

Biochip Simulator

Flow-Based Microfluidic Biochip Simulation

Morten Foged Schmidt

Submitted to the department of Informatics and Mathematical Modeling on July 31, 2012
at the Technical University of Denmark.

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673 reception@imm.dtu.dk
www.imm.dtu.dk

I SUMMARY

Microfluidic biochips are miniaturized devices that are able to integrate, on-chip, the functionalities needed to execute biochemical analysis applications. The development of microfluidic biochips is in a hasty development phase and the need for simulation and design tools is increasing. The design and architecture of biochips changes all the time and designers need to know how their biochip designs will work before physically building them. Flow-based microfluidic biochips are one kind of biochip, where simulation could help find new ways of constructing the biochip architecture and scheduling the experiments on biochips. To the best of our knowledge no simulation method has been developed so far.

This thesis presents a method to simulate the logic of flow-based microfluidic biochips. The presented simulation method could be characterized as a workflow, from creation of biochip architectures and biochemical application to the results in the form of useful formats and views. Since, no flow-based biochip simulation tools are available at the moment, this project includes a working implementation supporting the findings in this thesis. The tool is called Biochip Simulator.

II PREFACE

This thesis documents my M.Sc. in Computer Science and Engineering degree at the Technical University of Denmark (DTU). The thesis was performed in collaboration with the Embedded System Engineering (ESE) section at the department of Informatics and Mathematical Modelling (IMM). The thesis and implementation was done during a six-month period from February-July 2012.

The outcome includes an implementation of a flow-based microfluidic biochip simulator, called Biochip Simulator. A user guide and the documentation needed to use the simulator have been published to a web page, <https://sites.google.com/site/biochipsimulator/>. Video recordings showing evaluation cases are available on the website, together with a video showing the main features in the simulator as well.

III ACKNOWLEDGEMENTS

I thank professor Jan Madsen for supervising this thesis and research. I would also like to thank Ph.D. student Wajid Hassan Minhass and professor Paul Pop for their supervision on this thesis and research. Wajid Hassan Minhass has given me essential feedback as a flow-based microfluidic biochip architecture synthesis- and schedule designer. He met with me on a weekly basis to discuss problems and solutions. The three supervisors have participated in and scheduled meetings with me through the whole thesis period; here the progress of the thesis have been discussed and guidance have been provided.

Table of Contents

I	SUMMARY	4
II	PREFACE	5
III	ACKNOWLEDGEMENTS	6
CHAPTER 1	INTRODUCTION	11
SECTION 1.1	RELATED WORK	11
SECTION 1.2	OBJECTIVES & MOTIVATION	13
SECTION 1.3	CONTRIBUTION	14
CHAPTER 2	SYSTEM MODEL	16
SECTION 2.1	BIOCHIP ARCHITECTURE MODEL	16
SECTION 2.2	BIOCHEMICAL APPLICATION MODEL	23
SECTION 2.3	SCHEDULERS	24
SECTION 2.4	SUMMARY	25
CHAPTER 3	MOTIVATIONAL EXAMPLE	26
CHAPTER 4	FLOW-BASED BIOCHIP SIMULATION	29
SECTION 4.1	ARCHITECTURE DESIGN	30
SECTION 4.2	APPLICATION DESIGN	35
SECTION 4.3	VISUALIZE SIMULATION	37
SECTION 4.4	CONTROL DATA GENERATION	39
SECTION 4.5	SUMMARY	41
CHAPTER 5	IMPLEMENTATION	43
SECTION 5.1	BIOCHIP ARCHITECTURE DESIGN	45
SECTION 5.2	ARCHITECTURE MODEL	48
SECTION 5.3	BIOCHEMICAL APPLICATION DESIGN	51
SECTION 5.4	VISUALIZE SIMULATION	52
SECTION 5.5	DATA CONTROL GENERATION	54
SECTION 5.6	SUMMARY	56
CHAPTER 6	EXPERIMENTAL EVALUATION	57
CHAPTER 7	FUTURE WORK	64
CHAPTER 8	CONCLUSION	67
	REFERENCES	68
APPENDIX A	BIOCHIP SIMULATOR – COMPONENTS	70
APPENDIX B	BIOCHIP SIMULATOR – FILE FORMATS	79
APPENDIX C	BIOCHIP SIMULATOR – USER GUIDE	90

List of Figures

FIGURE 1 - MICADO SCREENSHOT. RED DOTS ARE PRESSURE SOURCES. BLUE DOTS ARE FLUIDIC INPUT SOURCES. FLUIDIC CHANNELS ARE IN-BETWEEN THE BLUE DOTS. (AMIN , MICADO, 2008)	12
FIGURE 2 - SIMULATING IDEA	13
FIGURE 3 - FUNCTION OF MICRO VALVES	17
FIGURE 4 - PHOTO OF A REAL MICRO VALVE	17
FIGURE 5 – SWITCH COMPONENT	18
FIGURE 6 – GRAPHICAL REPRESENTATION OF A MIXER	18
FIGURE 7 - PHOTO OF FLUID MIXER IN A FLOW-BASED MICROFLUIDIC BIOCHIP (VIDEO FROM (THIES, PROGRAMMABLE MICROFLUIDIC BIOCHIPS, 2007)) .	19
FIGURE 8 - STORAGE COMPONENT	19
FIGURE 9 - BIOCHIP ARCHITECTURE	20
FIGURE 10 - USE OF A FLOW PATH TABLE	22
FIGURE 11 – FLOW PATH SET INFORMATION	22
FIGURE 12 - EXAMPLE OF A BIOCHEMICAL APPLICATION	23
FIGURE 13 – SCHEDULER	24
FIGURE 14 - SCHEDULE PRODUCED FOR THE BIOCHIP ARCHITECTURE AND BIOCHEMICAL APPLICATION	25
FIGURE 15 - EXAMPLE OF BIOCHIP ARCHITECTURE	26
FIGURE 16 - BIOCHEMICAL APPLICATION GRAPH EXAMPLE	26
FIGURE 17 - ILLUSTRATIVE SCHEDULE EXAMPLE	27
FIGURE 18 – DETAILED SIMULATION WORKFLOW	29
FIGURE 19 - BIOCHIP SIMULATOR SCREENSHOT. BIOCHIP DRAWING BOARD...	30
FIGURE 20 - BIOCHIP SIMULATOR SCREENSHOT. PROPERTY VIEW.	31
FIGURE 21 - BIOCHIP SIMULATOR SCREENSHOT. COMPONENT LIBRARY.	32
FIGURE 22 – BIOCHIP SIMULATOR SCREENSHOT. COMPONENT PROPERTIES ..	32
FIGURE 23 - BIOCHIP SIMULATOR SCREENSHOT. CONNECTION CHANNEL CREATION (BLUE POINTS ARE CONNECTION POINTS)	33
FIGURE 24 - BIOCHIP SIMULATOR SCREENSHOT. FINALIZED BIOCHIP ARCHITECTURE	33
FIGURE 25 - BIOCHIP SIMULATOR SCREENSHOT. FLOW EXECUTION TIMES	34
FIGURE 26 - BIOCHIP SIMULATOR SCREENSHOT. TRANSPORT CALCULATION WHERE COMPONENT OUT1 IS MOVED	35
FIGURE 27 - BIOCHIP SIMULATOR SCREENSHOT. EXAMPLE FLOW PATH SETS .	35
FIGURE 28 - BIOCHIP SIMULATOR SCREENSHOT. APPLICATION DRAWING BOARD.	36
FIGURE 29 - BIOCHIP SIMULATOR SCREENSHOTS. OPERATION PROPERTIES...	36
FIGURE 30 - BIOCHIP SIMULATOR SCREENSHOT. FINALIZED BIOCHEMICAL APPLICATION.	37
FIGURE 31 - BIOCHIP SIMULATOR SCREENSHOT. A SIMULATION WHERE MULTIPLE OPERATIONS ARE EXECUTED ON A BIOCHIP ARCHITECTURE. THE CONTROL LAYER (RED) AND SINK LAYER (BLACK) IS SHOWN AS EXTRA INFORMATION.	38
FIGURE 32 - BIOCHIP SIMULATOR SCREENSHOT. A BIOCHEMICAL APPLICATION, WHERE AN OPERATION IS HIGHLIGHTED BECAUSE THE EXECUTION HAS BEEN STARTED.	39

FIGURE 33 - BIOCHIP SIMULATOR SCREENSHOT, MICRO VALVE CONTROL TABLE	40
FIGURE 34 - BIOCHIP SIMULATOR SCREENSHOT, PATH COLLISION,	40
FIGURE 35 - BIOCHIP SIMULATOR SCREENSHOT, FLOW COLLISION ERROR VIEW,	41
FIGURE 36 - BIOCHIP SIMULATOR SCREENSHOT, SIMULATION LOG,	41
FIGURE 37 - HIGH-LEVEL UML DESIGN DIAGRAM	43
FIGURE 38 - ARCHITECTURE CLASS	45
FIGURE 39 - LAYERS IN THE STORAGE COMPONENT IN THE SIMULATOR	46
FIGURE 40 - NEW COMPONENT CLASS	47
FIGURE 41 - TRANSPORT TIME CALCULATION FOR CONNECTION CHANNEL	50
FIGURE 42 - FLOWPATHSET CLASS	51
FIGURE 43 - LOG CLASS	54
FIGURE 44 - VALVES CLASS.....	55
FIGURE 45 - COLLISION CLASS	56
FIGURE 46 - HANDMADE BIOCHIP ARCHITECTURE DESIGN	58
FIGURE 47 - BIOCHIP SIMULATOR SCREENSHOT, BIOCHIP ARCHITECTURE DESIGN.....	58
FIGURE 48 - BIOCHIP SIMULATOR SCREENSHOT, BIOCHEMICAL APPLICATION DESIGN	58
FIGURE 49 - BIOCHIP SIMULATOR SCREENSHOT, FLOW PATH TABLE	59
FIGURE 50 - BIOCHIP SIMULATOR SCREENSHOTS, A SEQUENCE OF SCREENSHOTS SHOWING THE SIMULATION,	60
FIGURE 51 - BIOCHIP SIMULATOR SCREENSHOT, SIMULATION LOG,	60
FIGURE 52 - BIOCHIP SIMULATOR SCREENSHOT, MICRO VALVE CONTROL DATA TABLE.....	61
FIGURE 53 - BIOCHIP SIMULATOR SCREENSHOT, COLLISION ERRORS,	61
FIGURE 54 - BIOCHIP SIMULATOR SCREENSHOT, FLUID FLOW COLLISION,	62
FIGURE 55 - BIOCHIP SIMULATOR SCREENSHOT, FLUID COLLISION ERROR VIEW	62

List of Tables

TABLE 1 - MIXER CONFIGURATION	18
TABLE 2 - FLOW PATH TABLE	21
TABLE 3 - FLOW PATH TABLE AND ROUTING CONSTRAINTS	27
TABLE 4 - TEST CASES INFORMATION	57
TABLE 5 – SCHEDULER OUTPUT. SCHEDULE OF OPERATIONS TO EXECUTE IN THE BIOCHIP.	59

CHAPTER 1 INTRODUCTION

In recent years it has become interesting to miniaturize chemical and biological instrumentation. The result of this interest has culminated in a concept called "Lab-on-a-chip", also referred to as microfluidic biochip. The general idea is to create a fully integrated chemical or biological lab on a single chip with high throughput, reduced reagent consumption, low cost and automatic control (Cooper, Wentzlaff, Thorsen, Thies, Urbanski, & Amarasinghe, 2004). Miniaturizing the macroscopic chemical and biological processes to a sub-millimeter scale also has advantages like reduced sample volumes, faster biochemical reactions and ultra sensitive detection (Minhass, Pop, & Madsen, MPM11b, 2011).

Microfluidic biochips are perfectly suited to facilitate clinical diagnostics, particularly immediate point-of-care disease diagnoses. Microfluidic biochips can handle existing applications as DNA analysis, enzymatic and proteomic analysis, cancer and stem cell research and automated drug detection. Using microfluidic biochips for food control, biochemical weapons detection or environmental testing are also interesting possibilities (Minhass, Pop, & Madsen, MPM11a, 2011).

As a result of the many biochemical application possibilities, there are many variations of biochips (Mark, Haeberle, Roth, Stetten, & Zengerle, 2010). The many variations contributed to different architectures and designs of biochips. This thesis does research in the area of flow-based microfluidic biochips, in which the circuitry is composed of fluidic channels and is controlled by micro valves.

SECTION 1.1 RELATED WORK

Flow-based biochips become more and more complex and the tools available for biochip development are still in an early stage of development. Recent work has proposed automation techniques for placement and routing problems. A tool addressing design and development problems is Micado. Micado is a Computer-Aided Design (CAD) tool focusing on the design of control layers in biochips, featuring:

- Standard and customizable design rules

- Automatic routing between control valves and punches
- Automatic generation of control instructions and GUI

The tool works as a design tool, but it does not allow simulation and verification of biochip designs. Micado focuses on automated features that can help designers to build the control layers inside biochips. (Amin , Micado, 2008) The tool is very detailed and well suited for physical design of biochip control layers. To illustrate how detailed Micado is, a screenshot from the process of routing the architecture on a biochip is shown in Figure 1.

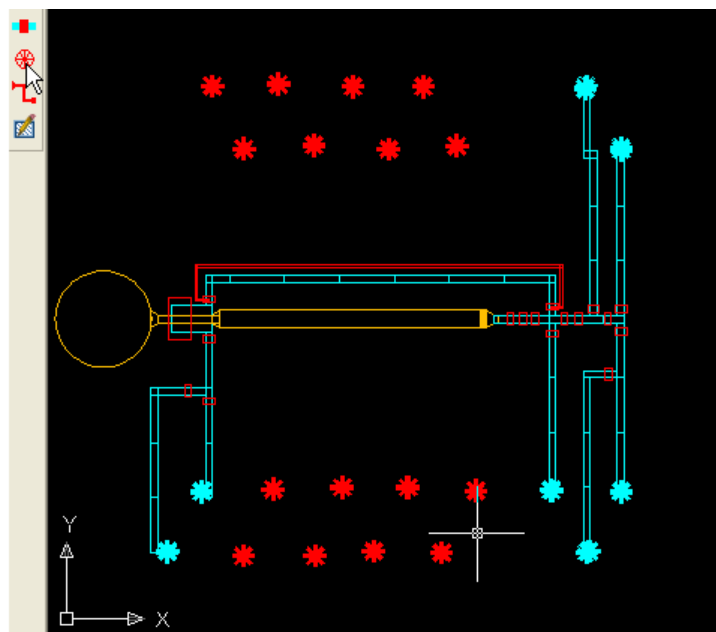


Figure 1 - Micado screenshot. Red dots are pressure sources. Blue dots are fluidic input sources. Fluidic channels are in-between the blue dots. (Amin , Micado, 2008)

The fact that designers will have to manually draw fluidic channels and micro valves within a biochip limits the usefulness, since it is time-consuming, and the ability to verify different chemical applications on a specific biochip architecture is not possible; this is done manually. The design of a biochip having hundreds of micro valves and a larger network of fluidic channels will almost be impossible to design within Micado (Amin, Computer-Aided Design for Multilayer Microfluidic Chips, 2008). Designers are also more likely to make mistakes, since no automated verification method to verify the correctness of a biochip design has been implemented. This manual approach will most likely result in inefficient biochip designs (Minhass, Pop, & Madsen, MPM11a, 2011).

AutoCAD is another tool used in the process of fabricating biochips. The tool is generally used for 2D or 3D designs and is basically an advanced drawing tool. A template file can be loaded into the drawing

tool to enable the designers to choose biochip elements and draw biochip designs. But the designer will have to use manual methods to create the biochip architectures (Getting started AUTOCAD, -), as the template file only provides limited design tools. The manufacturer also provides a set of design rules, which entirely is the designer's responsibility to follow (Basic Design Rules, -). Examples of design rules examples are "minimum spacing between channels" or "minimum spacing between access punches". There are many design rules and the risk of creating errors is increased, when the biochips designs become more complex. The ability to test biochemical applications on the biochip design is not an option; a real physical chip will have to be fabricated (Testing Your Device, -).

The general impression from the tools is that they do not scale with the increasing complexity of today's biochips. They use manual approach to verify the correctness of the designs, and do not have the ability to auto test biochemical application.

SECTION 1.2 OBJECTIVES & MOTIVATION

At the Embedded System Engineering (ESE) section at the department of Informatics and Mathematical Modeling (IMM) located at the Technical University of Denmark (DTU), a scientist team has been working in the aim of optimizing architectural synthesis and the schedules of operations executed on microfluidic flow-based biochips. The team has developed a way to optimize architectures and schedules (Minhass, Pop, & Madsen, MPM11a, 2011) (Minhass, Pop, & Madsen, MPM11b, 2011). But the knowledge of how the optimizations will work is difficult to verify. Simulation is one way to verify the correctness of a biochip architecture and schedule.

The goal of this thesis was to develop and present a simulation method and a working implementation of a simulator for flow-based microfluidic biochip. Figure 2 shows a graphical representation of the goal.

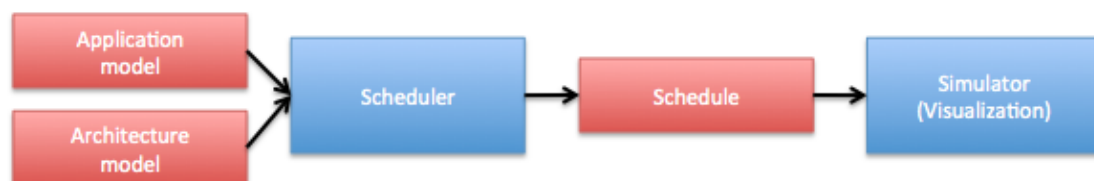


Figure 2 - Simulating idea

A *Scheduler* is a replaceable software component, which is able to receive a chemical or biological *Application Model* and a biochip

Architecture Model. The *Scheduler* generates an optimized schedule of operation to execute on the biochip *Architecture Model*. A generated *Schedule* is loaded into a *Simulator*, which is able to visualize the *Schedule* of operations executed on the biochip *Architecture Model*.

The simulator is intended to be a flexible tool, which enables designers to change the *Scheduler*, *Architecture Model* and chemical or biological *Application Model* and get a graphical representation of the results utilizing the *Simulator*.

The simulation method and tool is designed to help designers and scientists working with optimization. They will be able to test their results without having a real biochip. The tool will help explain situations that may be difficult to explain without graphical representation. It would be possible to test various chemical applications on multiple biochip architectures within a few minutes; this will increase the possibility to choose correct solutions and display the faults in incorrect solutions.

To the best of our knowledge no flow-based microfluidic biochip simulation method and tool has been developed or implemented until now.

SECTION 1.3 CONTRIBUTION

This thesis proposes a method to simulate the logic of flow-based microfluidic biochips. The contributions made within this thesis aim for more than a graphical representation of scheduler results. The developed simulation method could be characterized as a workflow, from creation of biochip architectures and biochemical application to the results in the form of useful formats and views. The following contributions have been made:

- An automated biochip architecture design method has been developed utilizing the functions and elements from physical flow-based microfluidic biochips.
- An algorithm that is able to collect all flow possibilities from biochip architectures to generate architecture models utilized by schedulers.
- An automated biochemical application design method has been developed utilizing the elements from biochemical application sequence graphs.
- A simulation method to visualize schedules of operations on virtual biochip architectures.

- An automated generation of control data from a simulation, which can be tested on physical biochip architectures or used for future optimization and verification of schedules.
- Biochip Simulator, a working implementation supporting the findings from this thesis.

This thesis is organized in 8 chapters. Chapter 2 presents essential information for the work that has been done. The chapter reviews the functions of flow-based microfluidic biochips, biochemical applications and the work that has been performed at DTU. In chapter 3 a motivational example is given to bring insight to the problems related to scheduling and simulation of flow-based microfluidic biochips. Chapter 4 proposes a simulation method and displays the functions within the method. All functions are described at user level utilizing the Biochip Simulator implementation. In Chapter 5 the implementation of the Biochip Simulator will be presented. Chapter 6 contains an evaluation of the work that has been performed. Finally in chapter 7 and 8 a discussion of the future work and a conclusion has been made.

CHAPTER 2 SYSTEM MODEL

This chapter contains information that is relevant for the work that has been done in the aim of simulating flow-based biochips. The sections are essential for the creation of the simulation method. First a presentation of the flow-based biochip architectural model, then a description of the biochemical application model used in flow-based microfluidic biochip development. Finally, a description of schedulers and the work that has been done at DTU is presented.

SECTION 2.1 BIOCHIP ARCHITECTURE MODEL

Biochips are manufactured using "Soft Lithography" technology. Soft lithography is widely used when components measured on the micrometer or nanometer scale are manufactured. The word "Soft" is used because the fabrication method uses elastomeric materials, most notably Polydimethylsiloxane (PDMS). PDMS is a biocompatible, transparent, rubber-like material, which is well suited for mass production and biotechnology (Stanford Microfluidic Foundry, -).

The physical biochip can have multiple layers, but biochips are typically divided into two layers. One layer contains channels where fluids flow; this layer is called the flow layer. The other layer contains channels that form micro valves. These micro valves are able to stop the fluidic flow when pressurized with air. In this way fluids in the flow layer are manipulated using a control layer (Microfluidic valve technology) (Minhass, Pop, & Madsen, MPM11a, 2011).

With several micro valves it is possible to create more complex lab instruments, e.g. mixers, micro pumps, switches or storage (Chou, Unger, & R. Quake, 2001). The micro valve can be considered as the basic building block in the biochip. An illustration of a micro valve is shown in Figure 3.

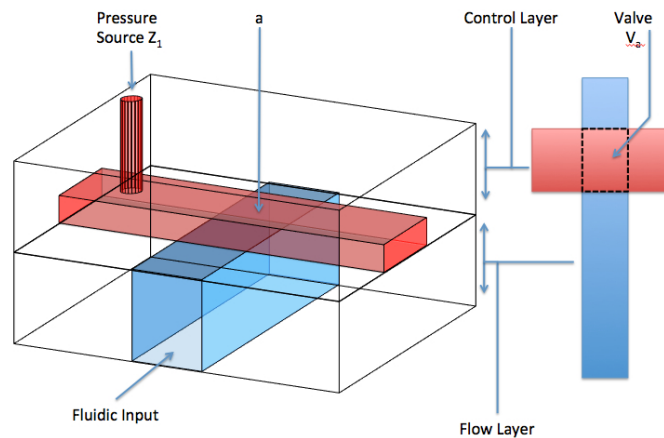


Figure 3 - Function of micro valves

The figure shows a red layer (control layer) and blue layer (flow layer). The control layer is connected to an external pressure source Z_1 . The flow layer is connected to a fluidic input source, which generates a pressure that makes the fluid flow. When the pressure source Z_1 is active, the fluid cannot pass point a . When the pressure source is deactivated, the fluid can pass.

Connections to external source are created by small holes in the biochip gaining access to the layers (flow- and control layer). By placing external tubing's into the connection holes the biochip now has access to fluidic reservoirs or pressure sources (Melin & Quake, 2007). A photo of a micro valve is shown in Figure 4.

