes Kepler University Linz, Altenbergerstraße 69, 4040 Linz, Austria
{jstoppe|keszocze|malu}@informatik.uni-bremen.de robert.wille@jku.at drechsler@uni-bremen.de

***Abstract*—In order to shorten the required time for the analysis of medical substances, Digital Microfluidic Biochips (DMFBs) have been suggested. They allow for handling small amounts of samples and reagents on a circuit board and, thus, automatically execute medical experiments that are usually conducted manually in laboratories. However, there are various challenges in the design of DMFBs. Issues such as routing and layouting are complex and currently being investigated by various researchers and engineers. Although first automatic solutions assist them, the obtained results are usually provided in a complex and non-intuitive fashion. This makes the utilization and evaluation of existing approaches for DMFB design a tedious task. In this paper, we present a solution for this problem by proposing a visualization scheme for DMFBs that explicitly addresses these problems. To this end, a grammar is proposed which allows the description of resulting designs and their properties. The contributions of this work allow for a design methodology which is easy to use and provides a hassle-free environment for the involved researchers and engineers.**

I. Introduction

Clinical experiments and diagnostic assays can take extraordinary amounts of time. Especially repetitive, tedious tasks are still largely conducted by manual labor. Processes require all kinds of different steps which usually are performed in dedicated machines: fluids (e.g. samples or reagents) are mixed, heated up, analyzed, separated etc., according to detailed instructions that are executed often hundreds of times to counter statistical outliers [1].

Digital Microfluidic Biochips (*DMFBs* or *biochips*, for short) are one approach to increase the automation for certain laboratory tasks. A grid of electrically actuatable cells is used to move small droplets of the required fluids across a small area – providing certain functionalities in particular areas (such as cells that can be heated or that can analyze the fluids).

This allows technologies from the EDA industry (which are cheap when mass-produced) to be used in a context that would greatly benefit from the according automation – making biochips an emerging technology that promises to be a key driver in the automation of diagnostics and biological experiments in general. However, several problems still need to be solved. Designing the experiment in order for it to be run on a DMFB is a core issue, including steps such as routing (see e.g. [2], [3]) and placement (see e.g. [4], [5], [6]).

Developing automatic solutions (i.e. algorithms) for these design steps requires engineers to test different setups, compare the results and debug their algorithms. Solutions, while being technically correct, can still observe negative aspects such as the following:

- *Unnecessary movements:* A droplet moves around, e.g. in circles, instead of waiting on a single cell until it is needed again.
- *Unnecessary cell usage:* Cells are used often, leading to degradation due to aging effects, even though different routes for droplets could have been chosen using the same amount of time steps.
- *Unnecessarily complex control logic:* The choice of a route leads to a complex control logic to drive the droplet while a different route taking the same amount of time steps may allow for a simpler logic.

These aspects are difficult to spot without being able to visually inspect the design.

In this work, we present a dedicated biochip visualization solution that helps designers to explore DMFB designs and supports scientists in evaluating algorithms for certain design aspects such as routing. To assist the storage and exchange of biochip designs (e.g. droplet movements, layouts), we additionally present a simple file format. The tool's benefits are illustrated using representative case studies. BioViz is available online at at http://www.informatik.uni-bremen.de/agra/bioviz/.

II. Background

Digital microfluidic biochips have been proposed to automate laboratory procedures. Many experiments in domains such as biochemistry or molecular biology are conducted manually – relying on both, expensive equipment and manpower to be conducted. Many of these experiments can be performed not only automatically but also using much less space and potentially less expensive equipment using DMFBs.

A. The Structure of DMFBs

DMFBs consist of two-dimensional electrical *grids* which are controlled by electrodes and their electrical *actuations*. These generate an electric field that allows the actuated grid element(s) or *cells* to attract and hold *droplets*, i.e. discrete portions of liquids. When the electrodes are turned on and off over time on neighboring cells, droplets can be moved across the grid (this principle is called *electrowetting-on-dielectric* [7]). This results in a platform that can be used to expose laboratory liquids such as blood or urine to several operations such as mixing, analyzing or heating. Using these structures, laboratory experiments can be carried out automatically on the grid instead of manually on traditional laboratory equipment.