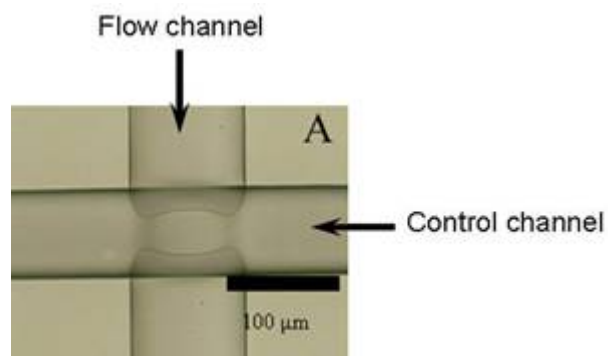


Figure 4 - Photo of a real micro valve

As the photo shows the dimensions of the micro valve is $100 \times 100 \mu\text{m}$. Because of its small size a biochip can accommodate hundreds of micro valves.

The combination of micro valves can create more complex components, such as switches, mixers, storage etc. In Figure 5 a switch is shown.



Figure 5 – Switch Component

The idea of a switch is to direct the flow in the flow layer, as shown in the conceptual view (Figure 5.b) a switch can direct an incoming flow in three new directions. The switch has a schematic view (Figure 5.a) showing that a switch has four micro valves (v_1, v_2, v_3, v_4), which can manipulate the flow direction, depending on the combination of closed valves. The micro valves are controlled by the pressure sources (z_1, z_2, z_3, z_4). For example, if a flow from the top to the right flow channel is performed v_1 and v_4 are open and v_2 and v_3 are closed.

Components created from micro valves normally have a set of phases or states. One example could be a mixer component, which is graphically presented in Figure 6.

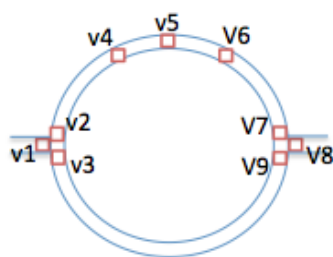


Figure 6 – Graphical representation of a mixer

Phase	v1	v2	v3	v4	v5	v6	v7	v8	v9
ip1	0	0	1	0	0	0	0	0	1
ip2	0	1	0	0	0	0	1	0	0
mix	1	0	0	Mix	Mix	Mix	0	1	0
op1	0	0	1	0	0	0	0	0	1
op2	0	1	0	0	0	0	1	0	0

Table 1 - Mixer Configuration

The mixer component has nine micro valves (v_1-v_9) controlling the component and five phases. Table 1 shows the phases of the mixer, for each phase, valves have to be open (0) or closed (1). The mixer enables a mix of two fluidic samples. If the top channel needs to be filled with a sample, the mixer should be in phase ip1, here v_3 and v_9 are closed and the rest of the valves are open. A new phase should be entered when the bottom channel needs to be filled. The mixer should then change phase to ip2. When the mixer has a sample in each channel (top and bottom), the mixer phase should change to mix, which transforms the mixer into a circular architecture by closing v_1 and v_8 . The mix is performed using the on-chip pump (v_4, v_5, v_6), the pump works by opening and closing the valves, v_4, v_5, v_6 , with a fixed

frequency that pushes the fluid around until it has been mixed. When the mixed samples are needed elsewhere the mixer is emptied by changing phase to op1 or op2 (Chou, Unger, & R. Quake, 2001) (Minhass, Pop, & Madsen, MPM11b, 2011).

An example where a fluid is manipulated by micro valves on a physical biochip is shown in Figure 7; the example includes a mixer component.

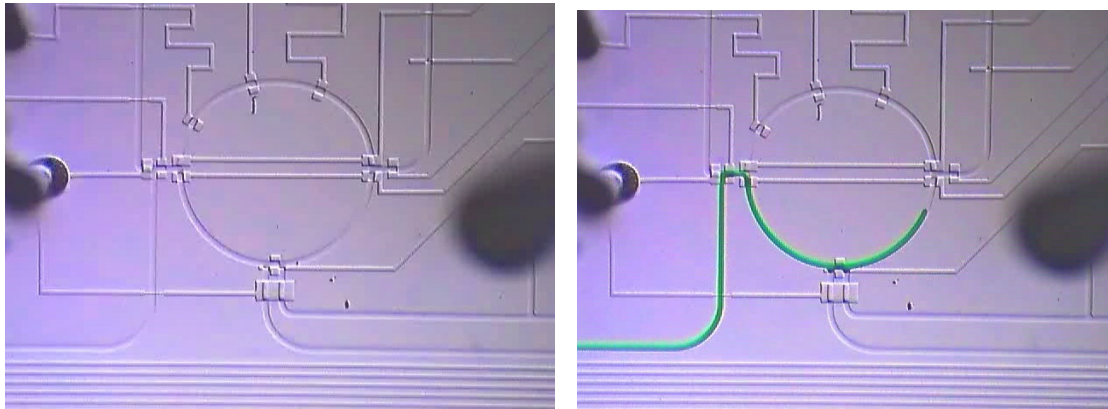


Figure 7 - Photo of fluid mixer in a flow-based microfluidic biochip (Video from (Thies, Programmable Microfluidic Biochips, 2007))

The photos demonstrate how micro valves work in a physical biochip. In the first photo an empty mixer is shown. In the second photo a green fluid flows into the bottom channel of the mixer. The photos show several channels that the fluid could flow into, but micro valves on the biochip control the fluid. This photo example shows the basics of flow-based biochips (Thies, Thi07, 2007).

Another complex component like a storage component can be created with 28 micro valves. The storage component allows eight fluidic samples to be stored. A graphical representation of the component is shown in Figure 8.

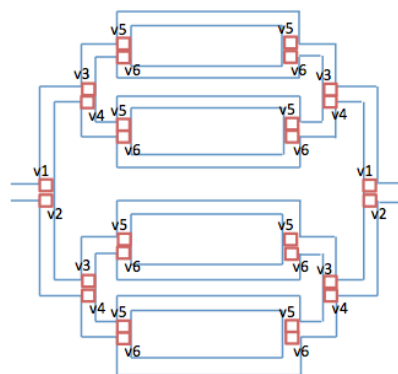


Figure 8 - Storage component

The storage has 16 phases and utilizes multiplexer theory to bring down the number of utilized pressure sources to only six (Amarasinghe, Thorsen, Urbanski, William, & Rhodes, 2005). Closing more valves at the same time does the optimization. An example could be, if store cell one (the top cell) should store a fluidic sample, valves named v2, v4, and v6 should be closed. Closing the valves at the bottom cell, for instance, does not influence a sample in that cell, since no pressure will occur in the cell channel.

The mixer and storage components described here are examples of components with several utilization options. Selecting which storage cell or, in the mixer, which channel (top or bottom) to store a fluidic sample in, is a decision that has to be made and controlled by the control layer.

The existence of components, which cannot be created just by placing micro valves and channels in the biochip, is also a possibility. Filters, heaters and detectors are examples of components that cannot be created from micro valves and channels alone.

To explain how fluidic samples are moved between components, the conceptual biochip shown in Figure 9 is used.

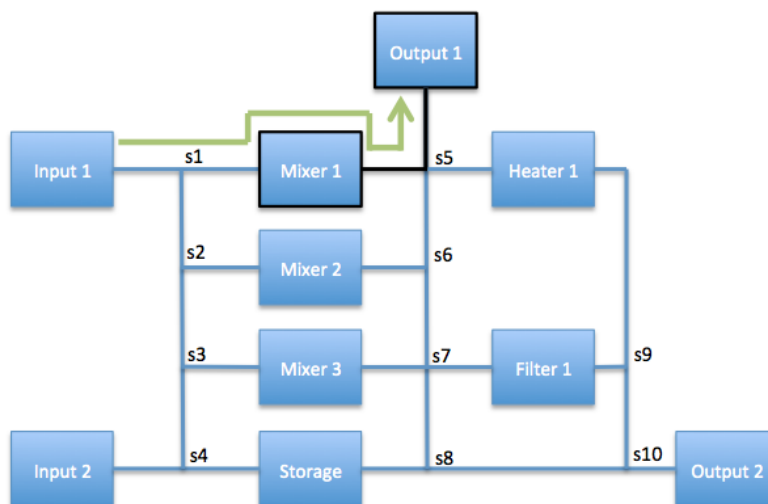


Figure 9 - Biochip Architecture

In flow-based biochips fluidic samples do not occupy the full length between components, fluidic samples occupy a fixed length of the flow channel. The sample length is calculated using a process called *metering*. This process is carried out by placing a fluid between two micro valves with a known distance between them. (Melin & Quake, 2007) To avoid fluidic samples to be split or destroyed, they are moved between components, immersed in a filler fluid (oil or similar) (Minhass, Pop, & Madsen, MPM11b, 2011). The external fluidic input ports are connected to a pump and a filler oil reservoir. To make a fluid sample

move from, e.g., *Mixer1* to *Output1* (black line); *Mixer1* needs to be connected to an input source e.g. *input1*. The channel connection is necessary because a pressure is needed to drive the fluid flow. The destination *Output1* needs to be connected to an output port or sink to absorb the pressure generated from *Input1* (green line). Micro valves available on the biochip will establish a connection between the pressure sources and output port (valves placed in *s1* and *s5*). When a connection is established, a pumping action is created where filler oil flows in the direction of *Output1*. This action makes the fluidic sample move from *Mixer1* to *Output1*. The pumping action stops when the fluidic sample has reached its destination.

The architecture presented in Figure 9 has several flow options. As the figure shows, the components are connected using a network of flow channels. All options can be extracted from the architecture and placed in a flow path table, see Table 2.

ID	Components	Execution time	Routing constraints
F1-1	Input1, s1, Mixer1	2s	F1-2,F2-1,F2-2,F31-1,F31-2
F1-2	Input1, s1, Mixer1	2s	F1-1,F2-1,F2-2,F31-1,F31,2
F2-1	Input1, s1, s2, Mixer2	2,5s	F1-1,F1-2,F2-2
F2-2	Input1, s1, s2, Mixer2	2,5s	F1-1,F1-1,F2-1
...			
F31-1	Mixer1, s5, Output1	1,5s	F1-1,F1-2;F31-2
F31-2	Mixer1, s5, Output1	1,5s	F1-1,F1-2,F31.1

Table 2 - Flow path table

The flow path table is considered as a library of possibilities on the biochip. The *ID* column represents a unique identifier for a fluidic flow in the biochip. Notice that flow paths involving components with more phases have an extra number added to their identification indicating which phase the component should be in. For example, if *F1-1* is performed the fluidic sample will be stored in the top cell of *Mixer1*. The *Components* column represents the components that a fluidic sample will have to pass before it reaches its destination. The *Execution time* column shows how long time the movement will take. The execution time is calculated from a flow rate and the flow channel length. A typical flow rate could be 10 mm/s. The *Routing constraints* column shows the flow paths that cannot be performed at the same time. To demonstrate how the flow path table works, Figure 10 is used.

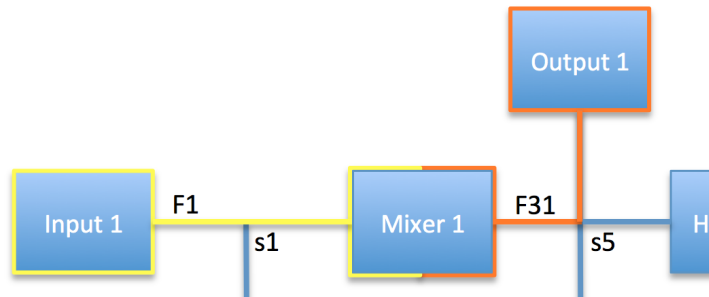


Figure 10 - Use of a flow path table

If a flow from *input1* to *output1* is wanted, the flow path table can be utilized to find the way. First, flow path *F1-1* transports a fluidic sample from *input1* to *mixer1* (yellow line). As the figure shows, the flow path crosses a switch *s1*, where micro valves are active to direct the flow to *Mixer1*. This operation is repeated to get another fluidic sample into *Mixer1*, but this time *F1-2* is used to store the fluid sample in the bottom cell. Then a mix operation is performed, this operation is performed in 2 seconds, which can be considered normal. Finally, the mixed fluid is transported to *output1*, by the use of flow path *F31-1* and *F31-2*. The fluidic movement and operation would take 2×2 seconds for *F1-1* and *F1-2*, 2×1.5 seconds for *F31-1* and *F31-2* and finally 2 seconds for the mix operation, which in total is 9 seconds. The biochip architecture allows multiple operations to be performed at the same time; here routing constraints are used to avoid collisions in pressurized channels. For example, if *F1-1* and *F1-2* are performed at the same time, the fluidic input source, *Input1* is used twice which is not possible.

A complete flow path table contains the following information.

A list of flow path sets which consists of:

- Identification
- Flow Path Components
- Sink Path Components
- Execution time
- Open micro valves
- Closed micro valves
- Routing constraints

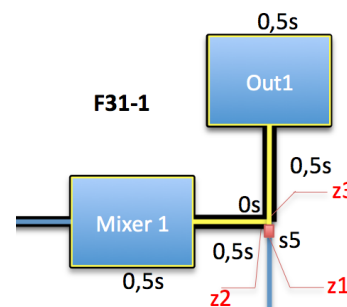


Figure 11 – Flow path set information

A conceptual example of a flow path set is shown in Figure 11. In this case the identifier is *F31-1*, which is unique for this particular flow. "Flow path components" are the components that the fluidic sample passes, *Mixer1-S5-Out1*, placed on the yellow line. The "sink path

components" are all components responsible for the movement of a fluid, also components that the fluidic sample does not pass; the black line marks them. Execution time states the time it takes for a fluid to move from the first component to the last in the flow path component list, in this case it is 2 seconds, calculated as $0,5s(Mixer1) + 0,5s(Mixer-S5) + 0s(s5) + 0,5s(s5-Out1) + 0,5s(Out1) = 2\ seconds$. The micro valve identification is given, in this case $z1$ will be closed and $z3$ and $z2$ are open to direct the flow. The routing constraints identify the flow path sets that cannot be executed at the same time, because their sink path would collide.

The flow path sets presented is the actual biochip architecture model that schedulers use for their calculations of flows and operations on a biochip.

SECTION 2.2 BIOCHEMICAL APPLICATION MODEL

Flow-based biochips are developed to execute biochemical applications. Some of the applications that biochips are able to execute have already been mentioned in the introduction. The biochemical applications used for flow-based microfluidic biochips can be represented using sequence graphs. An application contains relevant information needed for schedulers to schedule the operations on a biochip. A biochemical application example is shown in Figure 12.

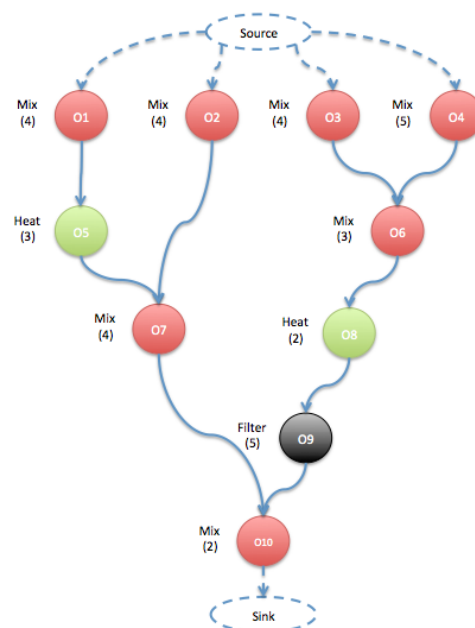


Figure 12 - Example of a biochemical application

The sequence graphs are directed in the sense that all graphs have a starting point (Source) with no predecessors and an endpoint with no successors (O10). All other operation points have predecessor operations needed for an operation to be ready to execute. Some operations need more than one. Each operation has an execution time and a type of operation e.g. mix, heat, or filter. The biochemical application graph contains the following information.

A list of operations which all consist of:

- A list of predecessor operations
- A type of operation
- An execution time

SECTION 2.3 SCHEDULERS

The work that has been performed at DTU is an optimization method, which enables designers to get an optimized schedule of operations to execute on a flow-based microfluidic biochip. The optimization method produces good quality solutions, but the solutions are not optimal in the sense that scheduling is an NP-complete problem, so heuristics like List-Scheduling are used. Figure 13 shows a graphical representation of the scheduler function.

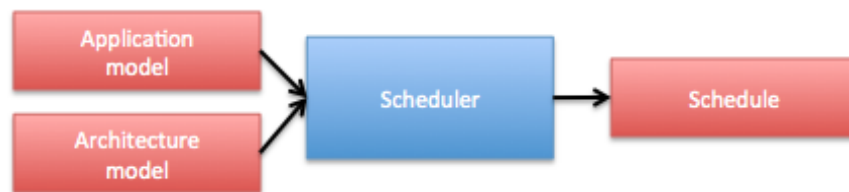


Figure 13 – Scheduler

The *Scheduler* receives a biochip *Architecture Model* and a biochemical *Application Model*. The *Scheduler* then optimizes the flows and operations to be executed on the biochip *Architecture Model*, and produces an optimized *Schedule*.

The scheduler that has been designed utilizes List-Scheduling. Scheduling in general denotes the concept of assigning jobs to devices, CPU's, in this case biochip components. List scheduling implies the process of assigning priorities to operations and listing them after their priorities. In each step operations are evaluated according to an urgency criteria. The urgency criteria are calculated utilizing information from a biochemical application model, which is described in Section 2.2. The urgency criteria are calculated as the length in time

from an operation to the end node in the graph. An operation is only ready to be executed when its predecessors have finished their execution. Since operations on a biochip cannot be stopped during execution, the scheduling is done as non-preemptive scheduling. There are also other parameters influencing the scheduling. The scheduler has to take the number of component resources into account. For example if only one mixer is available on the biochip, the schedule cannot execute two mix operations at the same time (Cottet, Delacroix, Kaiser, & Mammeri, 2002) (Minhass, Pop, & Madsen, MPM11a, 2011).

A schedule produced by the scheduler reflects the Gantt chart shown in Figure 14 where operations and flows to be performed are represented as rectangles.

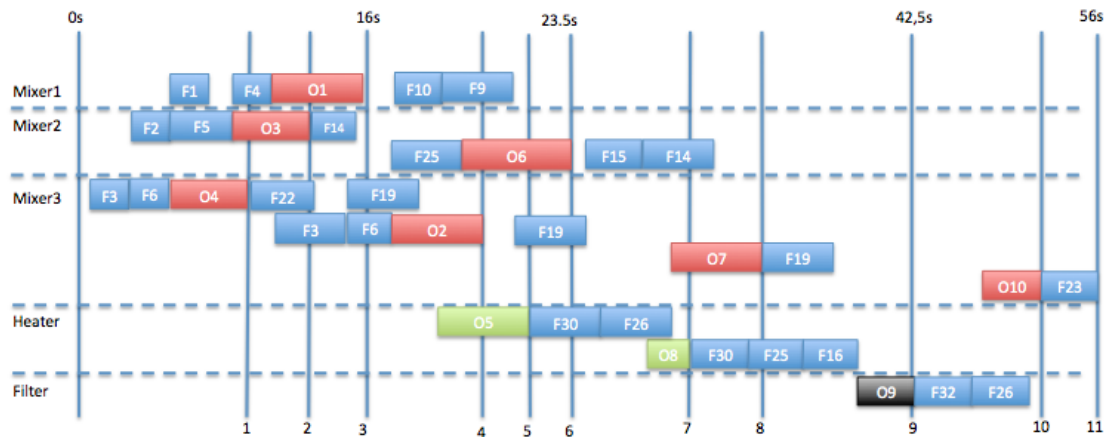


Figure 14 - Schedule produced for the biochip architecture and biochemical application.

The length of the rectangles represents the execution time. All blue rectangles are fluidic sample movement between components. The red, green and black rectangles are operations executed in their respective components, shown to the left (Mixer1, Mixer2, Mixer3, Heater and Filter) (Minhass, Pop, & Madsen, MPM11a, 2011).

SECTION 2.4 SUMMARY

Knowledge of the basic functions in flow-based microfluidic biochips has been presented. The functions of micro valves and fluidic sample movement in channels have been described. A presentation of a flow path table containing the flow possibilities on a biochip has also been provided. A description of the biochemical application model has been introduced. Finally, an explanation of the work that has been performed at DTU with a scheduler has been described.

CHAPTER 3 MOTIVATIONAL EXAMPLE

In order to illustrate the idea for flow-based biochip simulation, this section presents an example where a biochemical application modeled as a sequence graph, is scheduled into operations to be executed on a biochip. In this example, the conceptual biochip architecture in Figure 15 and the biochemical application graph shown in Figure 16 is used. Both models are typical examples taken from the DTU scheduler development (Minhass, Pop, & Madsen, MPM11b, 2011).

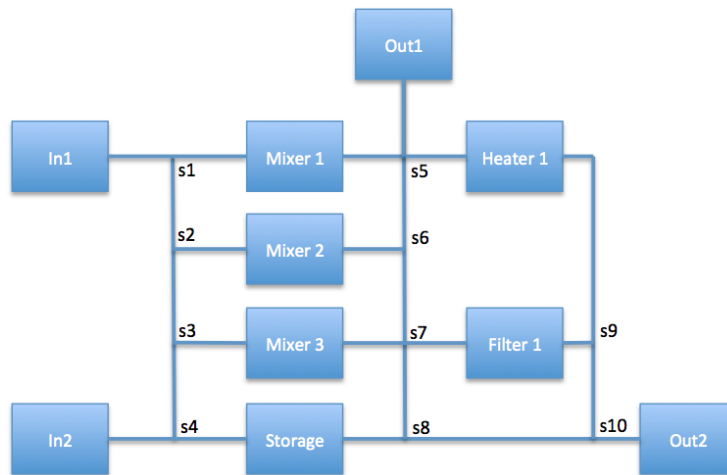


Figure 15 - Example of biochip architecture

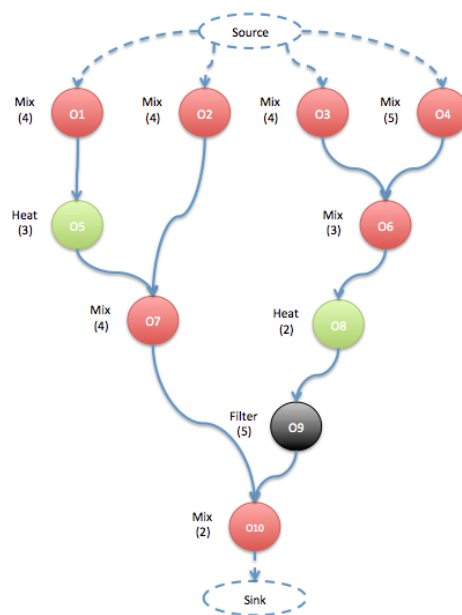


Figure 16 - Biochemical application graph example

The application graph shows that four inputs from sources are needed. The operations have operation types (mix, heat, filter) and execution times, e.g., operation O1, performs a mix operation in 4 seconds. The application finishes when O10 have been executed and the fluidic sample in O10 has been moved to a sink.

Before the scheduling of operations can be created, a flow path table, containing all flow possibilities is needed. All flow possibilities from the biochip architecture are shown in Table 3.

<p>F1: (In1,S1,Mixer1) 2s F2: (In1,S1,S2,Mixer2) 2.5s F3: (In1,S1,S2,S3,Mixer3) 3s F4: (In2,S4,S3,S2,S1,Mixer1) 3.5s F5: (In2,S4,S3,S2,Mixer2) 3s F6: (In2,S4,S3,Mixer3) 2.5s F7: (In1,S1,S2,S3,S4,Storage) 3.5s F8: (In2,S4,Storage) 2s F9: (Mixer1,S5,Out2) 2s F10: (Mixer1,S5,Heater1) 2s F11: (Mixer1,S5,S6,S7,Filter1) 3s F12: (Mixer1,S5,S6,S7,S8,Storage) 3.5s F13: (Mixer1,S5,S6,S7,S8,S10,Out1) 4s F14: (Mixer2,S6,S5,Out2) 2.5s F15: (Mixer2,S6,S5,Heater1) 2.5s F16: (Mixer2,S6,S7,Filter1) 2.5s F17: (Mixer2,S6,S7,S8,Storage) 3s</p>	<p>F18: (Mixer2,S6,S7,S8,S10,Out1) 3.5s F19: (Mixer3,S7,S6,S5,Out2) 3s F20: (Mixer3,S7,S6,S5,Heater1) 3s F21: (Mixer3,S7,Filter1) 2s F22: (Mixer3,S7,S8,Storage) 2.5s F23: (Mixer3,S7,S8,S8,Out1) 3s F24: (Storage,S4,S3,S2,S1,Mixer1) 3.5s F25: (Storage,S4,S3,S2,Mixer2) 3s F26: (Storage,S4,S3,Mixer3) 2.5s F27: (Storage,S8,S7,S6,S5,Heater1) 3.5s F28: (Storage,S8,S7,Filter1) 2.5s F29: (Storage,S8,S10,Out1) 2.5s F30: (Heater1,S9,S10,S8,Storage) 3s F31: (Heater1,S9,S10,Out1) 2.5s F32: (Filter1,S9,S10,Out1) 2.5s F33: (Filter1,S9,S10,Out1) 2.5s</p>	<p>Routing Constraints: F1: F2 ∨ F3 ∨ F4 ∨ F7 ∨ F24 F2: F1 ∨ F3 ∨ F4 ∨ F5 ∨ F7 ∨ F24 ∨ F25 F3: F1 ∨ F2 ∨ F4 ∨ F5 ∨ F6 ∨ F7 ∨ F24 ∨ F25 ∨ F26 F4: F1 ∨ F2 ∨ F3 ∨ F5 ∨ F6 ∨ F7 ∨ F8 ∨ F24 ∨ F25 ∨ F26 F5: F2 ∨ F3 ∨ F4 ∨ F6 ∨ F7 ∨ F8 ∨ F24 ∨ F25 ∨ F26 ∨ F27 F6: F3 ∨ F4 ∨ F5 ∨ F7 ∨ F8 ∨ F24 ∨ F25 ∨ F26 F7: F1 ∨ F2 ∨ F3 ∨ F4 ∨ F5 ∨ F6 ∨ F8 ∨ F24 ∨ F25 ∨ F26 ... F33: F13 ∨ F18 ∨ F23 ∨ F29 ∨ F30 ∨ F31 ∨ F32</p>
---	---	--

Table 3 - Flow path table and routing constraints

Using the DTU scheduler a schedule is produced for the biochemical application and the biochip architecture. Figure 17 shows the schedule.

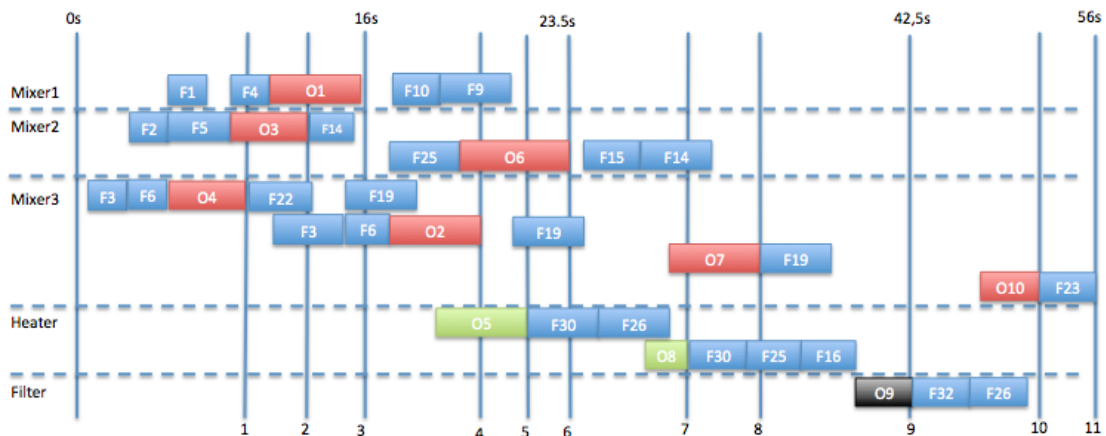


Figure 17 - Illustrative schedule example

The schedule is presented as a Gantt chart, where the operations and fluid flow are represented as rectangles. The length of the rectangles represents the execution time. All blue rectangles are fluidic movement between components. The red, green and black are operations.

Using simulation, it would be possible to verify that this schedule works as expected, if the fluids are moving between components as expected

and if components contain the fluids that are expected. One example could be the mix operation; a mix operation needs two fluids to operate correctly. From the schedule above it is difficult to see if there are two fluids in the mixers when the operations are executed. At the moment no simulation method is able to present and verify this schedule on a virtual flow-based microfluidic biochip.

CHAPTER 4 FLOW-BASED BIOCHIP SIMULATION

Based on the system model presented in Chapter 2 and the motivational example in Chapter 3, this chapter proposes a method to simulate the logic of flow-based microfluidic biochips. The motivational example and the biochip simulator are utilized to explain the simulation method at user level. A graphical representation of the simulation method is presented in Figure 18.

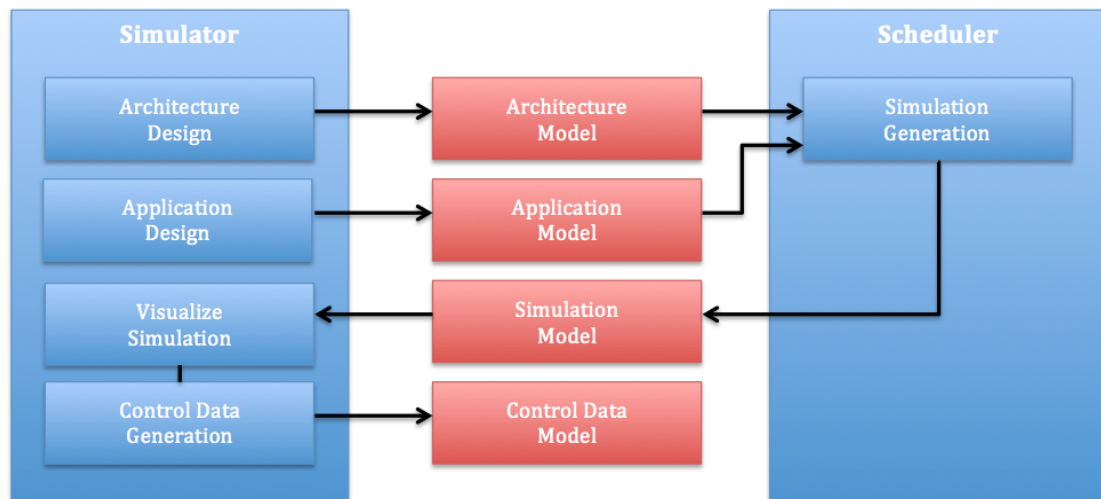


Figure 18 – Detailed simulation workflow

The simulation method is presented as a workflow where the *Simulator* and *Scheduler* exchange information. The aim was to develop a simulation method that involves more than visualization of schedules on a virtual biochip. This method includes biochip architecture and biochemical application design. Besides the visualization of a simulation, a method to generate control data for a simulation is also included.

First, the *Scheduler* should be able to load a biochip *Architecture Model* and a biochemical *Application Model*. These models are designed utilizing the *Simulator*. Next, the *Scheduler* generates a *Simulation Model*, based on the loaded architecture and biochemical models. The *Simulation Model* is loaded into the *Simulator*, which visualizes the *Simulation Model* of the biochemical *Application Model* executed on the biochip *Architecture Model*. While simulating, a *Control Data Model* is generated which contains relevant simulation data that enables further verification of the *Simulation Model*. The data model also contains useful

information that can be fed to a biochip controller to automatically execute the application on a biochip. Designers can also optimize their solution further utilizing the data.

SECTION 4.1 ARCHITECTURE DESIGN

To simulate a flow-based biochip, an architecture design has to be created. The biochip architecture design is performed using a drawing board shown to the right and tools shown to the left in Figure 19.

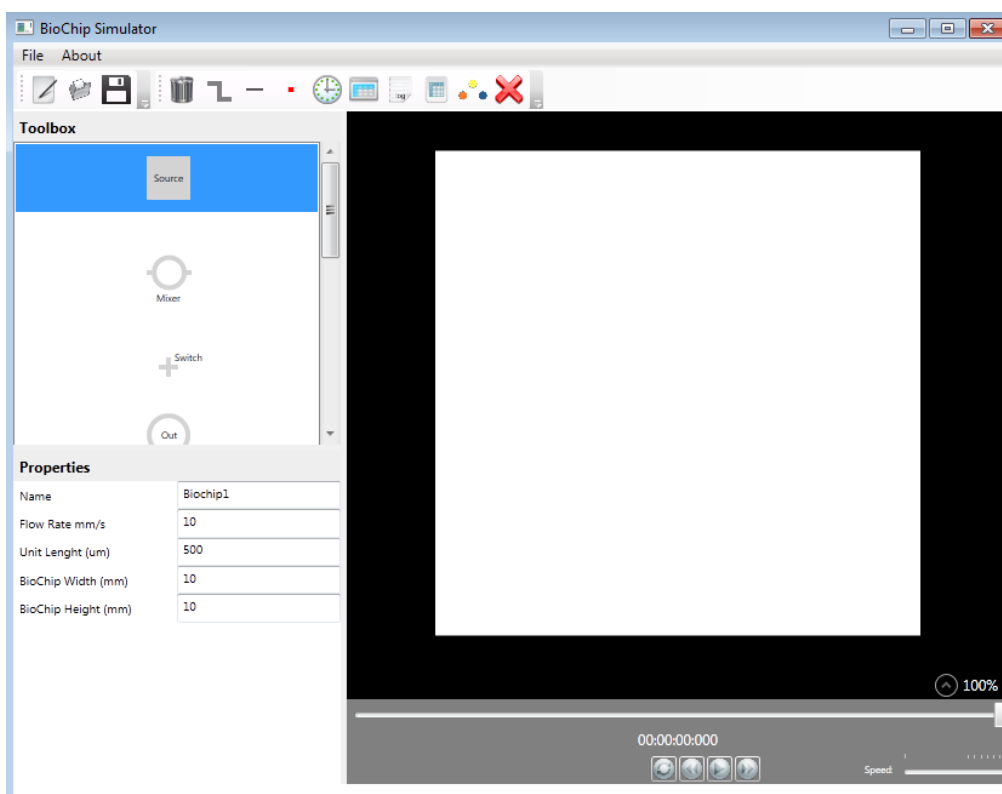
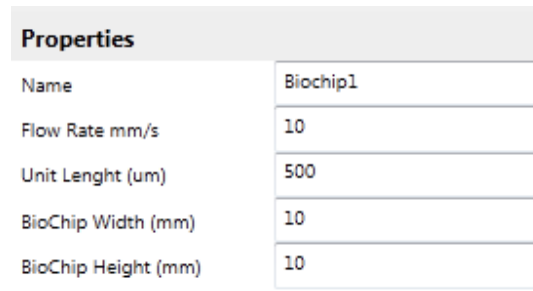


Figure 19 - Biochip Simulator screenshot. Biochip drawing board.

A biochip has parameters, which influence the fluid flow in biochips. An important parameter is the speed of fluid flows on the biochip, called flow rate. Since the simulation method is used for a logical representation, the architecture design has one flow rate that is used for all fluid flows. It is possible to change the fluid flow rate, but the flow rate applies to all flows in the biochip. One could argue that the fluids have different density, but this is considered out of scope, since the focus in this thesis is the logic of flow-based biochips. Flow-based biochips utilize the *metering* method described in Section 2.1, which introduces a *Unit length* parameter. The unit length parameter represents the fluid samples that the biochip operates with. A biochip also has dimensions; the architecture design is based on two-layered biochips (one flow- and control layer), which means that a width and

length parameter is needed. The biochip parameters are customizable and can be set using a property view, as shown in Figure 20.



Properties	
Name	Biochip1
Flow Rate mm/s	10
Unit Length (um)	500
BioChip Width (mm)	10
BioChip Height (mm)	10

Figure 20 - Biochip Simulator screenshot. Property View.

This biochip has a width and length of 10 mm. The components operate with fluidic samples with a unit length of 500 μm . The flow rate is 10 mm/second and the chip is named Biochip1.

SECTION 4.1.1 BIOCHIP COMPONENTS

The architecture design is based on flow-based components, which can be placed at any position on the biochip. The architecture design method contains eight components in a component library. All components are described in detail in Appendix A. The list of components are shown below:

- Mixer – Mixes two fluidic samples
- Storage – Stores up to eight fluidic samples
- Filter – Filters a fluidic sample
- Heater – Heats a fluidic sample
- Detector – Detection process of a fluidic sample
- Input source – Enables an input of fluidic samples
- Output – Enables an output of fluidic samples
- Switch – Directs a fluidic sample flow

The components are added to the drawing board using drag and drop. All available components are accessible in a toolbox as shown in Figure 21.

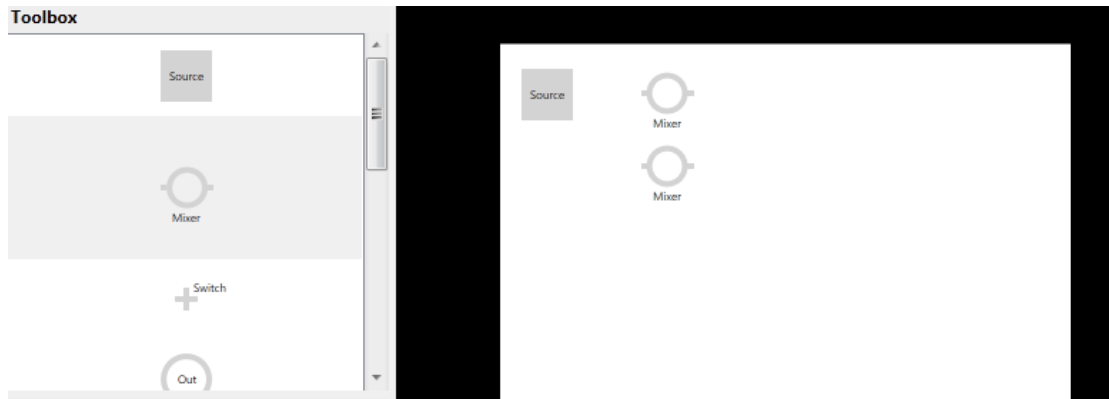
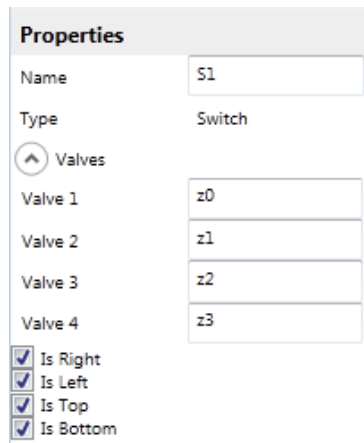


Figure 21 - Biochip Simulator screenshot. Component library.

By dragging the components from the toolbox to the biochip drawing board the addition of biochip components is possible. All component properties can be set using a property view, as shown in Figure 22.



(a) Properties for the component, S1



(b) Graphical representation

Figure 22 – Biochip Simulator screenshot. Component properties

The property view (Figure 22.a) allows the designer to customize the component (Figure 22.b), by selecting a name for the component, selecting the pressure sources that its micro valves are connected to and special properties for the individual component - in this example the directions that the switch can direct a fluidic sample flow. Another example could be the input source component where the fluidic samples can be customized, by fluid name and color.

SECTION 4.1.2 BIOCHIP COMPONENT CONNECTION

The movement of fluid samples in a biochip depends on a network of channels between the components. Therefore it must be possible to define this network. A connection channel must know which components it connects. Since some components, e.g., a switch can have multiple connections, the connection channel must have a connection point associated with a component. The component

connection point can have three functions. It can either be an input point, output point or two-way point. The point type depends on the component; for example, input source components cannot have fluidic flow entering the component, the input source component only contains connection points that allow fluidic flow leaving the component. The connection channels are created using a drawing feature as shown in Figure 23.

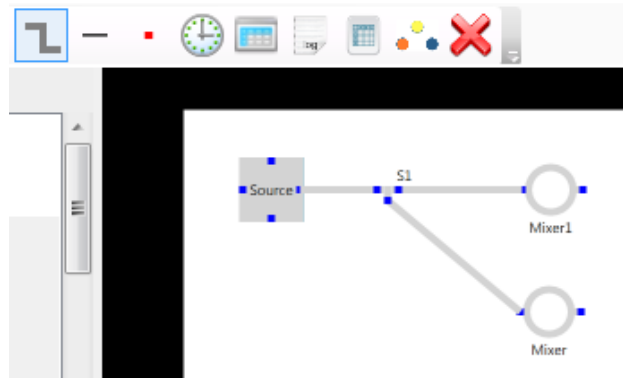


Figure 23 - Biochip Simulator screenshot. Connection channel creation (blue points are connection points).

The drawing feature allows the designer to click a connection point (blue dots) and then drag the connection channel to another connection point. When all components and the connection network have been created, the biochip architecture design is finished. A fully developed virtual biochip could look as shown in Figure 24, this is also the architecture used in the motivational example in Chapter 3.

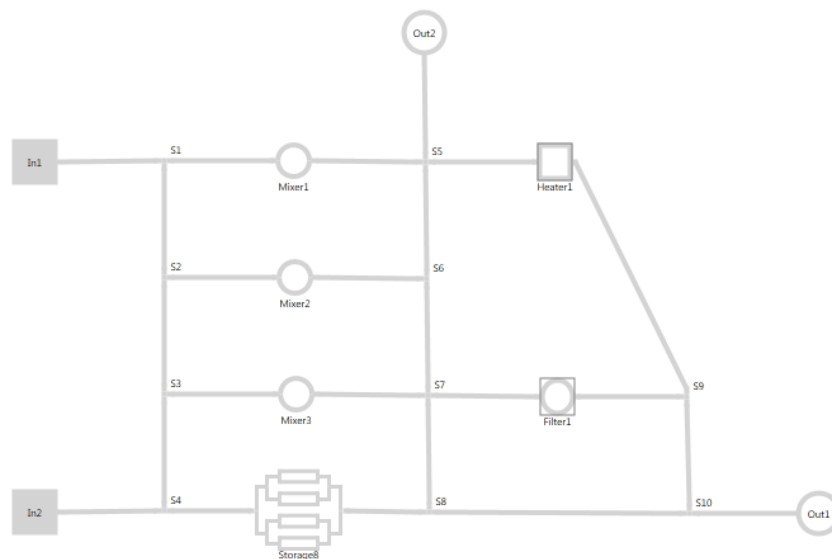


Figure 24 - Biochip Simulator screenshot. Finalized biochip architecture

SECTION 4.1.3 EXECUTION TIME

All fluidic flows in the biochip have an execution time. These execution times influence the produced schedule for a biochip architecture design; it is therefore possible to see all flow execution times for a biochip architecture design. See Figure 25.

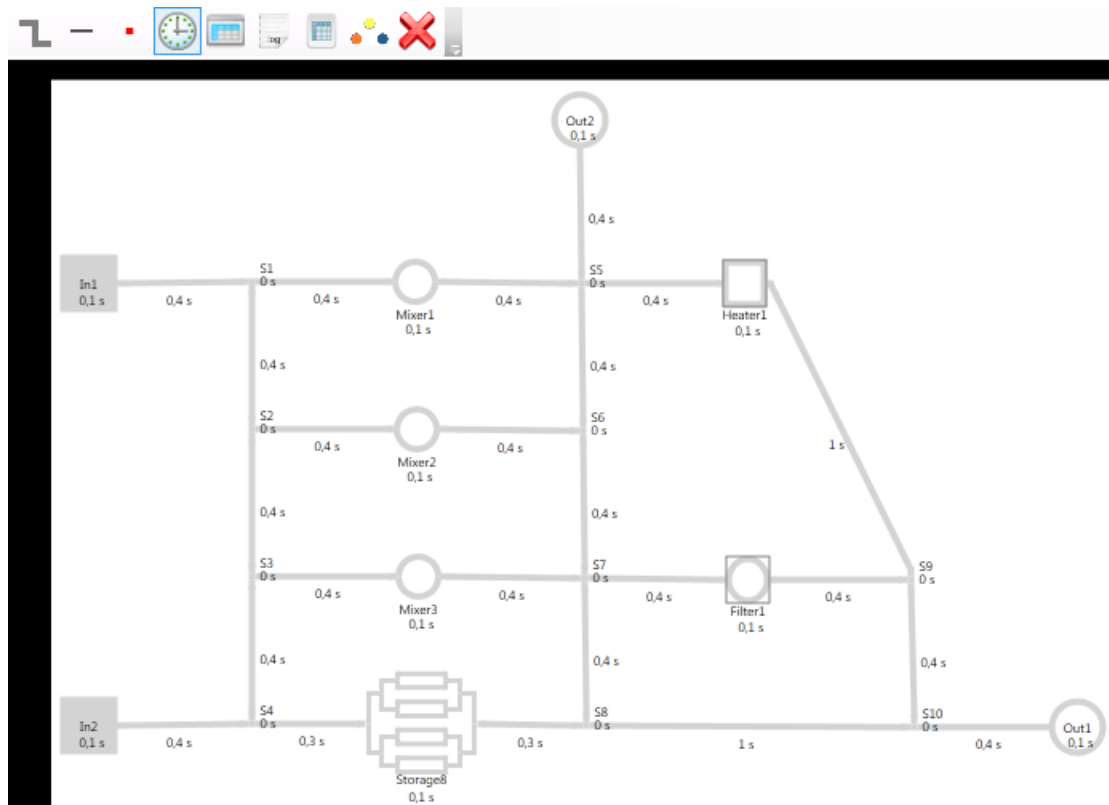


Figure 25 - Biochip Simulator screenshot. Flow execution times

For example, the execution time for a flow between *Heater1* and *S9* takes 1 second. In this architecture the fluidic flow takes 0.1 second in *Heater1*. Because of the high flow rate 10 mm/second the execution time in switches are close to 0 seconds.

The drag feature allows the designer to drag components and get a new flow execution time. It is also possible to change the biochip parameters, flow rate or unit length, and quickly see the consequences. An example is shown in Figure 26

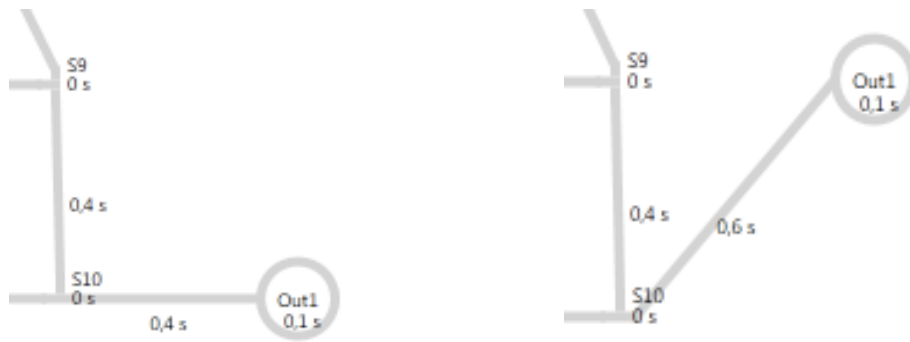


Figure 26 - Biochip Simulator screenshot. Transport calculation where component out1 is moved.