The available operations are realized using *modules* that may correspond to both, elements that are physically built into the system and ones that are realized virtually via droplet movement. There are several types of physical modules.

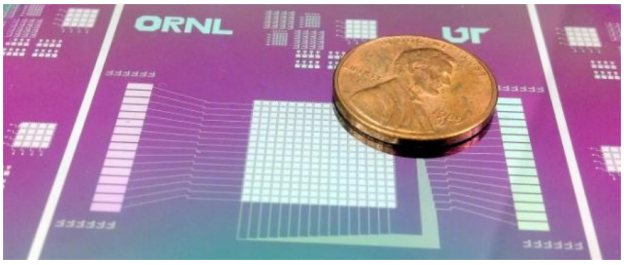


Fig. 1: A microfluidic biochip as presented in [8].

- *Dispensers* are providing a way to add new droplets to the grid. Liquids that are used in the experiments need to be stored in *reservoirs* next to the grid. The cells that connect these reservoirs to the grid are the ones where the droplets first "appear" before they are moved to the next destinations. Each type of liquid (such as blood, water or urine) needs to be stored separately.
- *Sinks* are the dispensers' counterpart for the disposal of droplets – allowing them to be removed from the grid. This may be needed to e.g. regain free space on the grid for more droplets. Unlike dispensers, sinks do not need to be separated by the types of liquid they handle, as any liquids being processed by them are being discarded anyway.
- *Heaters* allow droplets to be heated for experiments. By adding heating elements below certain cells, droplets placed on these can be heated up, allowing the DMFB to be used for experiments that require the liquids to be heated.
- *Detectors* are used to examine the results of a given experiment. These include all kinds of sensor devices which are usually placed below certain cells. These sensors can be used to e.g. measure the color, volume or other attributes of a droplet that is placed on a given cell.

Virtual modules on the other hand do not require physical characteristics to be added onto the chip. Instead, they can be realized implicitly via the droplets' movement – allowing them to be set up as needed during the system's operation. These virtual modules can also be of various types.

- *Mixers* provide a way to mix liquids. By routing the respective droplets to adjacent cells, they automatically merge into a single, larger droplet.
- *Splitters* on the other hand can split up droplets. If e.g. a droplet becomes too big due to mixing operations, activating cells on opposing sides of a droplet will split it in two parts.

Generally, these modules allow operations known from laboratory workflows to be executed on a DMFB. These modules may be implemented in various ways, e.g. concerning the amount of cells they require on a given grid and the time they need to perform a given operation. The resulting, physical chips are small and can be mass-manufactured inexpensively. Fig. 1 illustrates the size of a DMFB.

Note that the given list of modules is far from being complete. Depending on the experiments to be carried out, a large variety of physical and virtual modules may be developed or taken from module libraries to be used in DMFB design. While this paper focuses on the modules given here, the proposed grammar and visualization are easily extensible, so other modules can be added as required.

Also note that this paper considers a DMFB's behavior to be described in discrete time steps. While droplets are moving across the grid in an analogue fashion, we consider the time it takes for any given droplet to be moved to a neighboring cell to be a single, indivisible step in time.

B. Design Challenges

Generally, determining the droplets' routes is a hard problem. Droplets usually need to find short routes to their respective destinations to keep the processing time low. However, not only the determination of the cell actuations (which move droplets across the grid) is crucial, but also how to connect these cells to the respective controller. To keep the costs low, a design goal is to drive multiple cells by a single pin. Such a pin assignment has to allow for conducting the desired routing while, ideally, not restricting any other movements (see e.g. [2], [9]).

Closely related to the routing is the question of the design's layout: how are the cells supposed to be placed to come up with a device that handles various tasks well? Being able to see and thus directly grasp how certain layouts perform in combination with particular routing algorithms is a crucial requirement for the design process.

As different scenarios require different approaches, there are not only large numbers of routing algorithms available (see e.g. [2], [3], [9]), but they also differ greatly concerning their approaches. In order to create a well-designed appliance, a comparison between these algorithms for the according target designs is a necessity.

However, tools that support the designer in analyzing the obtained results are rare. Biochips are usually debugged using the software they are being simulated with – often with basic (e.g. console) outputs only. The comparison of routing and layouting approaches also is usually done by crunching numbers, as designers lack a specific tool to easily and quickly compare different layouts and routing solutions for DMFBs. Additionally, a crucial part of any design process is communicating the current state and approach e.g. to colleagues or stakeholders, for which lengthy console outputs are far from ideal.

This obvious shortcoming for the development of design methods for biochips is the major motivating factor for this work which focuses on a tool to assist developers in the tasks discussed above. We therefore propose a visualization engine for the design of DMFBs. This encompasses the routing of droplets on a chip, the layout of a chip and the analysis of various properties of the design such as the activation of cells or potential proximity issues during droplet movement.

Note that this visualization explicitly targets algorithmic challenges. Problems that arise from the usage of particular materials, physical realization or other properties are not the focus of this work. Instead, the resulting visualization engine is supposed to assist designers in interpreting their simulation traces.

In order to have an open and coherent interface for such a visualization, a common file exchange format is needed as well. This paper therefore not only details the work on a visualization engine, it also proposes an exchange format in the form of a grammar that can be used as a common foundation for the storage of data about biochip structures and behavioural logs.

III. BioGram – A Dedicated Grammar for DMFB Designs

Even though many researchers work in the field of DMFBs, relatively few publications have the storage format or input languages as a central theme.

Currently, the state-of-the-art method of describing protocols still is the use of notebooks. In 2010, an early attempt of formalizing experiments has been done in [10] where the *BioCoder* language for describing experiments on biochips was proposed. The language's goal is to supply researchers with a means to easily describe their experiments. The language is implemented on top of `C++` and can not be used for visualization.[1]

The *Digital Microfluidic Biochip Static Synthesis Simulator* tool [4] uses a plain text format both for input and output and comes with a visualization for the synthesized experiments. In principle, it is possible to use this tool to visualize results not obtained by the tool itself. Still, the language is very specific (i.e. hard to extend) and the visualization not as responsive as desired – in order to quickly analyze and debug results e.g. of routing algorithms or different layouts, a more user-friendly approach is desired.

Hence, we propose a language to describe synthesis and routing solutions for microfluidic biochips that is both, extendable and easy to use. This resulted in a language whose grammar is shown in Backus Naur form in Fig. 2. It was implemented as an ANTLR [11] grammar and embedded into the proposed visualization tool. In general, the design of this language follows the following principles:

Simplicity and Readability: The language is human-readable. This prevents the use of binary files. Furthermore, the language is easy to understand and learn. In fact, the content of a file is completely comprehensible without having to look at the documentation.

Tool agnosticism: We believe that a good language is easily usable by many researchers without focusing on specific approaches. In the design process, we took great care to cover as many use cases without adding anything too specific. The use cases we support contain grids of different layouts (i.e. also non-rectangular grids as in [12]), temporal blockages (used in e.g., [3]), pin assignments and pin actuations (used in many works on routing on biochips, see e.g. [2]), as well as the placement of modules (as needed in synthesis, see e.g. [4], [5], [6]).

Extendability: While the language supports a fixed set of issues, it can easily be extended by simply defining new blocks. This is possible due to the fact that very few aspects of biochips are encoded in nested structures. The design follows the concept of enclosing lists of certain aspects (i.e. droplet movement) by an opening keyword and a closing keyword. Furthermore, issues such as timing can easily be re-used when adding new descriptions to the language.

Example 1: Consider the routing solution provided in BioGram syntax as shown in Fig. 3a. The resulting visualization is provided in Fig. 3b.