Out1 has been moved and the consequence is an increased transport time from S10 to Out1 the execution time is increased from 0.4 second to 0.6 second.

SECTION 4.1.4 BIOCHIP FLOW PATH TABLE

All flow possibilities on a finalized biochip architecture design can be calculated and put in a flow path table. This flow path table is the actual architecture model that schedulers utilize to schedule operations and flows executed on the biochip. An example of the architecture model from the simulator is shown in Figure 27.

Name	Flow Path	Sink Path	Closed Valves	Open Valves	Time	Constrains
F32	Heater1 S9 S10 Out1	Heater1 S9 S10 Out1	z57 z61	z56 z58 z60 z63	2	F8-1 F8-2 F10-1 F10-2 F13-1 F13-2 F
F31	Heater1 S9 S10 S8 S7 Filter1	Heater1 S9 S10 S8 S7 Filter1	z57 z63 z53 z48 z49	z56 z58 z60 z61 z52 z55 z50 z51	3.4	F8-1 F8-2 F9-1-1 F9-1-2 F9-1-3 F9-1
F30	Heater1 S9 S10 S8 S7 S6 S5 Out2	Heater1 S9 S10 S8 S7 S6 S5 Out2	z57 z63 z53 z49 z51 z45 z41 z43	z56 z58 z60 z61 z52 z55 z48 z50 z44 z46 z40 z42	4.2	F7-1 F7-2 F8-1 F8-2 F9-1-1 F9-1-2 F9-1
F29-8	Heater1 S9 S10 S8 Storage8	Heater1 S9 S10 S8 Storage8	z57 z63 z52 z64 z66 z68	z56 z58 z60 z61 z53 z55 z65 z67 z69	2.9	F8-1 F8-2 F9-1-1 F9-1-2 F9-1-3 F9-1
F29-7	Heater1 S9 S10 S8 Storage8	Heater1 S9 S10 S8 Storage8	z57 z63 z52 z64 z66 z69	z56 z58 z60 z61 z53 z55 z65 z67 z68	2.9	F8-1 F8-2 F9-1-1 F9-1-2 F9-1-3 F9-1

Figure 27 - Biochip Simulator screenshot. Example flow path sets

The architecture model can be serialized to an XML file, which can be used by schedulers. All file formats are available in Appendix B.

SECTION 4.2 APPLICATION DESIGN

The simulation method enables the creation of biochemical applications. A designer is able to create biochemical applications by means of an integrated tool in the simulator. The same technology used for biochip architecture design is utilized for the creation of biochemical applications. The applications are created on a drawing board as shown in Figure 28.

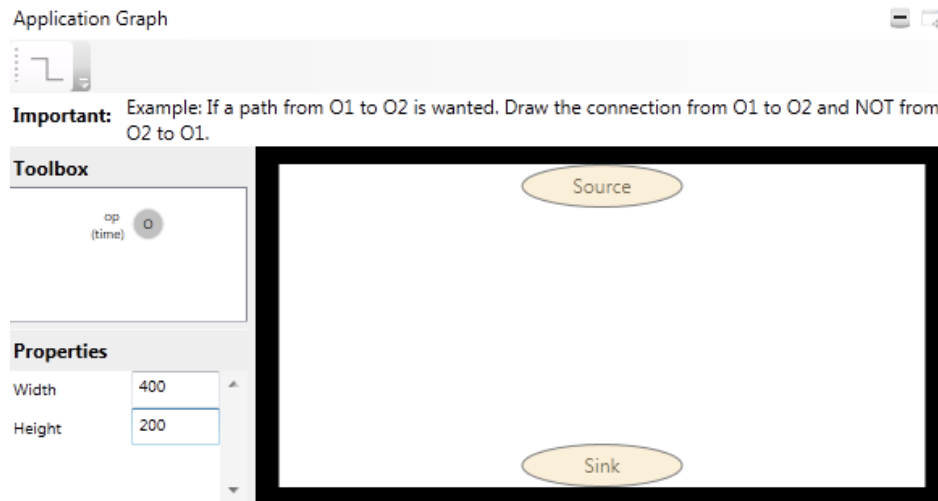
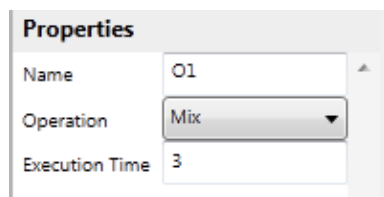


Figure 28 - Biochip Simulator screenshot. Application drawing board.

By dragging operation components from the toolbox to the drawing board, application operations are added to the biochemical application. All operation properties are set in a property view as shown in Figure 29.



(a) Property view



(b) Operation Component

Figure 29 - Biochip Simulator screenshots. Operation properties.

The property view (Figure 29.a) allows the designer to customize the operation component (Figure 29.b) by selecting a unique name, an operation type and an execution time. This operation, for example, is called O1, the operation type is a mix operation and it is performed for 3 seconds. The predecessors for the operations are set by drawing connections between the operation components. An example of a biochemical application is shown in Figure 30.

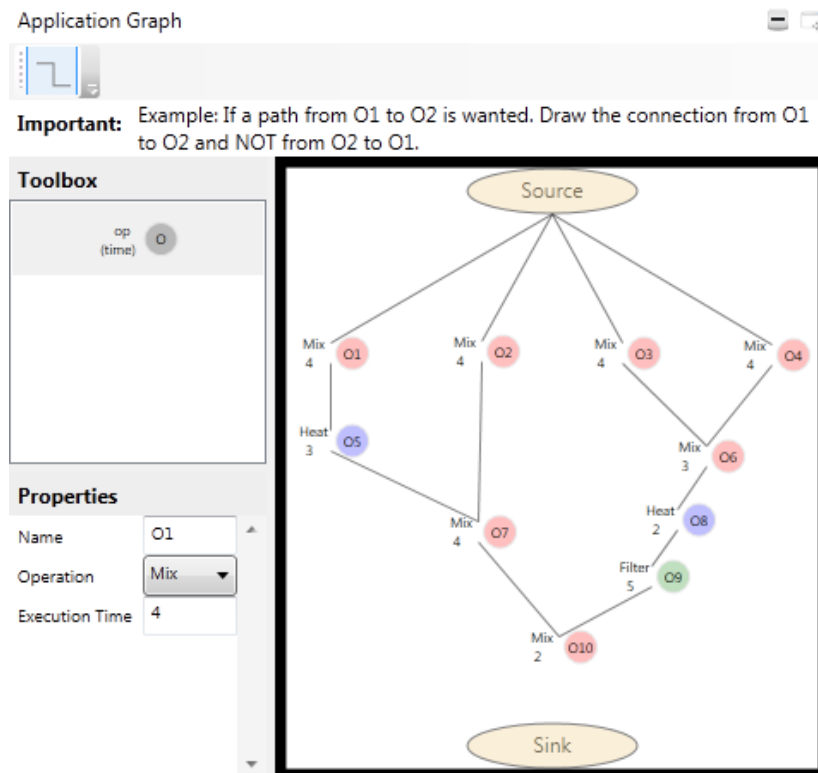


Figure 30 - Biochip Simulator screenshot. Finalized biochemical application.

The biochemical application model can be serialized to an XML file, which can be used by schedulers. All file formats are available in Appendix B.

SECTION 4.3 VISUALIZE SIMULATION

To simulate the results from schedulers a simulation model is needed. This simulation model is generated by the scheduler, which utilizes the information from the biochip architecture model and biochemical application model, to produce schedules.

A simulation model produced by a scheduler contains the following information.

A list of flow paths which consist of:

- A reference to a flow path set in a architecture model
- A start time that states when the movement of a fluid should be performed.

The above-mentioned elements in the simulation model only deal with the movement of fluidic samples. There are also elements that deals with the operations e.g. mix, heat, filter or detect. An operation element consist of:

- A reference to the operation identification in the application model
- A reference to the component that should perform the operation.
- An execution time that states for how long the operation should be performed.
- A start time that states when the operation should start.

The simulator de-serializes simulation models and converts the flows and operations in the model into simulation steps that visualize the processes on the virtual biochip architecture. The simulation model can be de-serialized from an XML file. All file formats are available in Appendix B.

By a play and step mechanism a visualization of a loaded simulation model is possible. An example is shown in Figure 31.

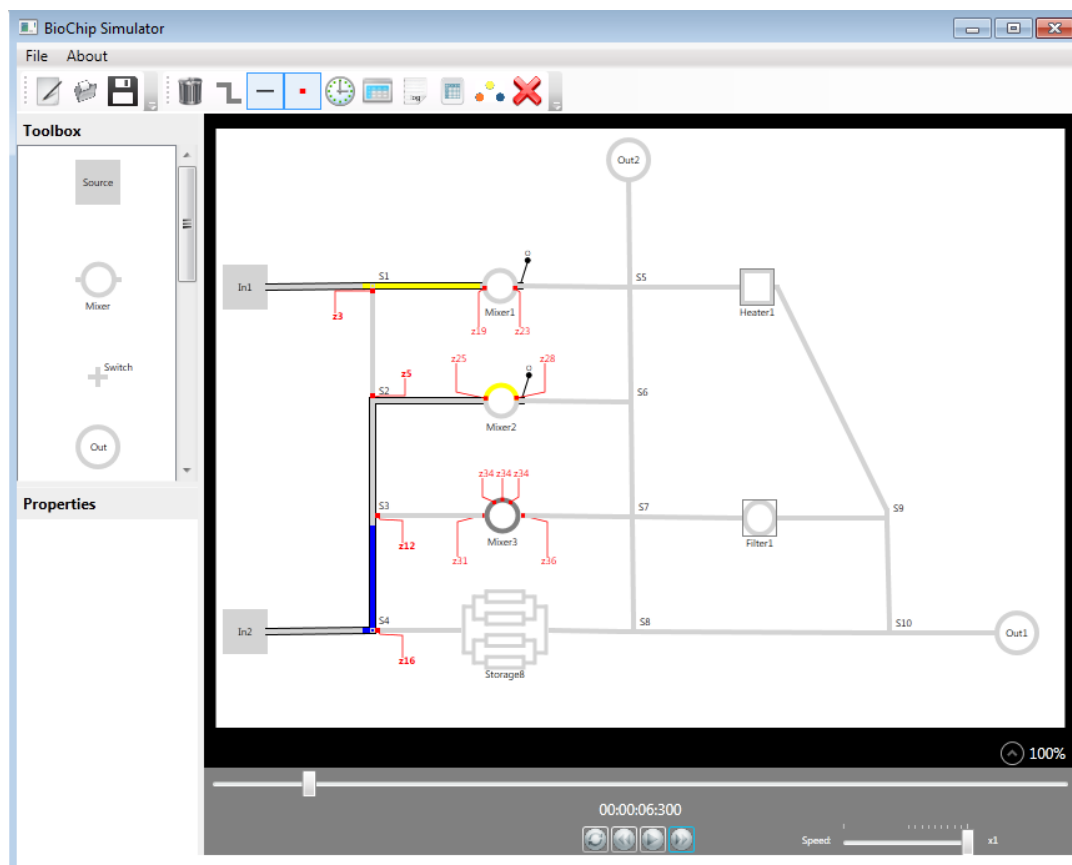


Figure 31 - Biochip Simulator screenshot. A simulation where multiple operations are executed on a biochip architecture. The control layer (red) and sink layer (black) is shown as extra information.

The figure shows how two flow paths are executed (blue fluid transported to mixer2 and yellow fluid transported to mixer1) and an operation in mixer3. The black lines mark the sink paths and the red control layer marks the valve activation. As shown in the figure the sink path ends in the mixers. This is possible because all components are

equipped with an input source and an output sink to absorb the pressure. This feature simplifies the scheduling, since a sink path and a flow path can be the same.

At the same time the biochemical application view is updated, showing that operation *O4* is executing, as shown in Figure 32.

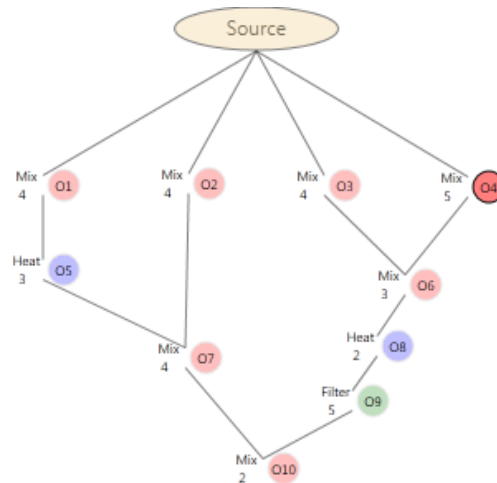


Figure 32 - Biochip Simulator screenshot. A biochemical application, where an operation is highlighted because the execution has been started.

The visualization method enables the designer to select which features to show during the simulation. It is possible to hide the sink path, control layer and biochemical application model during a simulation.

SECTION 4.4 CONTROL DATA GENERATION

Besides the visualization of a simulation model this simulation method also verifies the correctness of schedules by data generation. The control data model contains information that could support the process of optimizing biochips and schedulers further. Using the control data model makes it possible to go back and forth in a simulation model to get a snapshot of the operations and flows executed. This can support the verification of the micro valve states and fluidic samples that are moved between components. The control data model contains the following information.

- Execution time
- Flow path components
- Sink path components
- Executed operations
- Open and closed micro valves
- Fluidic information
- Automated Routing verification

Biochips are limited in many ways; one is the spacing between control channels. Micro valve control data makes it possible to optimize the use of pressure sources and control channels. One example could be if two micro valves are closed and open at the same time through a simulation, they could utilize the same pressure source and thereby minimize the use of control channel length. The micro valve data could also be fed into a biochip controller to auto-execute the application. An example of the data in the micro valve control table is shown in Figure 33.

Time	z2	z3	z4	z5	z7	z6	z9	z11	z12	z13	z14	z16	z17	z18	z19	z20	z21	z22	z23	z24	z25	z26	z
00:00:03:000	0	0	1	0	1	0	1	0	0	0	0	1	x	x	x	x	x	x	x	0	0	1	0
00:00:03:100	0	0	1	0	1	0	1	0	0	0	0	1	x	x	x	x	x	x	0	0	1	0	
00:00:03:200	0	0	1	0	1	0	1	0	0	0	0	1	x	x	x	x	x	x	0	0	1	0	
00:00:03:300	0	0	1	0	1	0	1	0	0	0	0	1	x	x	x	x	x	x	0	0	1	0	
00:00:03:400	0	0	1	0	1	0	1	0	0	0	0	1	x	x	x	x	x	x	0	0	1	0	
00:00:03:500	0	0	1	0	1	0	1	0	0	0	0	1	x	x	x	x	x	x	0	0	1	0	

Figure 33 - Biochip Simulator screenshot. Micro valve control table

The control data model verifies a schedule using automated verification. In this simulation method an automated verification of path collisions has been implemented. The automatic verification notifies the designers about flow collisions. One could argue that there are other types of errors. If fully automated verification of schedules should be performed, a verification model could be developed where all verification parameters are described and implemented like the collision verification. An example of the collision detection is shown in Figure 34 and Figure 35.

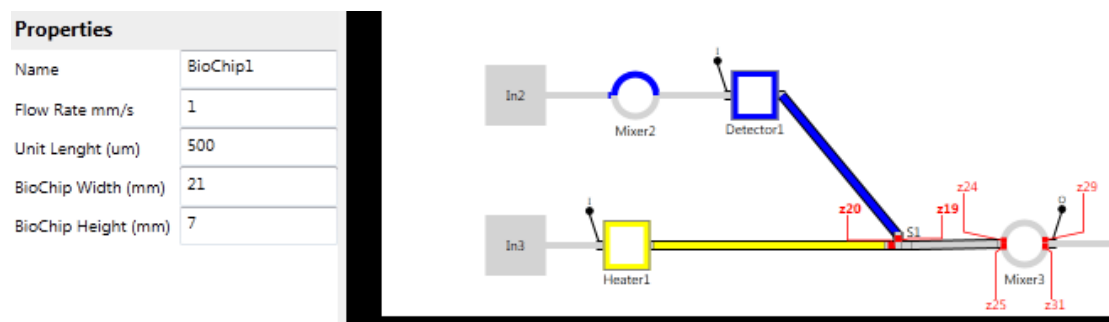


Figure 34 – Biochip Simulator screenshot. Path collision.

Time	Flow 1	Components	Flow 2	Components
2	F1-1	In1 Mixer1	F1-2	In1 Mixer1
2	F1-2	In1 Mixer1	F1-1	In1 Mixer1
19.5	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3
19.5	F6-2	Heater1 S1 Mixer3	F9-1	Detector1 S1 Mixer3
19.6	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3
19.6	F6-2	Heater1 S1 Mixer3	F9-1	Detector1 S1 Mixer3
19.7	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3
19.7	F6-2	Heater1 S1 Mixer3	F9-1	Detector1 S1 Mixer3
19.8	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3

Figure 35 - Biochip Simulator screenshot. Flow collision error view.

The figure clearly shows that the two fluidic movements are colliding. The blue and yellow fluids are about to collide in the switch (s1). The switch has activated micro valves that contradict the basic functions in the biochip architecture system model.

The control data model also allows a designer to inspect each simulation step by means of a log. The log is shown in Figure 36.

Time	Name	Flow Path	Sink Path	Closed Valves	Open Valves	Fluid
00:00:06:300	F5-2	In2 S4 S3 S2 Mixer2	In2 S4 S3 S2 Mixer2	z16 z12 z5 z28 z25	z13 z14 z9 z11 z7 z6 z24 z26 z27 z27 z27 z29 z30	F2
00:00:06:300	F3-1	In1 S1 Mixer1	In1 S1 Mixer1	z3 z19 z23	z2 z4 z17 z18 z20 z20 z20 z21 z22	F1
00:00:06:300	O4	Mixer3		z31 z34 z34 z34 z36	z32 z33 z35 z37	Mix(F1 + F2)
00:00:06:200	F5-2	In2 S4 S3 S2 Mixer2	In2 S4 S3 S2 Mixer2	z16 z12 z5 z28 z25	z13 z14 z9 z11 z7 z6 z24 z26 z27 z27 z27 z29 z30	F2
00:00:06:200	F3-1	In1 S1 Mixer1	In1 S1 Mixer1	z3 z19 z23	z2 z4 z17 z18 z20 z20 z20 z21 z22	F1

Figure 36 - Biochip Simulator screenshot. Simulation log.

Here all data from the simulation steps shown. It is possible to see the fluidic sample information. It is also possible to watch the flow path and sink path. Micro valve control data is also shown. Operation information is also shown. For example, *O4*, that mixes the fluids *F1* and *F2*.

SECTION 4.5 SUMMARY

A method and tool to simulate the logic of flow-based microfluidic biochips has been presented at user level. A method for designing biochip architecture has been developed and various parameters in the architectural design have been presented. A method to connect the components in a network of channels has been described and a method to create biochemical application has been provided. The biochemical applications are created using operation components with dynamic execution time and selective predecessors. A simulation model has been described. The simulation model supports the idea behind schedulers and the utilized architecture and application model. The

simulation model also supports graphical representation in the simulator. The last model, the control data model, enables a designer to review simulation data, which could be used for further optimization and verification of a given simulation model.

CHAPTER 5 IMPLEMENTATION

This chapter describes how the simulation method and Biochip Simulator was implemented. The simulator has many functions; biochip architecture design, extraction of flow possibilities, biochemical application design, visualization of operations and flows performed on a virtual biochip and the ability to present relevant data from a simulation. To explain how the simulator was implemented a high-level UML design diagram, shown in Figure 37, is used.

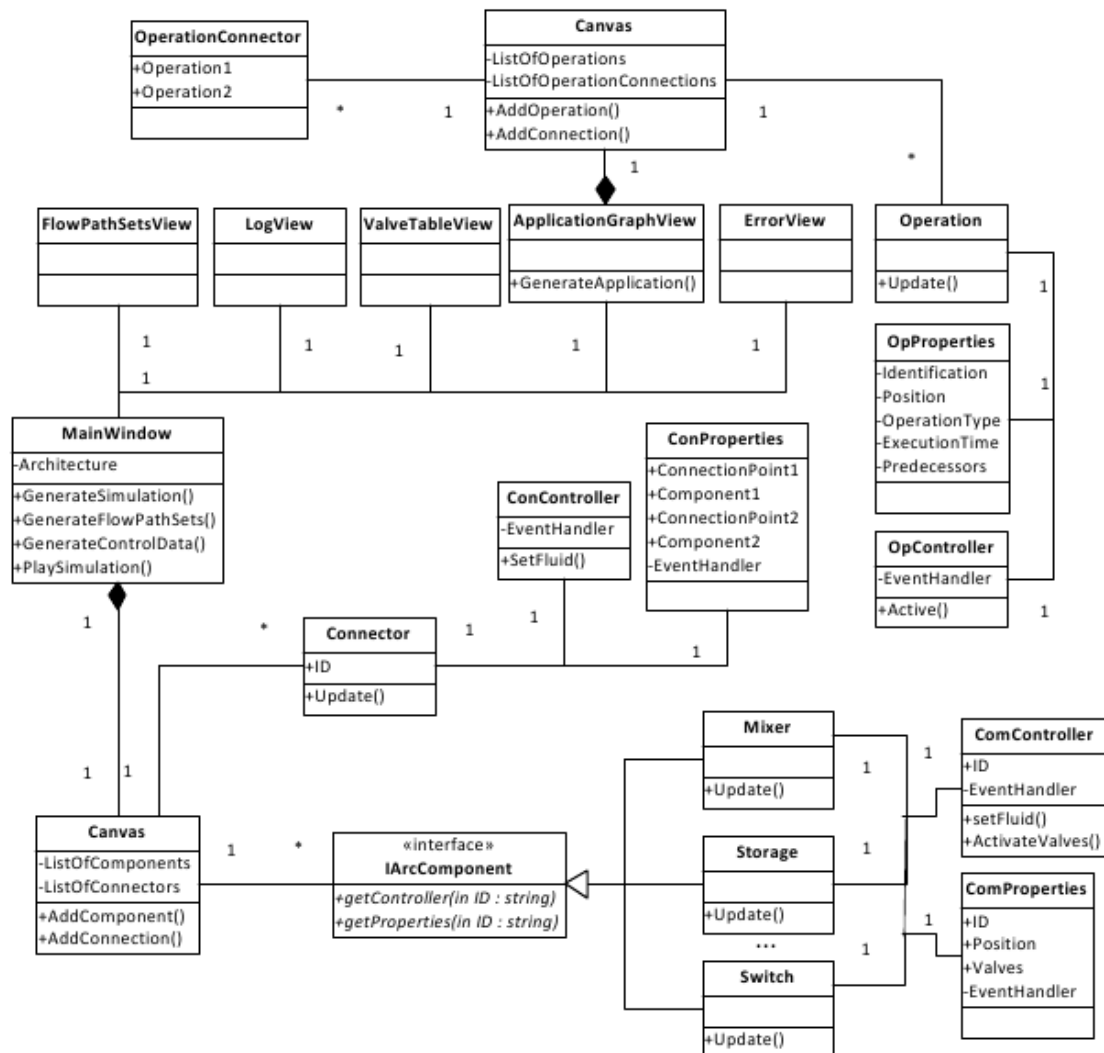


Figure 37 – High-level UML design diagram

The high-level design diagram does not show all methods and classes within the implementation of the simulator. The intention with the

diagram is to show the overall implementation and thereby reflect the simulation method presented in Chapter 4. The class, *MainWindow* can be considered as the main class in the implementation. *MainWindow* implements a graphical user interface (GUI) in the simulator. The GUI enables the creation of biochip architectures, by containing a *Canvas*, which can be considered as the drawing board where biochips are designed by adding connection channels (*Connector*) and components implementing the interface *IArcComponent*. *IArcComponent* is an interface that allows the control and customization of components added to the *Canvas*. The components are controlled using the *ComProperties* and *ComController* classes, which have *EventHandlers* that invokes *Update* methods each time a property is changed. The *ConProperties* and *ConController* classes control the *Connector* class. The advantage of having separated the controller and properties is the ability to XML serialize, which allows the simulator to convert objects into XML files. Having the controller and properties classes also separates the implementation in a presentation and data layer, which should make the implementation more logic and easy to understand for other developers.