In order to illustrate the extendability of the language, we consider a new type of biochip recently propsed that is referred to as Micro-Electrode-Dot-Array (MEDA); see e.g. [13]. Unlike conventional DMFBs, the MEDA architecture is based on the concept of a sea-of-micro-electrodes with an array of identical basic microfluidic unit components called microelectrode cells. The idea is that droplets and cells do not

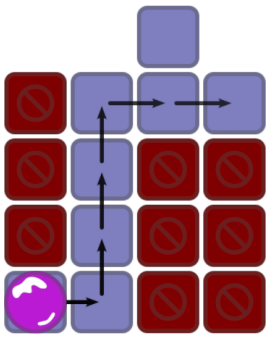
```
grammar ::= ⟨grid⟩ | ⟨blockages⟩ | ⟨nets⟩ | ⟨routes⟩ | ⟨meda_nets⟩
          | ⟨meda_routes⟩ | ⟨modules⟩ | ⟨droplets⟩ | ⟨fluids⟩ |
          ⟨pin_related⟩
    grid ::= 'grid' { ⟨position⟩ ⟨position⟩ } 'end'
blockages ::= 'blockages' { ⟨position⟩ ⟨position⟩ [ ⟨timing⟩ ] }
          'end'
    nets ::= 'nets' { ⟨source⟩ { ',' ⟨source⟩ } '->' ⟨target⟩ } 'end'
  source ::= ⟨DropletID⟩ ⟨position⟩
  target ::= ⟨position⟩
  routes ::= 'routes' { ⟨DropletID⟩ [ ⟨start_time⟩ ] ⟨position⟩ }
          'end'
pin_related ::= ⟨pin_ass⟩ | ⟨pin_acts⟩ | ⟨cell_acts⟩
 pin_ass ::= 'pin assignments' { ⟨position⟩ ⟨PinID⟩ } 'end'
pin_acts ::= 'pin actuations' { ⟨PinID⟩ ':' { ⟨actuation⟩ } } 'end'
cell_acts ::= 'cell actuations' {⟨position⟩ ':' {⟨actuation⟩} } 'end'
actuation ::= '1' | '0' | 'X'
 modules ::= ⟨mixers⟩ | ⟨detectors⟩ | ⟨dispensers⟩ | ⟨sinks⟩
  mixers ::= 'mixers' { ⟨DropletID⟩ ⟨time_range⟩ ⟨position⟩
          ⟨position⟩ } 'end'
time_range ::= '[' ⟨Int⟩ '-' ⟨Int⟩ ']'
detectors ::= 'detectors' { ⟨position⟩ [ ⟨spec⟩ ] } 'end'
    spec ::= ⟨Duration⟩ [ ⟨FluidID⟩ ]
   sinks ::= 'sinks' { ⟨ioport⟩ } 'end'
dispensers ::= 'dispensers' { [ ⟨FluidID⟩ ] ⟨ioport⟩ } 'end'
  ioport ::= ⟨position⟩ ⟨Direction⟩
 droplets ::= 'droplets' { ⟨DropletID⟩ ⟨FluidID⟩ } 'end'
   fluids ::= 'fluids' { ⟨FluidID⟩ ⟨Description⟩ } 'end'
position ::= '(' ⟨Int⟩ ',' ⟨Int⟩ ')'
  timing ::= '(' ⟨Int⟩ ',' (⟨Int⟩ | '*') ')'
```

Fig. 2: Extended Backus Naur Form for the BioGram grammar

```
grid                    blockages
  (1,1) (4,4)             (1,2) (1,4)
  (3,5) (3,5)             (3,1) (4,3)
end                     end

routes
  1 (1,1) (2,1) (2,2) (2,3) (2,4) (3,4) (4,4)
end
```

(a) Example routing solution provided in BioGram



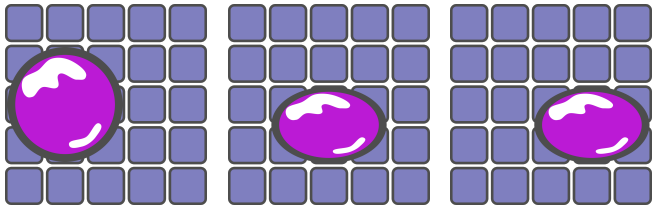
(b) Solution from Fig. 3a visualized in BioViz

Fig. 3: Example grammar and corresponding visualization.

1. BioCoder does provide the users with a textual representation for the experiment that consists of a combination of strings the user can annotate to every single operation.

```
grid
(1,1) (5,5)
end
meda routes
1 ((1,2), (3,4)) ((2,2), (4,3)) ((3,2), (5,3))
end
```

(a) Example MEDA description in BioGram



(b) Visualization of the distinct states from the description given in Fig. 4a

Fig. 4: Extension to MEDA biochips.

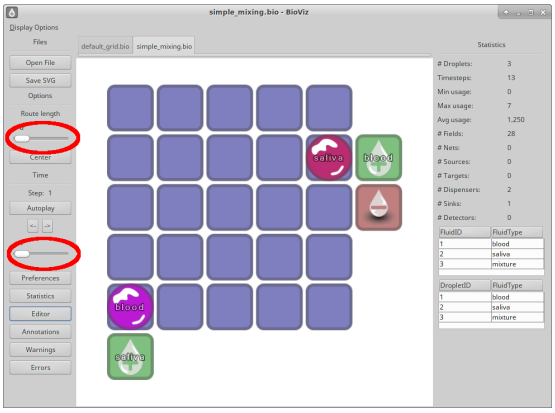


Fig. 5: Visualization of a simple biochip. User interface elements show visualization controls on the left, the visualization itself at the center, statistics on the right and open files on the top of the window.

have a one-to-one relationship any more. A droplet may cover multiple cells, change its size and even move diagonally. BioGram supports MEDA biochips by simply adding a 'meda' in front of the nets and routes and replacing the positions within the nets/routes by location ::= '(' ⟨position⟩ ',' ⟨position⟩ ')'. This has been left out in Fig. 2 to increase the readability.

Example 2: Consider the MEDA droplet shape changing and movement provided in BioGram syntax as shown in Fig. 4a. The resulting visualization is provided in Fig. 4b.

The character '#' begins a comment that ends at the end of the line. These comments are for users reading the BioGram file itself. A comment starting with '#!' is treated as an annotation that will be displayed by the BioViz tool that is introduced in the next section.

IV. An Interactive Visualization Engine

When designing biochips, both structural and behavioral attributes need to be addressed. The structural features are provided by the hardware: the cells that are responsible for moving the droplets, the pins that activate these cells, the available dispensers, detectors and other features all describe the structure of the biochip. The behavior of the chip on the other hand is defined by how the system's attributes change over time, such as the movement of the droplets across the hardware or the activation and deactivation of certain areas on the board, such as scanning cells or similar attributes.

A visualization that illustrates these attributes should increase the designer's ability to inspect a system's behavior. While the structural features (such as the cell layout) can easily be illustrated using static images, the behavioral attributes require a more dynamic approach. While classic hardware visualization tools have traditionally used static illustration methods even for timed features (such as waveforms for signals that can be drawn and printed), the multi-dimensionality of the behavior of these systems prohibits such an approach. Even the simple case of a single droplet that moves across a two-dimensional field needs more dimensions to be displayed as its coordinates already require two dimensions to be illustrated – *time* thus cannot simply be used as one of the coordinates (like e.g. done for waveforms) without resorting to drawing three-dimensionally (as done in [4]). We decided against this approach in order to keep the user experience simple.

The core idea is therefore to provide a dynamic visualization that lets the designer to

- easily scroll through time in order to quickly see a particular state of the design or to
- aggregate different states in a single, comprehensive view if such an overview is required.

This means that a core requirement of a visualization system for biochips is interactiveness. Designers should be able to quickly browse through different aspects of their design and to easily draw connections between different views.

Fig. 5 shows the visualization application with an example system that consists of a 5×5 square grid, two dispensers, two droplets and a single sink on the right below one of the dispensers. The user controls on the left of the window show two important sliders (highlighted in red). The lower one lets the system advance through time in a simple step-through manner. Fig. 6 illustrates this behavior, showing consecutive states of the chip as they would appear to the designer. In addition to the ability to show particular states, an arbitrary amount of previous and consecutive states can be aggregated as well using the upper slider. Fig. 7 shows this mode of operation, taking the second timestep of the simulation and overlaying the previous and consecutive positions of the droplets by adding arrows to the display.