When a biochip architecture design has been created, the flow path possibilities are created using the *GenerateFlowPathSets* method available in the *MainWindow* class. The flow path possibilities are then displayable in the view *FlowPathSetsView* and ready to be exported into XML files that are utilized by schedulers.

Biochemical applications are created using the *ApplicationGraphView* where the method *GenerateApplication* is available. The *ApplicationGraphView* contains a *Canvas*, which can be considered a drawing board for biochemical application models. The drawing board has a list of operations, which is used by the method that generates the application model. Here XML serializing is used again to convert the application models into XML files for schedulers.

When a biochip architecture and biochemical application model has been produced using the simulator, and the scheduler has generated a simulation model as described in Section 4.3, the simulation model is fed into the simulator, which uses XML serializing to de-serialize the model. The simulator then uses the *GenerateSimulation* and *GenerateControlData* methods available in the *MainWindow* class to generate a visualization of the simulation model and generate the simulation control data.

The control data is loaded into several views, a *LogView* showing each simulation step, a *ValveTableView* showing the valve control data, and an *ErrorView* showing collision error data.

All XML file formats are described in detail in Appendix B. The implementation of the simulator is performed using Windows programming; here WPF, XAML and C# have been used (Petzold, 2006).

SECTION 5.1 BIOCHIP ARCHITECTURE DESIGN

This section will describe the central implementation elements for the biochip architecture design feature.

The simulator enables the creation of biochip architecture design by drag and drop from a component library to a drawing board. By drawing connection channels between components a network of flow possibilities can be created. The drag and drop feature makes it easier for the designer to create biochip architectures and change the architecture model as shown in Section 4.1. All the drag and drop features, enabling the biochip architecture design method has been implemented in the *MainWindow* class.

The implemented biochip architecture design method allows designers to set the flow rate, unit length, width and length of a biochip, where all distances in the biochip architecture are calculated with the assumption that one pixel is equal to 25 μm . These parameters are set as properties in an architecture class, called *Architecture* shown in Figure 38.

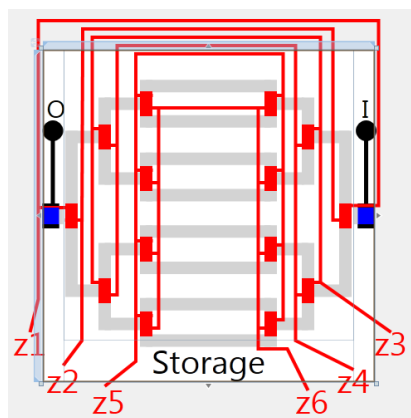
Architecture
-ID
-FlowRate
-UnitLenght
-Width
-Lenght
-ListOfComponents
-ListOfConnectors

Figure 38 - Architecture Class

The *ID* represents the name of the biochip design. *FlowRate*, represents the speed that fluidic samples move with. *UnitLenght*, represents the fluidic sample length that the biochip operates with. *Width* and *Lenght* represents the dimensions of the biochip. The *ListOfComponents* and *ListOfConnectors* represent two lists containing the property classes (*ConProperties* and *ComProperties*) from components and connection channels available on the biochip.

SECTION 5.1.1 COMPONENT DEVELOPMENT

The structure of a component is divided into five layers. The graphical presentation layer from the storage component is shown in Figure 39.



- Information Layer (labels)
- Connection layer (Blue)
- Control Layer (Red)
- Flow Layer (Gray)
- Sink Layer (Black)

Figure 39 - Layers in the storage component in the simulator

The information layer (label) is placed on top of the other layers and contains information labels, showing identification or transport time. The connection layer (blue) contains connection points placed on the component, which allows connection channels to collect information about the component and the connection point types (Input, output or two-way points), as explained in Section 4.1.2. The control layer (red) contains the micro valves controlling the component. The flow layer (gray) shows the fluidic flow channels inside the component. The sink layer (black) shows the transport direction and pressurized channels. To make a simulation more customizable all layers, except the flow layer, are hid able. The visibility of the layers is controlled using the component controller, *ComController*. All components in the simulator are implemented with an input source and output option (black layer), which have been implemented to simplify the process of calculating flow possibilities on the biochip.

All components implement an interface, *IArcComponent* that allows the simulation method to control and change the properties available in the *ComController* and *ComProperties* classes. This interface allows all components to be controlled in the same way, which makes the visualization of a simulation and the creation of biochip architecture designs easier. The fact that all components are controlled in the same way also makes it easier to add new components to the component library.

If a designer wishes to add a new component, a copy of any other component in the simulator could be used as a template. The new component will however have to be customized visually, using XAML

(Petzold, 2006). If the component has phases controlled by micro valves, these will have to be changed using C# (Petzold, 2006). A new component basically works as the following class, shown in Figure 40.

NewComponent
-EvenhandlerProperties
-EventHandlerController
-Controller
-Properties
+UpdateFlowLayer()
+UpdateSinkLayer()
+UpdateControlLayer()
+UpdateProperties()
+EventPropertiesChanged()
+EventControllerChanged()

Figure 40 - New Component Class

The *NewComponent* class is not a real component, but illustrates how components in the simulator are implemented. The *NewComponent* class has a *ComController* and a *ComProperties* property (*Controller* and *Properties*); these are set when the component is initialized. If they are changed an event will occur and the update methods are executed. For example, if a fluidic sample is set in the *Controller*, the component will be notified, and the *EventControllerChanged* method will execute; here the *UpdateFlowLayer* method is called and the component will be updated visually, showing the fluidic sample in the flow channel.

When a new component has been added to the simulator, the mechanism for adding components to the drawing board in the *MainWindow* class will also have to be updated. If the new component performs an operation (like a mixer, heater, filter or detector), the list of available operations will have to be updated, with the name of the operation as well.

SECTION 5.1.2 COMPONENT PROPERTIES

The *ComProperties* class contains the key information for a biochip architecture design, for example, position information. The *ComProperties* class holds a position for a component; the positioning is based on coordinate system theory, where components have a y-axis and x-axis position. When a drag on the drawing board has been performed, the position property will be updated with a new position. The *ComProperties* class also contains the valve identities, which can be manipulated through the property view in the simulator, presented in Section 4.1.1. Micro valve identifications are named after their pressure source, which means it is possible to activate one or more micro valves with one pressure source using the *ComController* class. All properties for a component are located in the *ComProperties* class.

SECTION 5.1.3 CONNECTION CHANNELS

The implementation of the connection channels is similar to the component implementation with a few exceptions. The concept of having a controller and properties class has been used again. The *ConController* and *ConProperties* classes have been implemented as shown in the high-level UML diagram, shown in Figure 37. One of the differences from the component implementation is the information that connection channels need, e.g. end point information. The connection channels also need information about the components it connects. The connection channels do not have phases or micro valves. The *ConController* is therefore only needs a method to hold a fluid sample.

SECTION 5.1.4 SAVE AND LOAD ARCHITECTURE DESIGNS

The simulator implementation enables designers to save and load their work. When a biochip architecture designs has been created, the *ComProperties* and *ConProperties* classes can be saved in an XML file, which can be loaded into the simulator at any time using XML serialization. This feature permits auto-generated biochip architecture designs to be loaded, the auto-generated biochip architectures can be created in another tool, which uses optimization methods to create optimized architectures for chemical applications. This feature makes the simulator more useful for designers interested in the biochip architecture design.

SECTION 5.2 ARCHITECTURE MODEL

When a biochip architecture design has been created in or loaded into the simulator, an architecture model has to be created. As mentioned, schedulers use an architecture model to produce schedules of operations to execute on a biochip. The architecture model contains the flow possibilities on a biochip. An algorithm that enables the simulator to collect all flow possibilities on the biochip has been developed. The flow possibilities are generated utilizing the algorithms presented below.

Algorithm 1. An algorithm that utilize algorithm 2 to find all flow possibilities on a biochip including micro valve, phase information and routing constraints.

Function GenerateFlowPathSets

```
For all C ∈ Components Do  
  If C can create input then  
    Initialize FlowPathSet  
    FindPaths(C, FlowPathSet) (Algorithm 2)  
  End if  
End for
```

```

For all F1 ∈ flowpathsets Do
  For all C ∈ F1.components Do
    If C is connection component Then
      F1.add(valve combination information)
    Else
      For all Phase ∈ C.Phases Do
        F2 = CloneFlowPath(F1)
        F2.identification = new identification
        Flowpathsets.add(F2)
      End for
    End if
  End for
End for
For all F1 ∈ flowpathsets Do
  For all F2 ∈ flowpathsets Do
    For all C ∈ F2.Components Do
      If F1.Components.contains(C) Then
        F1.RoutingConstraints.Add(F2)
      End if
    End for
  End for
End for
End function

```

Algorithm 2. An algorithm that finds all possible flows from one component to all possible end components. Execution time is added while the algorithm is executed.

```

Function FindPaths(Component C1, FlowPathSet F1)
  F1.Components.Add(C1)
  F1.executiontime = add calculated execution time C1
  Connectors = findConnector(C1)
  For all Con ∈ Connectors Do
    C2 = Linked component to C1 where fluid input is allowed.
    If C2 is switch component Then
      F2 = CloneFlowPath(F1)
      F2.executiontime = add calculated execution time C2
      FindPaths(C2, F2)
    Else
      F1.Components.Add(C2)
      F1.executiontime = add calculated execution time for C2
      AddFlowPathToFlowPathSet(F1)
    End if
  End for
End function

```

For each component producing a pressure in a flow channel. The recursive method *FindPaths*, presented in Algorithm 2, is used to

produce all flow paths from the component creating the input pressure. If a flow path hits a switch, which has multiple flow directions, the flow path set is cloned. As mentioned, *FindPaths* is a recursive method, the recursive method has a stop condition, it stops when a component has an output in the flow channel, which means the pressure in the flow channels can be absorbed in a sink.

The execution time of a flow path is calculated within Algorithm 2. The execution is calculated individually for each component and then added to the total execution time of the flow path. The execution time has influence on the schedules produced by the schedulers, while the priority of fluidic movement are based on the execution time. The execution time of a fluid flow in a component is calculated utilizing the flow rate and unit length properties from the biochip architecture, see Equation 1.

$$\text{Execution Time (s)} = \frac{\text{Unit length (um)}}{1000 \cdot \text{Flow Rate } \left(\frac{\text{mm}}{\text{s}}\right)}$$

Equation 1 - Transport time calculation for components

The unit size in mm divided by the flow rate in mm/s is equal to execution time in seconds for component flows.

The execution time of a fluid move in network channels placed between components is calculated from the positions as shown in Figure 41.

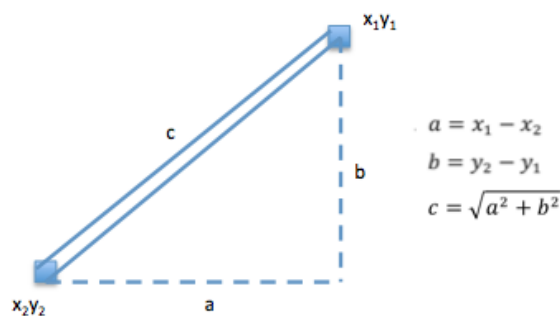


Figure 41 - Transport time calculation for connection channel

Here the length, c , between two connection points is calculated using the coordinates, (x_1, y_1) and (x_2, y_2) , on the biochip architecture. Finally, Equation 1 is utilized again, but instead of the unit size, c is used to calculate the execution time between the component connection points.

When all flow path sets have been found, all phases and valve combinations are found. The flow paths are cloned and provided with a new identification for each phase. E.g. if a flow path set has the flow *input-switch-mixer*, the mixer has 2 phases, which means there are 2

flow possibilities, as the mixer can have fluidic samples in the top or bottom channel. The flow path set would then have the identifications F1-1 where the fluid would be stored in the top channel, and F1-2 where the fluid would be stored in the bottom channel. The storage component is similar, except there are 8 phases for inputs, which means there will be 8 different flow path sets, one for each store cell in the storage component. The same applies for a flow leaving a mixer or storage - here there would be 2 flow possibilities for the mixer and 8 for the storage. This observation is important for implementations of schedulers, since the schedulers would have to determine the channel cells to move fluids to and from.

The final information source available in the flow path set is the routing constraints. These are generated by the use of the components added to the flow path sets. If two flow path sets use the same component, they cannot be executed at the same time, since their pressurized channels will collide. The identities from colliding flow path sets are added as routing constraints for each flow path set.

All flow path sets are listed in the *FlowPathSetsView* as shows in Figure 42.

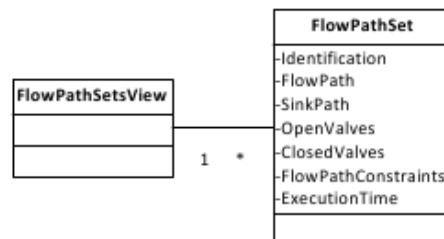


Figure 42 - FlowPathSet class

The flow path sets are exportable to an XML file that schedulers can use.

SECTION 5.3 BIOCHEMICAL APPLICATION DESIGN

Like the creation of biochip architecture designs the simulator also enables creation of biochemical applications, by drag and drop from an operation library to a drawing board. By drawing connections between the operations the predecessors for an operation is set. This feature have been implemented within the *ApplicationGraphView* class.

Like the biochip architecture components; the operation components are developed to use one property and controller class. An operation component has four main parameters, a unique identifier, an execution

time, a list of predecessors and an operation type. These parameters are set in the *OpProperties* class for the individual *Operation* class, using a property view as described in Section 4.2.

Using the *OpController* it is possible to show the operations that have been executed, by the use of an *Active* method. This feature makes it possible for the designers to follow a biochemical application step by step through a simulation.

The operations available in the simulator depend on the components in the component library. A list of the available operations is shown below:

- Filter
- Detect
- Heat
- Mix

New operations are added by the addition of new components. The process of adding components is described in Section 5.1.1.

SECTION 5.4 VISUALIZE SIMULATION

The visualization of a simulation model is the core function in the simulator. The visualization method enables the conversion of simulation models from schedulers into simulation states of fluid flows and operations on a virtual biochip architecture in the simulator.

The implemented visualization method is based on the pseudo code presented in Algorithm 3 below.

Algorithm 3. An algorithm generating simulation states, which can be used for visualization of a simulation model loaded into the Biochip Simulator.

Function GenerateSimulation

```

lsSteps = Initialize List Of Simulation Steps
tmpFlow = Initialize temp Flow Path Set
lsRunningSteps = Initialize List Of Running Simulation Steps
For all T ∈ TimeSteps Do
    For all S ∈ SimulationModel.SimulationSteps Do
        If S.StartTime = T Then
            tmpFlow = Findflow(S.flowidentification)
            SetLayers(tmpFlow)
            fluid = GetFluid(First component in tmpFlow)
            For all C ∈ tmpFlow.path.Components Do
                SimStep = initialize Simulation step
                SimStep.Fluid = fluid
                SimStep.FlowPathSet = tmpFlow information
                SimStep.ComponentInUse = C
                SimStep.StartTime = Calculated Start Time
                SimStep.EndTime = Calculated End Time
                lsStep.Add(SimStep)
            End for
            SetFluid>Last component in tmpFlow, fluid)
        End if
    End for
    FindRunningSteps(lsRunningSteps, lsStep, T)
    FindCollisions(lsRunningSteps, T)
    UpdateAllMicroValves(lsRunningSteps, T)
    ListOfControllers = GetAllComponentControllers()
    SimulationStates.Add(ListOfControllers);
End for
End function

```

The algorithm uses time steps to iterate through a simulation model. For each time step a check for simulation model steps to execute is performed. When a simulation model step matches a time step, a flow path set is found and the layers are set (sink and control layer). The setup of the control layer changes the phases of all components in the sink path. This means that a pressure from the input to the output component can be created. The fluid used in the flow is set using the first component in the flow path. Now that the fluid and control layer has been set, a simulation step is created for each component in the flow path. In each simulation step, the component in use, execution start time, execution end-time, and fluid information is set. The simulation step is then added to the list of simulation steps. The last component in

the flow is updated to contain the processed fluid. This allows a new flow from the last component later on.

Now, when all simulation steps have been created, a list of running simulation steps is updated. If any, the collision data from the running simulation steps is collected and stored. All micro valve information is updated and stored, and all controllers for the components are collected and stored in a list of controllers. This means that a state for each component is stored for each time step in the simulation model.

SECTION 5.5 DATA CONTROL GENERATION

The simulator includes control data generation, which can be used for further optimization and verification. Using the control data for micro valve pressure optimization is one part where the data could be useful. All data generation for the simulator data views are created during the simulation visualization.

SECTION 5.5.1 LOG

All activity in a simulation is stored in a log. The activities from a simulation are loaded into a View, *LogView* class. The data loaded into the simulator are selected parts from the simulation steps generated during the creation of the visualization of the simulation. This view is useful, for example, when the fluid information of an operation has to be verified. The use of the log view has been described in Section 4.4. The class used for logging is shown in Figure 43

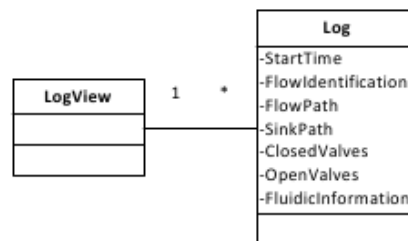


Figure 43 - Log class

The class holds the following information. *StartTime* holds the simulation step execution time. *FlowIdentification* holds the identification name of the flow path set, for example F31-1. *FlowPath* holds the component identities that the fluidic sample passes. *SinkPath* holds the components identities, from pressure source to sink. *ClosedValves* holds the active pressures source identities. *OpenValves* holds the pressures source identities that have to be none active. Finally, *FluidicInformation* holds information of the fluidic sample used at the specified start time.

SECTION 5.5.2 MICRO VALVE CONTROL DATA

The valve control data is shown in a table, *ValveTableView*; here are the data for closed, open, and unused valve pressure sources stored. The *ValveTableView* holds the micro valve control data for each time step in a simulation. The data is generated during the visualization of the simulation, using the flow path set information.

The valve control data view is generated using all pressure source identifications. These are set in the *ValveTableView* class. For each time step in a simulation a *Valves* class is initialized. The *Valves* class is shown in Figure 44.

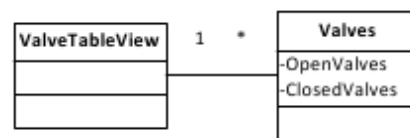


Figure 44 - Valves Class

All open valves for a time step are set in a list of *OpenValves* and all closed valves for the same time step are set in a list of *ClosedValves*. In the *ValveTableView* class the open and closed valves are marked and the rest of the valves are marked as free sources. In this way a designer can observe the patterns and other activities in the control layer in biochips. The micro valve data could also be fed into a biochip controller to auto-execute the application or used for further optimization, where more valves could be connected to the same pressure source, and reduce the total length of control channels in the biochip.

SECTION 5.5.3 AUTOMATED VERIFICATION

The simulator has an automated verification mechanism that verifies the correctness of a schedule by checking the flow path execution for collisions. The verification is performed when the visualization of the simulation is performed. For each time step in the simulation, a verification of the running flow paths and the routing constraints is performed. By a comparison of the flow path set identifier and the sets of routing constraints this verification can be performed.

If, e.g. the flow path sets *F1-1* and *F31-1* are executed at the same time, the simulation visualization method checks the routing constraints for *F1-1* for the *F31-1* identity. If *F31-1* is found in the constraints, a collision error object will be initialized and added to the list of collision errors in the *ErrorView* class. The collision error class is shown in Figure 45.

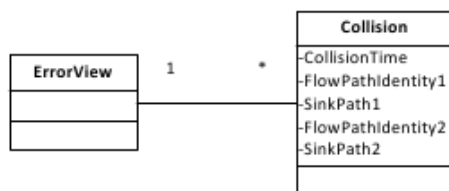


Figure 45 - Collision Class

A collision time is stored in the property *CollisionTime*. A *Collision* also holds the information of the two flow path sets that are colliding, in *FlowPathIdentity1* and *FlowPathIdentity2*. The sink paths are also stored in *SinkPath1* and *SinkPath2*. The automated verification is helpful, when the biochip designs and biochemical applications become more complex, since the verification is difficult just by the visualization. The automated verification mechanism helps the designer to find the places in a simulation, where collisions occur.

SECTION 5.6 SUMMARY

The implementation of the Biochip Simulator includes more than visualization of schedules produced by schedulers. Methods to create biochip architecture designs and biochemical applications have been implemented. The creation of biochip architectures and chemical application includes the implementation of eight components, a connection mechanism and a mechanism that allows designers to draw chemical application graphs. An algorithm analyzing the flow possibilities on a biochip has been described and implemented. The flow possibilities are presented in a table view and the possibility to save all information in files has been implemented. A method that enables visualization of schedules has been implemented. The method enables a graphical representation of schedules containing operations and flows to be executed on virtual biochip architectures. Relevant data generation that shows important data from a simulation has been implemented. The data generation mechanism includes automatic verification of flow collisions.

CHAPTER 6 EXPERIMENTAL EVALUATION

The work have been evaluated by using the simulator and thereby the developed simulation method. This includes the creation of biochip architectures and biochemical applications, the use of a scheduler to schedule the operations, verification of the schedule by a visual presentation and data generation in the simulator.

The evaluation uses different test cases where the simulator has been used to verify the schedule and biochip architecture. Table 4 shows information about the test cases. The cases presented are all cases used by the DTU scientist team working with schedulers. Test case one in Table 4 presents the mixing stage of a Polymerase Chain Reaction (PCR) application (Su & Chakrabarty, 2006).

Test Case	Type	Components	Operations	Possible Flow Path
1	Real Life	23	7	42
2	Synthetic	16	10	23
3	Synthetic	36	20	66
4	Synthetic	52	30	92

Table 4 - Test Cases Information

All cases have been executed and a video recording has been made. The recordings are available on the thesis home page, <https://sites.google.com/site/biochipsimulator/evaluation-test-cases>, located under the evaluation pages. All test cases were tested using the same workflow, which is presented below using test case 2.

In Figure 46 a handmade drawing is shown. The drawing was made during a test of the scheduler produced at DTU. The drawing includes the application graph and the biochip architecture for test case 2.

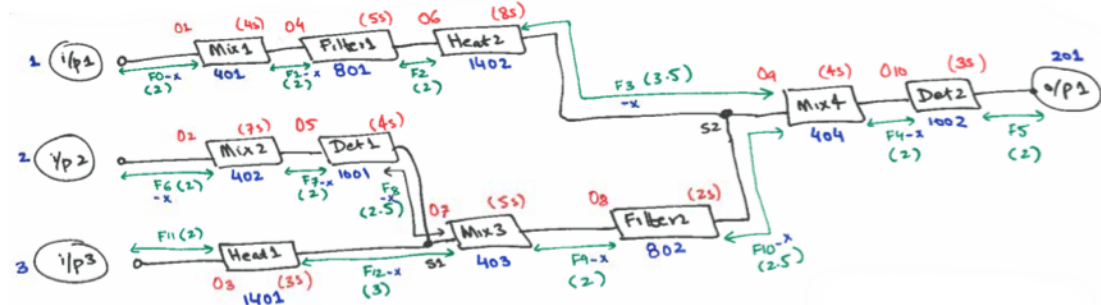


Figure 46 - Handmade biochip architecture design

The architecture follows the structure of the biochemical application. Which means that each operation in the biochemical application has a dedicated component on the biochip to perform the operation.