Fig. 8 illustrates how the cells' actuations are displayed using the same paradigm. The visualization can switch between showing distinct states of the design or aggregating several of them to enable designers to quickly see an overview of certain properties.

For the grid, several different cell types are currently embedded into the visualization: blockage, detector, source, sink, start and target (see Fig. 9). More types could easily be added, but as these are the types that are used in current benchmarks, they form a valid base for the visualization to support most use cases.

Another important feature when dealing with designs that change certain properties over time is to enable designers to see how elements behave. When simply displaying discrete states, it may be hard to correctly see how the elements map over time. We therefore provide smooth transitions between

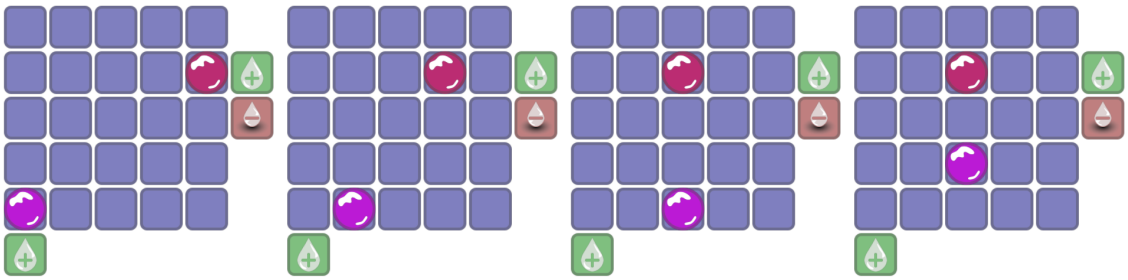


Fig. 6: The designer can step through the discrete timesteps of the design, allowing him to quickly see routing properties and check what happens at which state of the simulation.

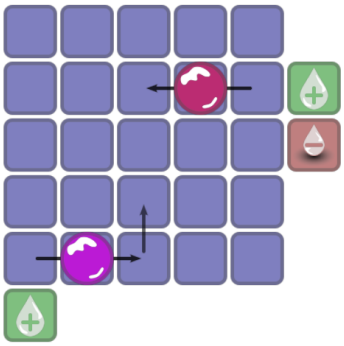


Fig. 7: Routes can be overlaid in addition to displaying particular state of the design.

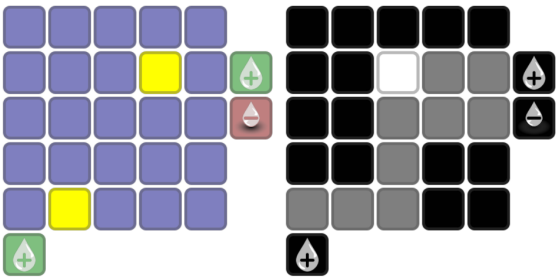


Fig. 8: Left: Information can be shown for the current state – droplets are hidden to emphasize actuation, indicated via yellow-marked cells. Right: Aggregated information – amount of cell actuation on a color scale from black to white.

these states, allowing designers to better understand how the transitions between these states work. Fig. 10 illustrates this transition for droplet movement. However, the same principle applies to e.g. cell colors and viewport shifts. Thus, the designer is constantly being aware of how the states of the design change.

As the designer may freely move and zoom around the chip, he may move the virtual camera far away to get a broad overview of the circuit. In order to still provide a usable representation of the system, the visualization switches to a lower level of detail once the parts otherwise become too small to recognize properly. Fig. 11 illustrates how, at a certain zoom level, the individual elements are reduced to simple square boxes that merely hold the color information. This provides the designer with data from the simulation traces while at the same time avoiding display issues down to the point where individual parts are merely one pixel in size.

Fig. 9: Currently supported cell types: default, blockage, detector, source, sink, start, target

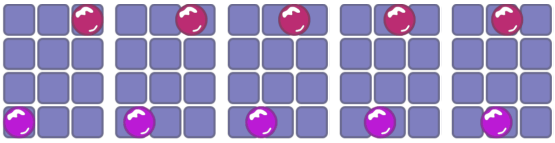


Fig. 10: States are smoothly transitioned to better illustrate the transitions from one system state to the next

BioViz further supports directly editing the loaded file and showing its provided annotations. To simplify debugging, it may show potential warnings (e.g. droplet paths that are not physically realizable) and parsing errors. The statistics panel on the right can be turned off.

As biochips are currently an emerging technology and still subject to fundamental research, designers still need to communicate the properties of their design in printed form. While this, by definition, does not allow any animated properties to illustrate the design properties, it still is an important factor in the development of a given circuit. The implementation supports exporting the current state as an *.svg* file, i.e. as a vector graphics file that can be easily embedded into websites or converted to pdf for e.g. LaTeX documents, as seen in Fig. 12.

The visualization prototype was built using OpenGL. Instead of providing static illustrations of the system, the core requirement of a responsive, dynamic visualization makes the utilization of such a graphics framework an obvious choice to be able to implement a more dynamic view that can be used to smoothly view changes over time and switch between different views.

As a core framework, the *libgdx* library was used as a wrapper. The remaining logic was implemented using Java, allowing the visualization tool to be used on all major operating systems.

The visualization was tested with examples from [14], [15], [16] as well as several new designs. It can handle large layouts and still provide a smooth user experience – allowing designers to interactively inspect their simulation traces and easily see if their algorithms behave as expected.

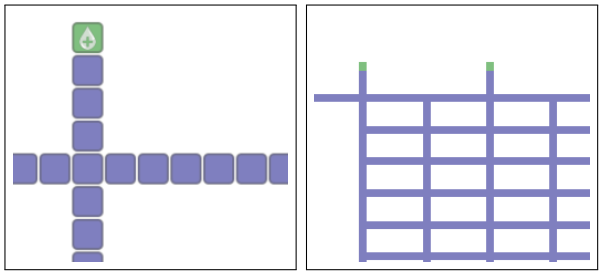


Fig. 11: Detail is omitted when zooming out to provide a better overview

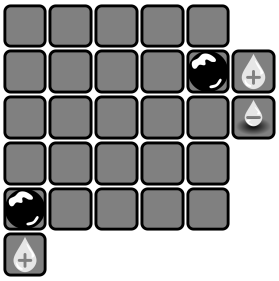


Fig. 12: The visualization state may be exported as an SVG file, allowing easy processing for e.g. print media (such as turning the outcome black and white).

V. Summary

We have presented an easily usable, smooth, interactive visualization engine for microfluidic biochips and a grammar to interface with it.

The grammar is both, easy to use and extend – allowing designers to not only use a common foundation when developing in the biochip domain but also expand the given representation in whatever way they need it. As the grammar was developed with its application in mind, translating it to automated parsers is an easy task – as shown by the implementation in ANTLR, which is currently providing the visualization engine with a parser to read the underlying files. The given grammar could thus be used as a foundation for other works as well, providing a standardized way to store and exchange data about a DMFB design.

Several results of routing algorithms and both, existing and new layouts and have been translated into this grammar – illustrating that it supports the features that are currently being used in the design of biochips.

The visualization itself is a cross-platform application that provides designers with a smooth, interactive view on their systems – something that has not been done before. Running on Java and OpenGL, it does not depend on third party tools and should thus "just run" on most systems. It allows designers to quickly and easily get an overview of both, their system's behavior and its layout – allowing them to quickly evaluate the results of the synthesis and routing processes.

The proposed set of grammar and visualization thus assists designers in the development of microfluidic biochips, enabling them to quickly grasp the properties of how a system behaves and draw the according conclusions for their design processes.

BioViz is available online at http://www.informatik.uni-bremen.de/agra/bioviz/.