Figure 47 shows the biochip architecture converted into a biochip architecture design in the simulator and Figure 48 shows the biochemical application created using the simulator.

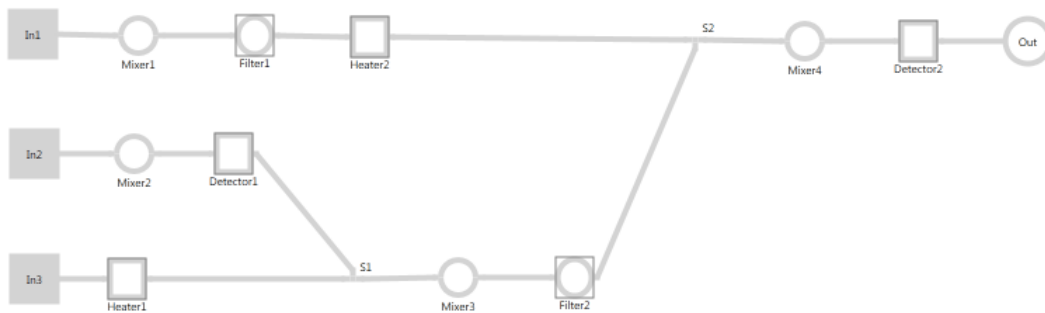


Figure 47 - Biochip Simulator screenshot. Biochip architecture design

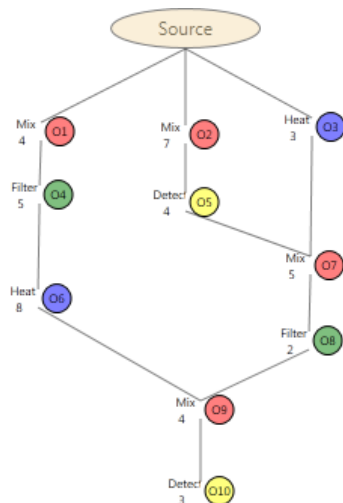


Figure 48 - Biochip Simulator screenshot. Biochemical application design

Figure 49 presents a snapshot of the flow possibilities on the biochip architecture, which is utilized by the scheduler to schedule the operations and flows to execute. Table 5 presents the schedule produced by the DTU scheduler.

Flow Path Sets						
Name	Flow Path	Sink Path	Closed Valves	Open Valves	Time	Constrains
F13	Detector2 Out	Detector2 Out			0.3	F12-1 F12-2
F12-2	Mixer4 Detector2	Mixer4 Detector2	z41 z36	z35 z37 z38 z39 z40 z42 z43	0.3	F8-1 F8-2 F11-1 F11-2 F12-1 F13
F12-1	Mixer4 Detector2	Mixer4 Detector2	z37 z43	z35 z36 z38 z39 z40 z41 z42	0.3	F8-1 F8-2 F11-1 F11-2 F12-2 F13
F11-2	Filter2 S2 Mixer4	Filter2 S2 Mixer4	z32 z41 z36	z33 z34 z35 z37 z38 z39 z40 z42 z43	0.7	F8-1 F8-2 F10-1 F10-2 F11-1 F12-1 F12-2
F11-1	Filter2 S2 Mixer4	Filter2 S2 Mixer4	z32 z37 z43	z33 z34 z35 z36 z38 z39 z40 z41 z42	0.7	F8-1 F8-2 F10-1 F10-2 F11-2 F12-1 F12-2
F10-2	Mixer3 Filter2	Mixer3 Filter2	z29 z24	z23 z25 z26 z27 z28 z30 z31	0.3	F6-1 F6-2 F9-1 F9-2 F10-1 F11-1 F11-2

Figure 49 - Biochip Simulator screenshot. Flow Path Table

SCHEDULE					
Start Time (s)	Schedule for? Edge (1)/ Operation (0)	Flow Path ID	Associated Operation Number	Bound to	Finish Time (s)
0	1	F1-1	O1	Mixer-1	2
0	1	F2-1	O2	Mixer-2	2
0	1	F3	O3	Heater-1	2
2	1	F1-2	O1	Mixer-1	4
2	1	F2-2	O2	Mixer-2	4
2	0	----	O3	Heater-1	5
4	0	----	O1	Mixer-1	8
4	0	----	O2	Mixer-2	11
8	1	F4-1	O4	Filter-1	10
10	0	----	O4	Filter-1	15
11	1	F5-1	O5	Detector-1	13
13	0	----	O5	Detector-1	17
15	1	F7	O6	Heater-2	17
17	1	F9-1	O7	Mixer-3	19,5
17	0	----	O6	Heater-2	25
19,5	1	F6-2	O7	Mixer-3	22,5
22,5	0	----	O7	Mixer-3	27,5
27,5	1	F10-1	O8	Filter-2	29,5
29,5	0	----	O8	Filter-2	31,5
31,5	1	F11-1	O9	Mixer-4	34
34	1	F8-2	O9	Mixer-4	37,5
37,5	0	----	O9	Mixer-4	41,5
41,5	1	F12-1	O10	Detector-2	43,5
43,5	0	----	O10	Detector-2	46,5
46,5	1	F13	O10	Detector-2	48,5

Table 5 – Scheduler output. Schedule of operations to execute in the biochip.

Using the simulator a visualization of the schedule was made. A sequence of screenshots from the simulation is shown in Figure 50. The schedule was successfully executed. It was clearly presented that mixers and other components only contained the amount of fluids that was allowed. It was clear that mixers did not start mixing fluids, without two fluid samples within in the mixer components. In Figure 50.a it is shown that the two first mixes are performed with two fluid samples in both mixers (green and blue) and a heat operation is performed in heater1 (yellow). In Figure 50.b two fluids (pink and green) are ready to be mixed in mixer 4. In Figure 50.c the out is transported from detector2 to the out component.

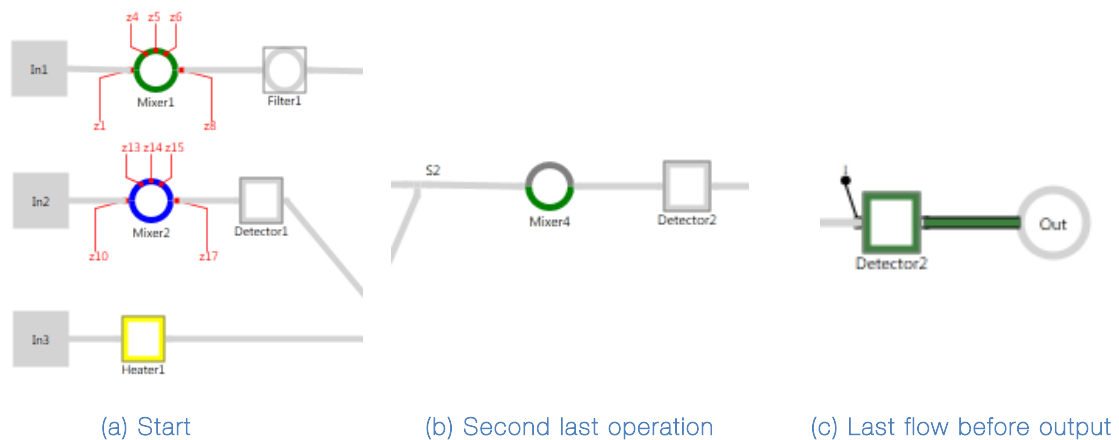


Figure 50 - Biochip Simulator screenshots. A sequence of screenshots showing the simulation.

Figure 51, Figure 52 and Figure 53 show snapshots of the control data, which enable further verification of fluids, micro valves, execution time, sink path, flow path. It was clear that the operations within the biochemical application were performed with the expected fluidic samples. The micro valve table showed the state of all micro valve placed on the biochip for all time steps in the simulation and there were no collision errors in the schedule produced by the scheduler.

Simulation Log						
Time	Name	Flow Path	Sink Path	Closed Valves	Open Valves	Fluid
00:00:46:700	F13	Detector2 Out	Detector2 Out			Detect(Mix(Filter(Mix(Detect(Mix(F2 + F2)) + Heat(F3)))) + Heat(Filter(Mix(F1 + F1))))
00:00:46:600	F13	Detector2 Out	Detector2 Out			Detect(Mix(Filter(Mix(Detect(Mix(F2 + F2)) + Heat(F3)))) + Heat(Filter(Mix(F1 + F1))))
00:00:46:500	F13	Detector2 Out	Detector2 Out			Detect(Mix(Filter(Mix(Detect(Mix(F2 + F2)) + Heat(F3)))) + Heat(Filter(Mix(F1 + F1))))
00:00:46:400	O10	Detector2				Detect(Mix(Filter(Mix(Detect(Mix(F2 + F2)) + Heat(F3)))) + Heat(Filter(Mix(F1 + F1))))
00:00:46:300	O10	Detector2				Detect(Mix(Filter(Mix(Detect(Mix(F2 + F2)) + Heat(F3)))) + Heat(Filter(Mix(F1 + F1))))

Figure 51 - Biochip Simulator screenshot. Simulation log.

Valves Control Table

Information: 1 : Closed X : Free 0 : Open

Time	z1	z2	z3	z4	z5	z6	z7	z8	z9	z10	z11	z12	z13	z14	z15	z16	z17	z18	z19	z20	z22	z23	z24	z25	z26	z27	z28	z29	z30	z31
00:00:22:800	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	1	1	0	1	0
00:00:22:900	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	1	1	0	1	0
00:00:23:000	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	1	1	0	1	0
00:00:23:100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	1	1	0	1	0

Figure 52 - Biochip Simulator screenshot. Micro Valve control data table.

Time	Flow 1	Components	Flow 2	Components
------	--------	------------	--------	------------

Figure 53 - Biochip Simulator screenshot. Collision errors.

All test cases were successfully tested using the functions available in the simulator. Building biochips and biochemical applications are functions making the biochip simulator a more holistic simulation tool. The simulator produces control data that could be fed to a real biochip controller to auto-execute biochemical application. But designers should be aware of the design rules and execution time for the flows. The simulator does not have the ability to design physical biochips and the execution time for instance, is calculated with one flow rate for all fluids. It is important to mention that the simulator was intended to visualize the logic of flow-based microfluidic biochips and not produce designs for real biochips. To use this tool for physical biochip simulation the assumptions about fluid density and design rules must be handled by another tool or considered into the calculations.

Simulation makes it possible to follow the fluids and the logic functions in the biochip. It is possible to adjust parameters as the flow rate and unit size. An example where an adjustment of a parameter results in a collision is shown in Figure 54.

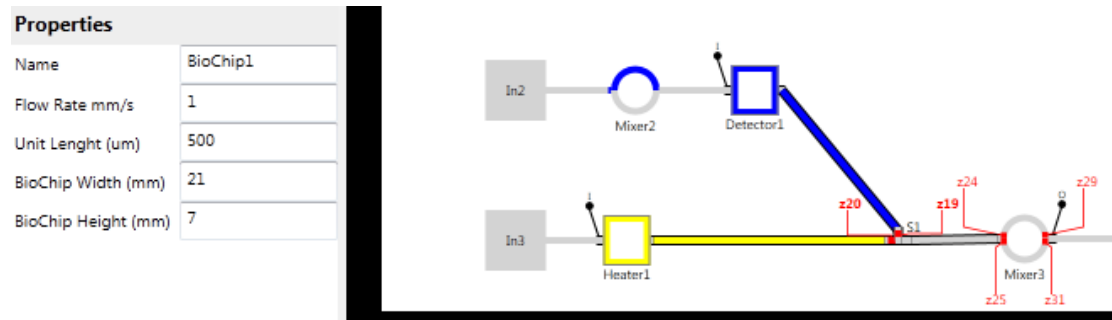


Figure 54 - Biochip Simulator screenshot. Fluid flow collision.

The flow rate is changed to 1 mm/s from 10 mm/s. The figure clearly shows that something is wrong. Which is useful information for a schedule designer. The fluid collision is also shown in the collision error view as shown in Figure 55.

Collision Errors					
Time	Flow 1	Components	Flow 2	Components	
2	F1-1	In1 Mixer1	F1-2	In1 Mixer1	
2	F1-2	In1 Mixer1	F1-1	In1 Mixer1	
19.5	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3	
19.5	F6-2	Heater1 S1 Mixer3	F9-1	Detector1 S1 Mixer3	
19.6	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3	
19.6	F6-2	Heater1 S1 Mixer3	F9-1	Detector1 S1 Mixer3	
19.7	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3	
19.7	F6-2	Heater1 S1 Mixer3	F9-1	Detector1 S1 Mixer3	
19.8	F9-1	Detector1 S1 Mixer3	F6-2	Heater1 S1 Mixer3	
19.8	F6-2	Heater1 S1 Mixer3	F9-1	Detector1 S1 Mixer3	

Figure 55 - Biochip Simulator screenshot. Fluid collision error view

The collision error view shows one kind of error that typically occurs in schedules. But there are other kinds of errors; one error could be that a mixer tries to mix two fluids without containing two fluids. This kind of error could be shown as well in the error view. The fact that designers can adjust the biochip design and the parameters enables the designers to find new ways of constructing biochips.

The simulator makes it possible to present the logic functions of flow-based microfluidic biochips. Scientists can use this information, an example could be that mixers and storage components have a set of phases or states and the schedulers should be able to pick the right state for these components.

The usability of the simulator is also something that has been important. As the related work in Section 1.1 shows the tool available at the moment are not optimal when larger biochips should be designed. The simulator has drag and drop features allowing the designers to easily

use the tool and build chips. The fact that files can be saved and loaded into the simulator also allows development of other optimization tools. One example could be an architecture optimization tool, which follows the standard file formats of the simulator.

The simulator and simulation method presents the first steps in flow-based microfluidic biochip simulation. Scheduler designers have found the simulator useful; errors and missing knowledge in the scheduler software have been discovered. Scheduler designers have been able to visualize their work and thereby verify the correctness of schedules.

CHAPTER 7 FUTURE WORK

This chapter presents the future work for the Biochip Simulator and simulation method. Here are ideas and observations made during the implementation and analysis phase presented. The ideas and observations could be investigated in greater detail before an actual implementation could take place.

Design Rules

The simulation method could be even more realistic if more design rules from physical biochips were implemented. Some design rule examples are presented in the list below:

- Minimum spacing between flow channels
- Minimum spacing between control channels
- Minimum spacing between external ports
- Width of flow channels width of valves
- Minimum spacing between components
- Pressure source speed

If the design rules were implemented the simulation could be used for simulation of 1:1 biochips and not just the logic of flow-based biochips. The data collected from a simulation could be directly used for real biochips (Designing Your Own Device, -).

The creation of biochip architectures is limited in many ways. The channels and valves cannot be too close to each other, and the maximum thickness of the channels is dependent on the type of micro valves. These design rules are considered out of scope in this thesis, since the focus is on the logic of flow-based microfluidic biochips.

Automated verification of schedules

The simulation method does not perform fully automated verification of schedules. One error check has been implemented, but there are other kinds of errors that could be observed automatic. This means that a verification of a schedule depends on the users ability to see the fault in the schedule. An automatic verification of biochip architectures could be implemented using the design rules presented and automatic verification of schedules could be implemented using the flow and operation information from the simulation models from schedulers.

Flow Execution time

The calculation of the fluid flow execution time is not exact. The corners in the flow layer channels on real biochips courses a friction when fluids are moved through the corner. That friction slows down the fluid, and thereby the execution time is increased. The density of fluids can vary and change the flow execution time. To make the simulation method more realistic these calculations could be implemented.

Automated Architecture creation

At the moment the simulation method offers the ability to create architectures using drag and drop. But the simulator could also create biochip architectures automatically. By entering the number of components that is available and then optimize the biochip architecture by the use of architectural synthesis. The creation of the biochip architecture should also include the information from chemical applications. The scientist team at DTU already implements this function in an experimental project.

Customized connection channels

The ability to make connection channels be more customized would make the tool more realistic. At the moment connection channels cannot corner. It is only possible to make a connection channel that goes in a straight line. This missing function limits the design options.

It should be mentioned that the implemented simulator has the classes and flow execution time calculation method optimized to contain and calculate connection channels with corners, which simplifies this implementation.

3D simulation

Biochips these days can have multiple flow and control layers. The developed simulation method cannot simulate multilayered biochips. This function could be implemented by showing multiple layers at the same time in the simulation view.

It could also be interesting to make a solution with Microsoft .NET 3D graphics, which allows the creation of 3D figures on the screen (Microsoft).

Chemical Library

Introducing a chemical library where the simulation method could lookup chemical or biochemical reactions could be implemented. This would make the simulation more realistic. A standard chemical library could be

found or developed and included in the simulator. The operations performed in the simulator could then display real chemical reactions.

Manual sink path creation

The simulation method does not allow a manual creation of sink path for a flow path. This means that a sink path will always be the same as the flow path. For a more realistic approach, the simulation method should allow manual sink path creation.

It must be mentioned that this feature is more or less implemented; since all components already have a sink path and the flow path sets are able to contain sink path components, adding the manual approach could be done by letting the user change the flow path set table available in a view in the simulator. The flow possibility calculation algorithm already finds routing constraints using the components from the sink path.

CHAPTER 8 CONCLUSION

This thesis presents the problem of flow-based microfluidic biochip simulation and proposes a solution to it. The problem was solved by the development of a simulation method and tool that enables simulation of the logic functions in flow-based microfluidic biochips. The simulation method includes more than a visualization of operations and flows executed on a virtual flow-based biochips. The method includes a solution to the creation of virtual biochip architectures and biochemical applications. An algorithm capable of generating all flow possibilities on a biochip has been implemented; here routing constraints, micro valve information, execution times, sink paths and flow paths are generated. A visualization of a schedule is possible using a developed simulation function that enables the simulation of the logic in flow-based microfluidic biochips. Finally, data generation methods have been presented and implemented. During the development of the simulation method, observation and ideas have been collected and presented as future work.

The findings from the development of the simulation method have been implemented in a tool called Biochip Simulator. All Biochip Simulator documentation, executable files and test cases are available on the thesis home page. On the thesis home page the following elements are available:

- "Biochip Simulator – User Guide" – A detailed guide that describes how to use the simulator.
- "Biochip Simulator – Component design" – A document describing the available simulation components and their functions.
- "Biochip Simulator – File Formats" – A document that describes the XML file formats used by the simulator.
- Biochip Simulator executable file – An executable file which allows users to use the simulator.
- Biochip Simulator evaluation – All the files and videos that have been used for the evaluation of the simulator.

To the best of our knowledge this tool is the first to present a method to simulate and visualize the logic of flow-based microfluidic biochips.

REFERENCES

- Amarasinghe, S., Thorsen, T., Urbanski, J. P., William, T., & Rhodes, C. (2005). Digital microfluidics using soft lithography.
- Amin, N. (2008). *Micado*. Retrieved from <http://groups.csail.mit.edu/cag/micado/index.html>
- Amin, N. (2008, 12). Computer-Aided Design for Multilayer Microfluidic Chips.
- Basic Design Rules*. (-). Retrieved from Stanford Microfluidics Foundry: <http://www.stanford.edu/group/foundry/Basic%20Design%20Rules.html>
- Bill, T. (n.d.). *Programmable Microfluidics*. Retrieved from <http://groups.csail.mit.edu/cag/biostream/>
- Bruegge, B., & Dutoit H., A. (2004). *Object-Oriented Software Engineering - using UML, Patterns, and Java*. (S. Edition, Ed.)
- Chakrabarty, K. (2007). *Automated Design of Microfluidics-Based Biochips: Connecting Biochemistry to Electronics CAD*. Duke University .
- Chakrabarty, K. *Design, Testing, and Applications of Digital Microfluidics-Based Biochips*. Duke University, Department of Electrical and Computer Engineering , Durham.
- Chou, H.-P., Unger, M. A., & R. Quake, S. (2001). A Microfabricated Rotary Pump.
- Cooper, M., Wentzlaff, D., Thorsen, T., Thies, W., Urbanski, J., & Amarasinghe, S. (2004). Programable microfluidics.
- Cormen h., T., Leiserson E., C., Rivest L., R., & Clifford, S. (2007). *Introduction To Algorithms*. (S. Edition, Ed.)
- Cottet, F., Delacroix, J., Kaiser, C., & Mammeri, Z. (2002). *Scheduling In Real-Time System*. Wiley.
- Designing Your Own Device*. (-). Retrieved from Stanford Microfluidic Foundry: <http://stanford.edu/group/foundry/Basic Design Rules.html>
- Getting started AUTOCAD*. (-). Retrieved from Stanford Microfluidic Foundry: http://www.stanford.edu/group/foundry/Getting_started_AUTOCAD.html
- Mark, D., Haeberle, S., Roth, G., Stetten, F. v., & Zengerle, R. (2010). *Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications*. Chem.

Melin, J., & Quake, S. R. (2007). *The Evolution of Design Rules for Biological Automation*. Stanford University and Howard Hughes Medical Institute.

Microfluidic valve technology. (n.d.). Retrieved from Stanford Microfluidics Foundry: <http://www.stanford.edu/group/foundry/Microfluidicvalvetechnology.html>

Microsoft. (n.d.). *3-D Graphics Overview*. Retrieved from MSDN: <http://msdn.microsoft.com/en-us/library/ms747437>

Minhass, W. H., Pop, P., & Madsen, J. (2011). Architectural synthesis of flow-based microfluidic large-scale integration biochips.

Minhass, W. H., Pop, P., & Madsen, J. (2011). System-level modeling and synthesis of flow-based microfluidic biochips.

Petzold, C. (2006). *Applications = Code + Markup*. Microsoft.

Roy, S. (2010). *CAD Algorithms for Digital Microfluidic Biochips*.

Stanford Microfluidic Foundry. (-). Retrieved from Stanford Microfluidic Foundry: <http://www.stanford.edu/group/foundry/index.html>

Su, F., & Chakrabarty, K. (2006). "*Benchmarks*" for digital microfluidic biochip design and synthesis. Duke University.

Testing Your Device. (-). Retrieved from Stanford Microfluidics Foundry: <http://www.stanford.edu/group/foundry/Testing%20Your%20Device.html>

Thies, W. (2007). *Programmable Microfluidic Biochips*. Retrieved from <http://www.youtube.com/watch?v=nOgmTc2wA3k>

Thies, W. (2007). *Programmable microfluidics* . Retrieved from <http://groups.csail.mit.edu/cag/biostream/>

Troelsen, A. (2005). *Pro C# 2005 and the 2.0 .NET platform*. Apress.

Whitesides, G. M. (2006). *The origins and the future of microfluidics*. Nature.