VI. Future Work

The visualization itself is currently smooth and provides an interactive way of browsing existing simulation traces, but does not itself control any other tool that assists in the design process. Interactivity could also mean, however, that the problems themselves are e.g. defined from within the visualization – giving the designer the ability to interactively design a biochip and define core constraints for the following simulation. In that case, the visualization itself would become less of a purely visual assistant and more of a visual design tool – enabling designers to use it as a graphical editor.

The tool could also be equipped with a plugin system – allowing designers to not only adapt the grammar (which then needs to be hard-coded into the tool) but also the tool itself. This would allow designers to more easily alter the given technology to their needs – allowing them to more quickly come up with visualizations that match their specific use case.

VII. Acknowledgements

The research reported here was supported by the German Research Foundation (DFG) under the subproject P02 "Heuristic, Statistical and Analytical Experimental Design" of the Collaborative Research Center SFB 1232 "From colored states to evolutionary structural materials" and the Reinhart Koselleck project under contract no. DR 287/23-1.

References

[1] Max Halperin, Eugene Rogot, Joan Gurian, and Fred Ederer. Sample sizes for medical trials with special reference to long-term therapy. *Journal of chronic diseases*, 21(1):13–24, 1968.

[2] Tsung-Wei Huang and Tsung-Yi Ho. A Two-Stage ILP-Based Droplet Routing Algorithm for Pin-Constrained Digital Microfluidic Biochips. In *International Symposium on Physical Design*, pages 201–208, 2010.

[3] Oliver Keszocze, Robert Wille, and Rolf Drechsler. Exact routing for digital microfluidic biochips with temporary blockages. In *Int'l Conf. on CAD*, pages 405–410, 2014.

[4] Daniel Grissom, Kenneth O'Neal, Benjamin Preciado, Hiral Patel, Robert Doherty, Nick Liao, and Philip Brisk. A Digital Microfluidic Biochip Synthesis Framework. In *VLSI of System-on-Chip*, pages 177–182, 2012.

[5] Oliver Keszocze, Robert Wille, Tsung-Yi Ho, and Rolf Drechsler. Exact One-pass Synthesis of Digital Microfluidic Biochips. In *Design Automation Conf.*, pages 142:1–142:6, 2014.

[6] Fei Su and Krishnendu Chakrabarty. Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips. In *Design Automation Conf.*, pages 825–830, 2005.

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

[8] Daniel Grissom. Design of topologies for interpreting assays on digital microfluidic biochips. 2014.

[9] Oliver Keszocze, Robert Wille, Krishnendu Chakrabarty, and Rolf Drechsler. A General and Exact Routing Methodology for Digital Microfluidic Biochips. In *Int'l Conf. on CAD*, pages 874–881, 2015.

[10] Vaishnavi Ananthanarayanan and William Thies. Biocoder: A programming language for standardizing and automating biology protocols. *Journal of biological engineering*, 4(1):1–13, 2010.

[11] Terence J. Parr and Russell W. Quong. Antlr: A predicated-ll (k) parser generator. *Software: Practice and Experience*, 25(7):789–810, 1995.

[12] Yang Zhao and Krishnendu Chakrabarty. Simultaneous Optimization of Droplet Routing and Control-Pin Mapping to Electrodes in Digital Microfluidic Biochips. *IEEE Trans. on CAD*, 31(2):242–254, 2012.

[13] Gary Wang, Daniel Teng, and S.-K. Fan. Digital microfluidic operations on micro-electrode dot array architecture. *IET nanobiotechnology*, 5(4):152–160, 2011.

[14] Yang Zhao, Krishnendu Chakrabarty, and Bhargab B. Bhattacharya. Testing of low-cost digital microfluidic biochips with non-regular array layouts. *Journal of Electronic Testing*, 28(2):243–255, 2012.

[15] Ping-Hung Yuh, Chia-Lin Yang, and Yao-Wen Chang. Bioroute: A network-flow based routing algorithm for digital microfluidic biochips. In *Int'l Conf. on CAD*, pages 752–757, 2007.

[16] Minsik Cho and David Z Pan. A high-performance droplet routing algorithm for digital microfluidic biochips. *IEEE Trans. on CAD*, 27(10):1714–1724, 2008.