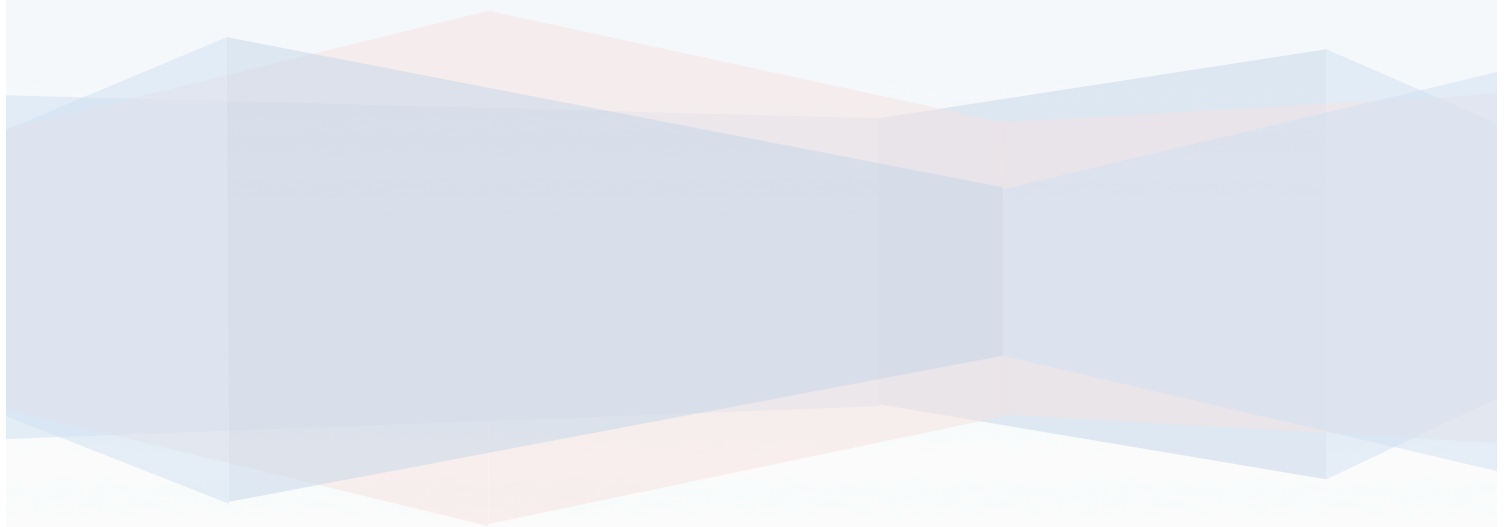
APPENDIX A BIOCHIP SIMULATOR – COMPONENTS

See documentation available on the thesis web site or on the next page.

BioChip Simulator

Components Design

Morten Foged Schmidt



This Page Intentionally Left Blank

Table of Contents

1	INTRODUCTION	4
2	COMPONENTS	5
2.1	MIXER	5
2.2	STORAGE	5
2.3	SWITCH	6
2.4	DETECTOR	6
2.5	HEATER	6
2.6	FILTER	7
2.7	INPUT	7
2.8	OUTPUT	7

1 INTRODUCTION

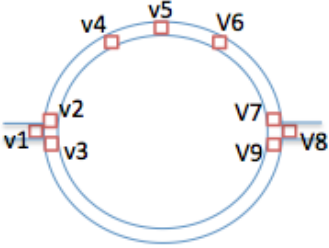
This document contains information about the components library available in the simulator. A table containing phases or states, operations and a description is available for each component.

2 COMPONENTS

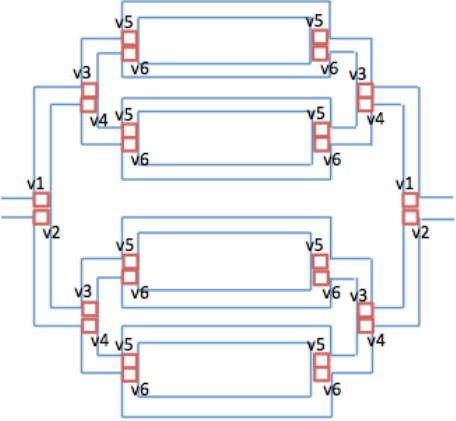
The subsections contain a description for each of the component available in the simulator.

Components with micro valves have a phase or state table. The phase table describe which micro valve to open (0) or close (1) to achieving a phase or state on the component.

2.1 MIXER

Mixer										
<p>The mixer component enables a mixing operation, where two fluids are mixed. The mixed fluid can be used in new components but the mixed fluid can only be moved in unit sizes (top or bottom). The mixer has five phases as shown the table below.</p> <p>The mixer has an operation called mix, which can be performed for a specified seconds.</p>										
	Phase	v1	v2	v3	v4	v5	v6	v7	v8	v9
	ip1	0	0	1	0	0	0	0	0	1
	ip2	0	1	0	0	0	0	1	0	0
	mix	1	0	0	Mix	Mix	Mix	0	1	0
	op1	0	0	1	0	0	0	0	0	1
	op2	0	1	0	0	0	0	1	0	0

2.2 STORAGE

Storage							
<p>The storage component enables a storing of fluids on the biochip. The stored fluids can be kept for later use on the biochip. The storage has 16 phases as shown the table below.</p>							
	Phase	z1	z2	z3	z4	z5	z6
	lp1	0	1	0	1	0	1
	lp2	0	1	0	1	1	0
	lp3	0	1	1	0	0	1
	lp4	0	1	1	0	1	0
	lp5	1	0	0	1	0	1
	lp6	1	0	0	1	1	0
	lp7	1	0	1	0	0	1
	lp8	1	0	1	0	1	0
	op1	0	1	0	1	0	1
	op2	0	1	0	1	1	0
	op3	0	1	1	0	0	1
	op4	0	1	1	0	1	0

	op5	1	0	0	1	0	1
	op6	1	0	0	1	1	0
	op7	1	0	1	0	0	1
	op8	1	0	1	0	1	0

2.3 SWITCH

Switch																																				
The switch component enables the fluid flow to be directed between components. The switch has four phases as shown the table below.																																				
<p>The diagram shows a central square with four valves labeled V1, V2, V3, and V4. V1 is at the top, V2 on the left, V3 at the bottom, and V4 on the right. Four ports labeled z1, z2, z3, and z4 are connected to the valves. z1 is top-left, z2 is bottom-left, z3 is bottom-right, and z4 is top-right.</p>	<table border="1"> <thead> <tr> <th>Phase</th> <th>z1</th> <th>z2</th> <th>z3</th> <th>z4</th> </tr> </thead> <tbody> <tr> <td>Dir1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>Dir2</td> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>Dir3</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>Dir4</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>Dir5</td> <td>0</td> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>Dir6</td> <td>1</td> <td>0</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	Phase	z1	z2	z3	z4	Dir1	1	1	0	0	Dir2	0	1	1	0	Dir3	0	0	1	1	Dir4	1	0	1	0	Dir5	0	1	0	1	Dir6	1	0	0	1
	Phase	z1	z2	z3	z4																															
	Dir1	1	1	0	0																															
	Dir2	0	1	1	0																															
	Dir3	0	0	1	1																															
	Dir4	1	0	1	0																															
	Dir5	0	1	0	1																															
Dir6	1	0	0	1																																

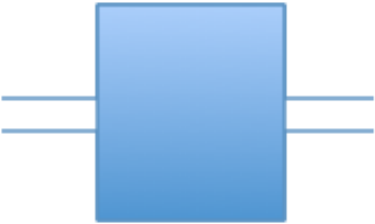
2.4 DETECTOR

Detector	
The detector component enables a detection operation on the biochip. The detector has an operation called detect, which can be performed for a specified seconds.	
<p>The diagram shows a blue rectangular component with two horizontal lines on the left and two on the right, representing input and output channels.</p>	Has no valves or phases.

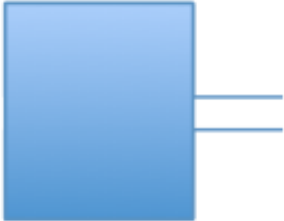
2.5 HEATER

Heater	
The heater component enables a heat operation on the biochip. The heater has an operation called heat, which can be performed for a specified seconds.	
<p>The diagram shows a blue rectangular component with two horizontal lines on the left and two on the right, representing input and output channels.</p>	Has no valves or phases.

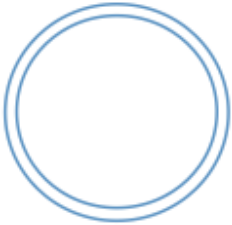
2.6 FILTER

Filter	
The filter component enables a filter operation on the biochip. The filter has an operation called filter, which can be performed for a specified seconds.	
	Has no valves or phases.

2.7 INPUT

Input	
The input component enables the input of fluids into the biochip. It is possible to change the colors on fluids and change to name of fluids.	
	Has no valves or phases.

2.8 OUTPUT

Input	
The output component works as a sink, where waste or the fluid results can be sent.	
	Has no valves or phases.

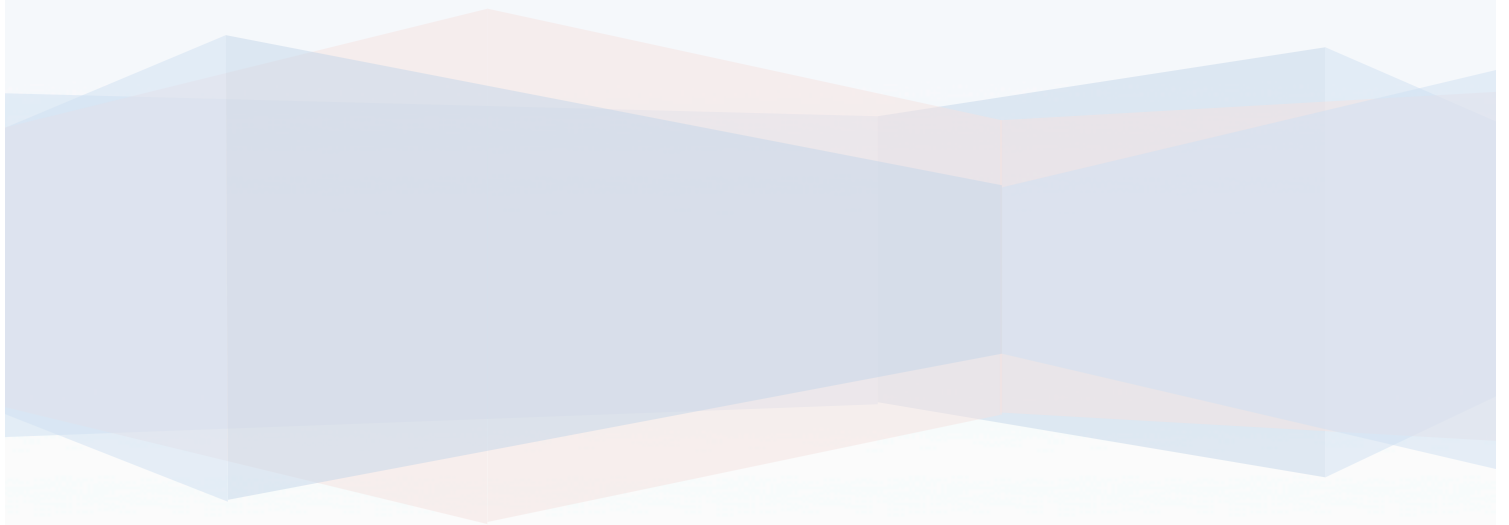
APPENDIX B BIOCHIP SIMULATOR – FILE FORMATS

See documentation available on the thesis web site or on the next page.

BioChip Simulator

File Formats

Morten Foged Schmidt



This Page Intentionally Left Blank

Table of Contents

1	INTRODUCTION	4
2	FILE FORMATS	5
2.1	ARCHITECTURE FILE	5
2.2	APPLICATION GRAPH FILE	6
2.3	FLOW PATH SET FILE	7
2.4	SIMULATION FILE	8

1 INTRODUCTION

This document contains information about the files processed and generated by the simulator. The simulator operates with four file formats. The formats are listed below:

- Architecture – Contains the architecture of the biochip.
- Flow Path Set – Contains all available path on a biochip.
- Application Graph – Contains the operations that the biochip should perform.
- Simulation – Contains a simulation that the simulator should visualize.

All files are XML based. The file structure must be obeyed to get the simulator to work properly.

2 FILE FORMATS

This section contains information about each of the file formats mentioned in the introduction.

2.1 ARCHITECTURE FILE

This file holds information about the architecture of the biochip. The example below shows an empty architecture.

```
<Architecture>
  <ID></ID>
  <FlowRate></FlowRate>
  <UnitSize></UnitSize>
  <Height></Height>
  <Width></Width>
  <ListOfArcComponents>
  </ListOfArcComponents>
  <ListOfArcConnectors>
  </ListOfArcConnectors>
</Architecture>
```

The architecture file contains an *ID*, which is the name of the biochip. *FlowRate* is the speed (mm/s) that the fluid flows with inside the biochip. *UnitSize* represents the fluid unit size that one component in the biochip can contain. *Height* and *Width* represents the dimensions of the biochip. The last two elements (*ListOfArcComponents* and *ListOfArcConnectors*) in the architecture file contain the properties of the components and connections in the biochip.

A component is represented as shown below.

```
<ArcComponentProperties>
  <ID>Mixer2</ID>
  <ArcComponentType>arcmixer</ArcComponentType>
  <FlowRate>5</FlowRate>
  <UnitSize>500</UnitSize>
  <FlowExecutionTime>0.1</FlowExecutionTime>
  <Position>
    <X>188.5</X>
    <Y>178</Y>
  </Position>
  <Valves>
    <string>z24</string>
    <string>z25</string>
    <string>z26</string>
    <string>z27</string>
    <string>z27</string>
    <string>z28</string>
    <string>z29</string>
    <string>z30</string>
  </Valves>
  <IsRight>>false</IsRight>
  <IsTop>>false</IsTop>
  <IsLeft>>false</IsLeft>
  <IsBottom>>false</IsBottom>
</ArcComponentProperties>
```

Since all components in the simulator have the same property class associated with them, it could be, that some of the properties are not used or not set. The element *ID* represents the identity of the component. the identity should be unique. *ArcComponentType* represents the type of the component; this could be an *ArcMixer* or *ArcSwitch* just to mention some

examples. Again *FlowRate* represents the speed (mm/s) for a fluid inside the component. The *UnitSize* element represents the fluid unit size for the specific component. *FlowExecutionTime* is the calculated flow time from a fluid entering the component until it is inside the component. The *Position* element determines the placement of the component in the biochip. The element *Valves* is a list of strings representing the identities of the valves in the component. If the component is an input source or similar the input fluid is determined by the element called *Fluid*, which is represented by a *Color* and a *FluidName*. The last four elements hold *Booleans*, which determines whether the component is a left, right, top or bottom directed component. These four last properties makes it possible to construct an *ArcSwitch* were the channels could be hidden or visible depending on these properties.

A connection is represented as shown below.

```
<ArcConnectorProperties>
  <FlowExecutionTime>0.4</FlowExecutionTime>
  <FlowRate>5</FlowRate>
  <Component1>S4</Component1>
  <Component2>In2</Component2>
  <EndPointDirectionForComponent1>Both</EndPointDirectionForComponent1>
  <EndPointDirectionForComponent2>Output</EndPointDirectionForComponent2>
  <ConnectionPoint1>connectionPoint2</ConnectionPoint1>
  <ConnectionPoint2>connectionPointOutPut2</ConnectionPoint2>
  <LinePoints>
    <Point1>
      <X>119.5</X>
      <Y>402</Y>
    </Point1>
    <Point2>
      <X>39.5</X>
      <Y>403</Y>
    </Point2>
  </LinePoints />
</ArcConnectorProperties>
```

A connection has two endpoints. The endpoints are specified by a component identities (*Component1* or *Component2*) and an endpoint identity (*ConnectionPoint1* or *ConnectionPoint2*). The connection endpoints also have a direction (*EndPointDirectionForComponent1* or *EndPointDirectionForComponent2*) that can be *input*, *output* or *both*. This property help making the flow path set. A connection also has a *FlowExecutionTime*, which contains information about how long time it will take a fluid to flow through the channel. The *FlowRate* is the speed (mm/s). The last element *LinePoints* holds the points from a path from *component1* to *component2*. Which means that it is possible to make a path with corners.

2.2 APPLICATION GRAPH FILE

This file contains information about the application graph, which represents the operations that a biochip should execute. The example below shows an empty application graph.

```
<ApplicationGraph>
  <Connections>
  </Connections>
  <AppOperations>
  </AppOperations>
  <Width></Width>
  <Height></Height>
</ApplicationGraph>
```

A graph has a *width* and *height*, which determines the design space. *Connections* and *AppOperations* are lists that contain the actual graph.

A Connection example is shown below.

```
<AppOperationConnLineProperties>
  <AppOperation1>Source</AppOperation1>
  <AppOperation2>O2</AppOperation2>
  <Point1>
    <X>155</X>
    <Y>27</Y>
  </Point1>
  <Point2>
    <X>114</X>
    <Y>101</Y>
  </Point2>
</AppOperationConnLineProperties>
```

The connection is represented by two points at two different sources. In the example above, a connection from source to O2 is drawn, when the application graph is loaded into the simulator.

An operation example is shown below.

```
<AppOperationProperties>
  <ID>O2</ID>
  <Point>
    <X>89</X>
    <Y>96</Y>
  </Point>
  <InputSources>
    <string>Source</string>
  </InputSources>
  <Operation>0</Operation>
  <ExecutionTime>4</ExecutionTime>
</AppOperationProperties>
```

The operation has an ID that is used to identify the operation when simulating. It has a position represented by the *Point* element. An operation also has *InputSources* that represents the inputs to the operation; in this example the operation needs an input from source, before it can perform the actual operation. The element *Operation* determines which operation to execute.

The operations that are available:

0. Mix
1. Heat
2. Filter

The last element is the *ExecutionTime*, which specifies for how long the operation should run.

2.3 FLOW PATH SET FILE

This file contains information about the flow paths that are available on a specific biochip. The example below shows an empty flow path file.

```
<ArrayOfFlowPathSet>
</ArrayOfFlowPathSet>
```

The element type that the *ArrayOfFlowPathSet* contains is specified below.

```
<FlowPathSet>
  <Name>F1-1</Name>
  <FlowPath>
```

```
    <string>In1</string>
    <string>S1</string>
    <string>S2</string>
    <string>S3</string>
    <string>Mixer3</string>
  </FlowPath>
  <SinkPath>
    <string>In1</string>
    <string>S1</string>
    <string>S2</string>
    <string>S3</string>
    <string>Mixer3</string>
  </SinkPath>
  <ClosedValves>
    <string>z4</string>
    <string>z6</string>
    <string>z11</string>
    <string>z33</string>
    <string>z37</string>
  </ClosedValves>
  <OpenValves>
    <string>z3</string>
    <string>z5</string>
    <string>z12</string>
    <string>z34</string>
    <string>z38</string>
  </OpenValves>
  <FlowPathConstraints>
    <string>F1-2</string>
    <string>F2-1</string>
    <string>F2-2</string>
    ...
  </FlowPathConstraints>
  <ExecutionTime>1.8000000000000003</ExecutionTime>
  <Type>Flowpath</Type>
</FlowPathSet>
```

A *FlowPathSet* has a *Name* which is the identity of the flow path. The array of strings in *FlowPath* and *SinkPath* represents components that are used for in the specific *FlowPathSet*. The two string arrays *ClosedValves* and *OpenValves* represents the open and closed valves for the specific flow path. These valves must either be closed or open. *FlowPathConstraints* holds all the identities of flow path sets that cannot be executed at the same time as this specific flow path set. The *ExecutionTime* element contains the execution time for the whole flow path set. *Type* represents the type of the flow path. A flowpathset can either be an *Operation* or a *Flowpath*. This is relevant when the simulation file is loaded into the simulator.

2.4 SIMULATION FILE

This file contains information about the actual simulation. The simulation file should be loaded into the simulator together with an Architecture File and an Application Graph File.

The example below shows an empty simulation file.

```
<Simulation>
  <ListOfFlowPaths>
  </ListOfFlowPaths>
  <SimulationTime>
  </SimulationTime>
</Simulation>
```

A simulation file contains a list of flow path elements and a simulation time, which represents the run time for the whole simulation in seconds. The list of flow paths is represented by elements of *FlowPathSet*.

A flow path set example is shown below.

```
<FlowPathSet>
  <Name>F1-1</Name>
  <StartTime>0</StartTime>
  <Type>Flowpath</Type>
</FlowPathSet>
```

The element *Name* represents an identity or reference to a flow path from the flow path sets (See section 2.3). The element *StartTime* represents a point in time where the flow starts. The last element *Type* determines whether an operation or a flow path should be executed.

The flow path set example below shows how to execute an operation.

```
<FlowPathSet>
  <Name>04</Name>
  <FlowPath>
    <string>Mixer3</string>
  </FlowPath>
  <ExecutionTime>5</ExecutionTime>
  <StartTime>5.5</StartTime>
  <Type>Operation</Type>
</FlowPathSet>
```

Again the element *Name* is a reference. But this time *Name* is a reference to an operation from the application graph file (See section 2.2). The *FlowPath* element only contains one reference to a component performing the operation. *ExecutionTime* determines for how long the operation runs. The last element tells the simulator that flow path is an operation.

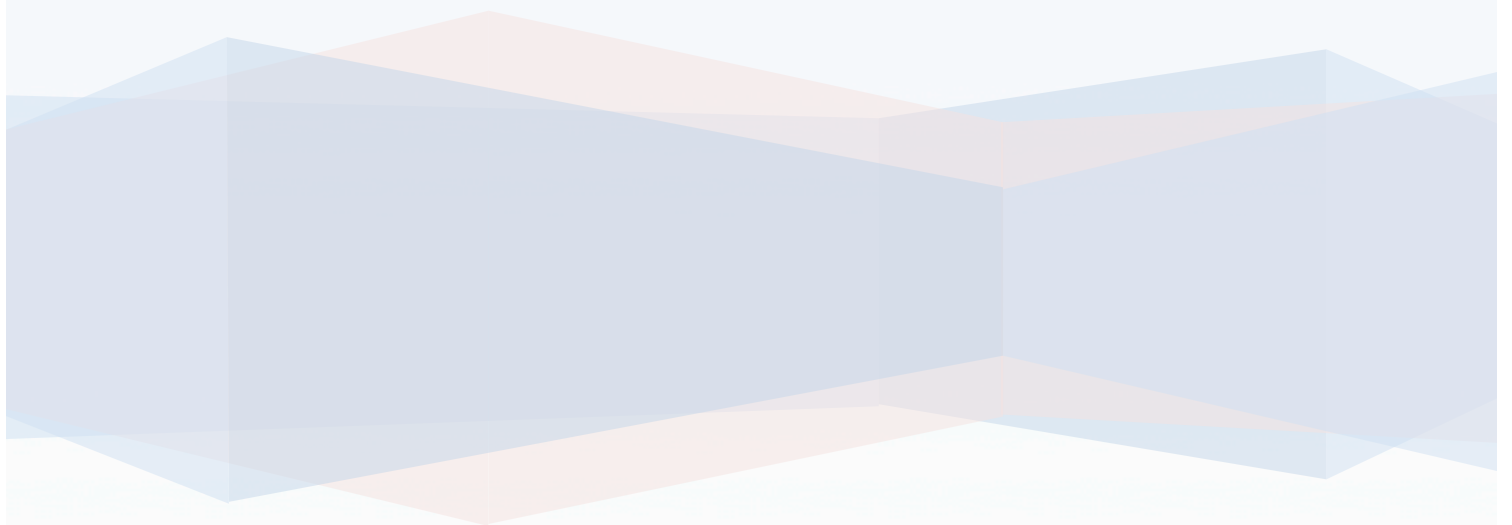
APPENDIX C BIOCHIP SIMULATOR – USER GUIDE

See documentation available on the thesis web site or on the next page.

BioChip Simulator

User Guide

Morten Foged Schmidt



This Page Intentionally Left Blank

Table of Contents

1	INTRODUCTION	4
2	THE BIOCHIP SIMULATOR	5
3	CREATE A BIOCHIP	6
4	CREATE APPLICATION GRAPH	10
5	LOAD AND SAVE	13
6	SIMULATION FEATURES	15
7	PLAY MENU	16
8	SHORT DEMONSTRATION	17
8.1	STEP 1 - START UP THE SIMULATOR	17
8.2	STEP 2 - LOAD FILES	17
8.3	STEP 3 - FEATURES	19
8.4	STEP 4 - VISUALIZING SIMULATION	19

1 INTRODUCTION

This document contains an introduction to the features and functions available in the BioChip Simulator. The demonstration at the end of this document can of course be read as a short introduction to the BioChip Simulator.

2 THE BIOCHIP SIMULATOR

This section describes the structure of in the BioChip Simulator. Below is a screen dump showing the overall user interface.

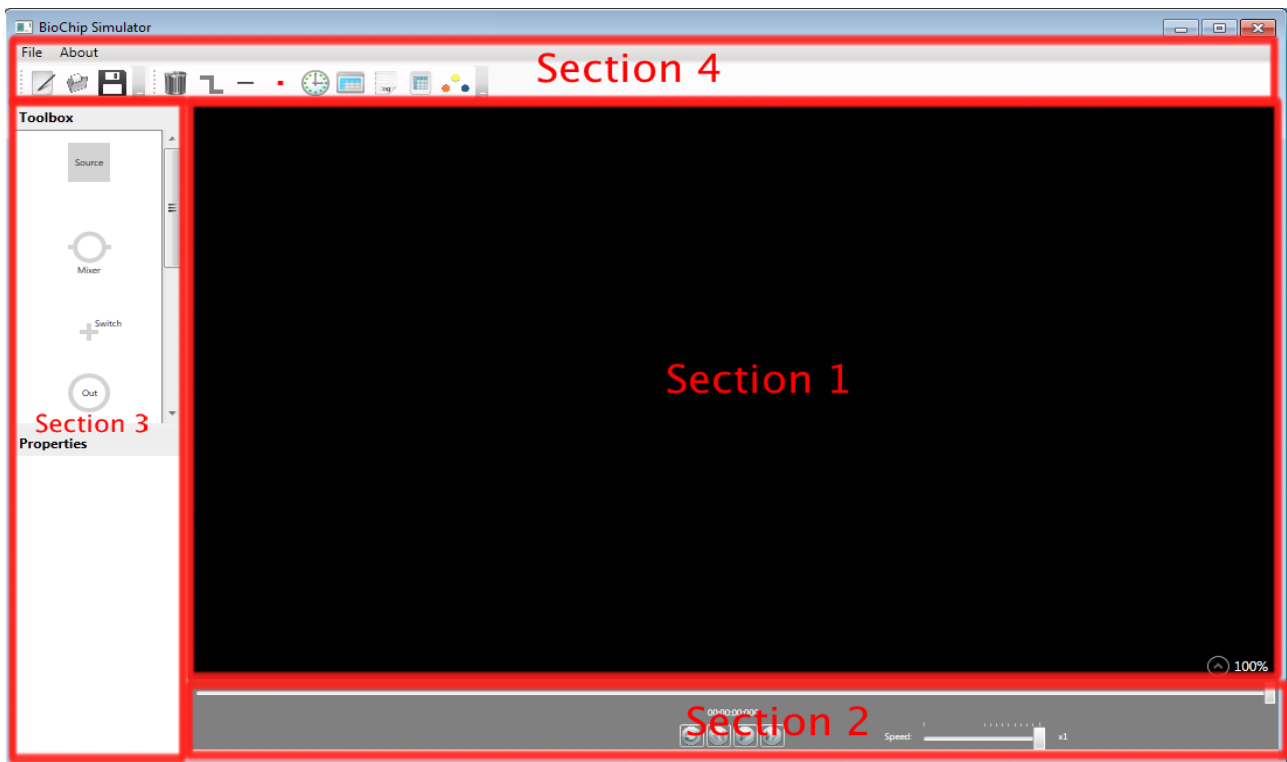



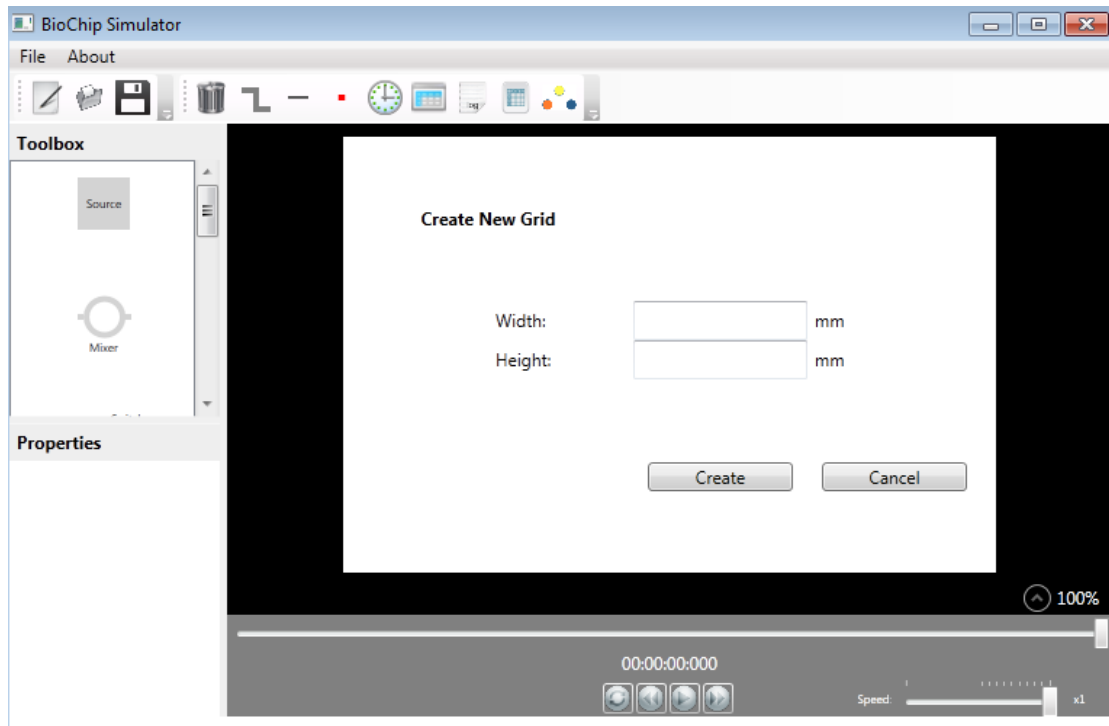
Figure 1- BioChip Simulator structure

The application is divided into 4 sections, which is described in the table below.

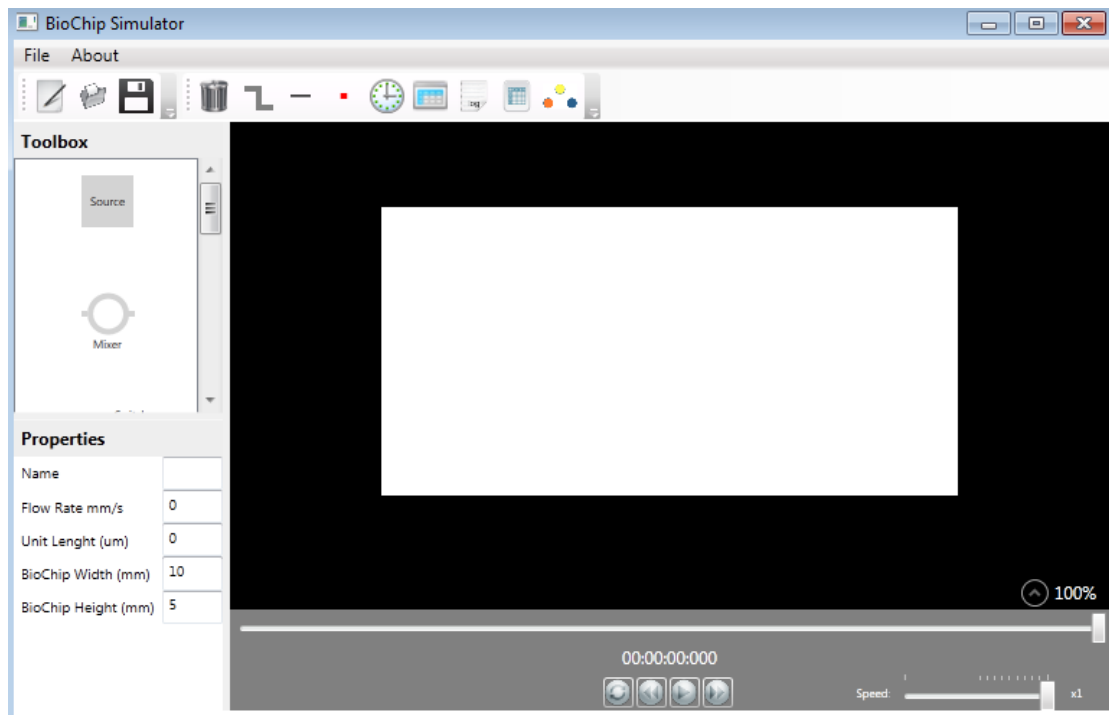
Section	Description
1	Here users are allowed to edit a biochip by drag and drop and show a simulation visually. It is also possible zoom in and out.
2	This section contains a play menu. Here it is possible to set the speed of a simulation and step back and forward.
3	This section contains a toolbox and a property view. The toolbox contains all available biochip components except the connection channel. The property view allows users to change components.
4	The section is a toolbar that enables all kinds of features to be switch on and off.

3 CREATE A BIOCHIP

By clicking the button, , at the toolbar in section 4 a new view will show up in section 1. The view looks as shown below.



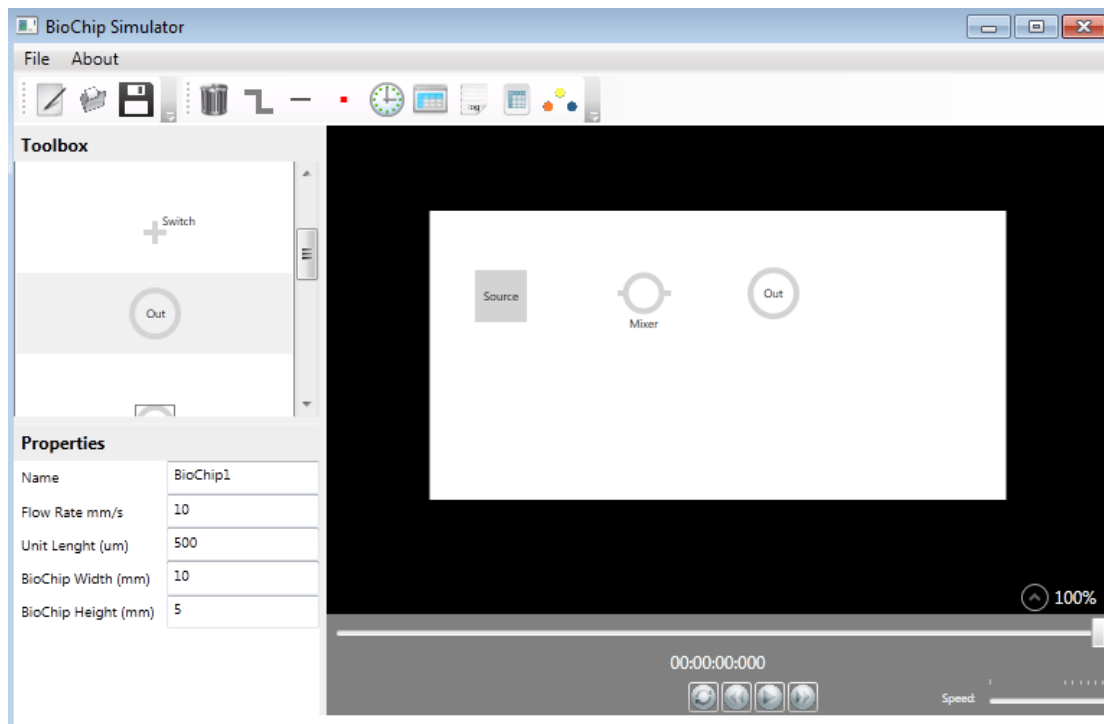
The new view will ask for the dimensions of the biochip. When the dimensions have been entered and the create button has been clicked a new biochip drawing board will show in section 1. The window is shown below.



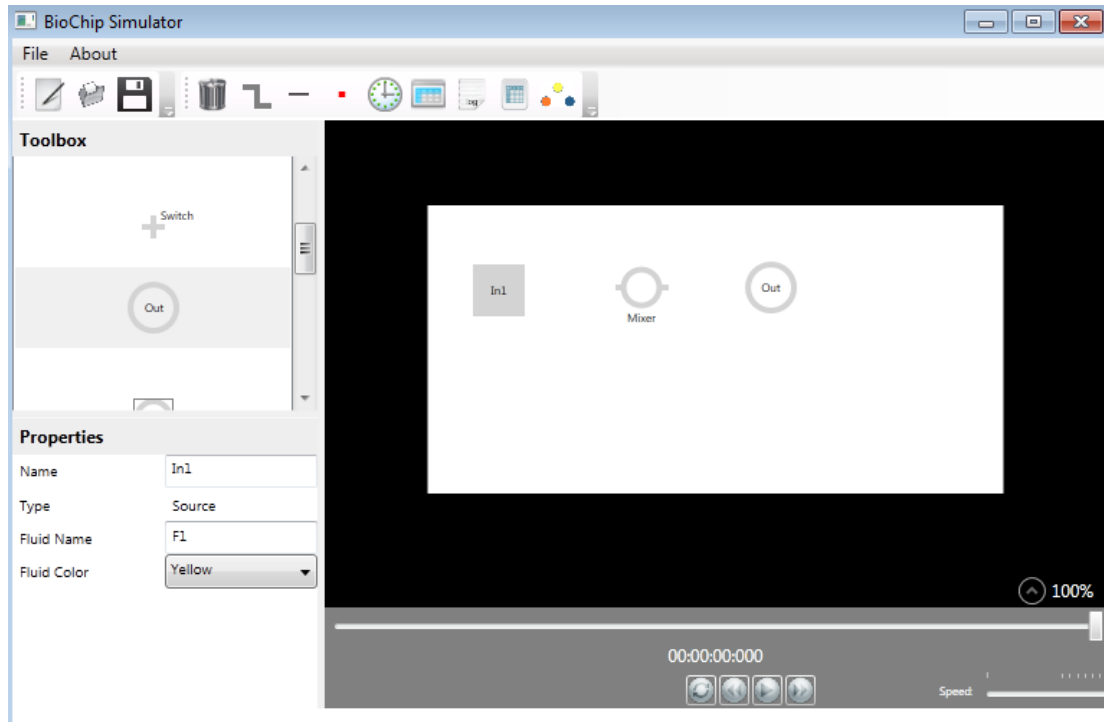
Now it is possible to give the biochip a name, flow rate and unit length, as shown below.

Properties	
Name	BioChip1
Flow Rate mm/s	10
Unit Length (um)	500
BioChip Width (mm)	10
BioChip Height (mm)	5


When the steps have been performed the biochip architecture is ready to be created. Dragging components from the component toolbox does this. When the desired components has been placed the biochip could look as shown below.

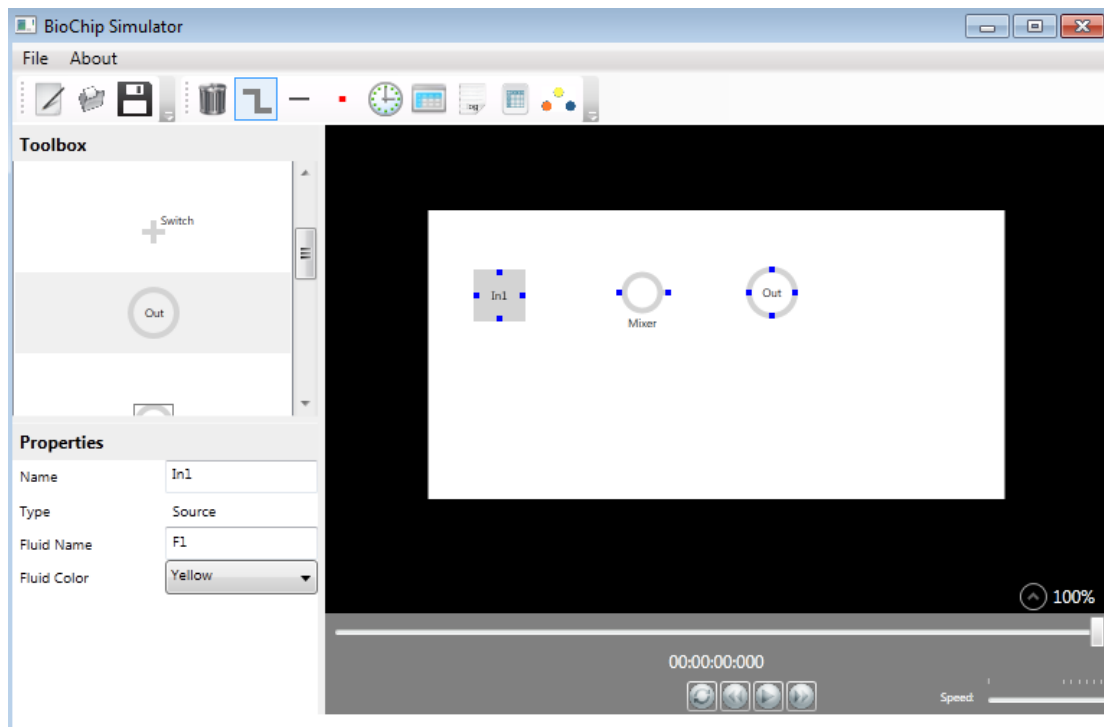


Now we have to name the components and give the pressure source connected to then an identity. By clicking the components the property view changes as shown below.

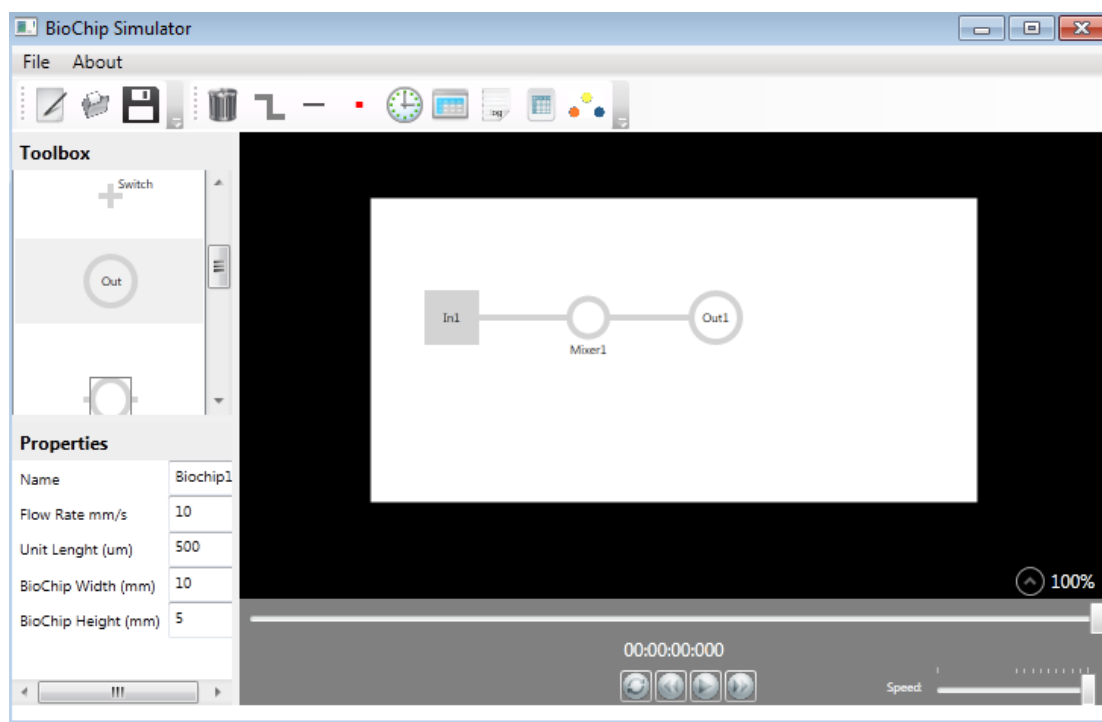


When all properties has been set we are ready to connect the components.

This is done by clicking the  button in the toolbar. As shown below all available connection points will be visible.




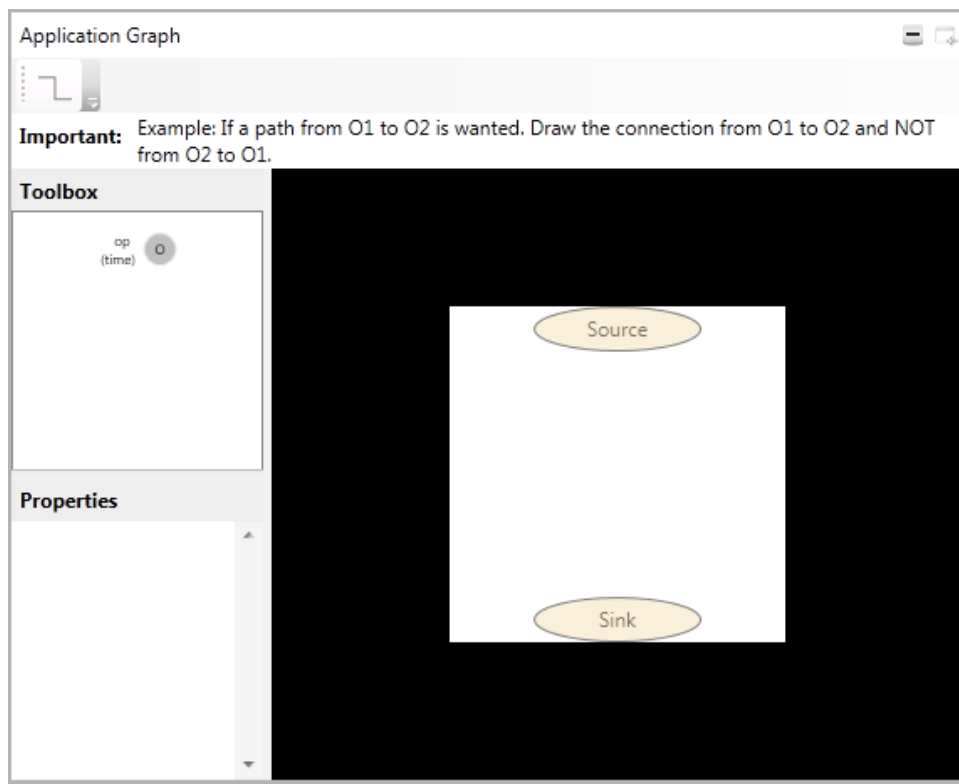
By dragging the mouse and holding down the mouse button we can now create connection channels. When the network of connection have been created the biochip architecture will look as shown below.



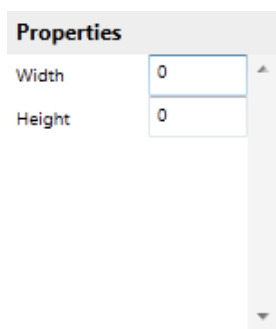
Now we have a biochip architecture.

4 CREATE APPLICATION GRAPH

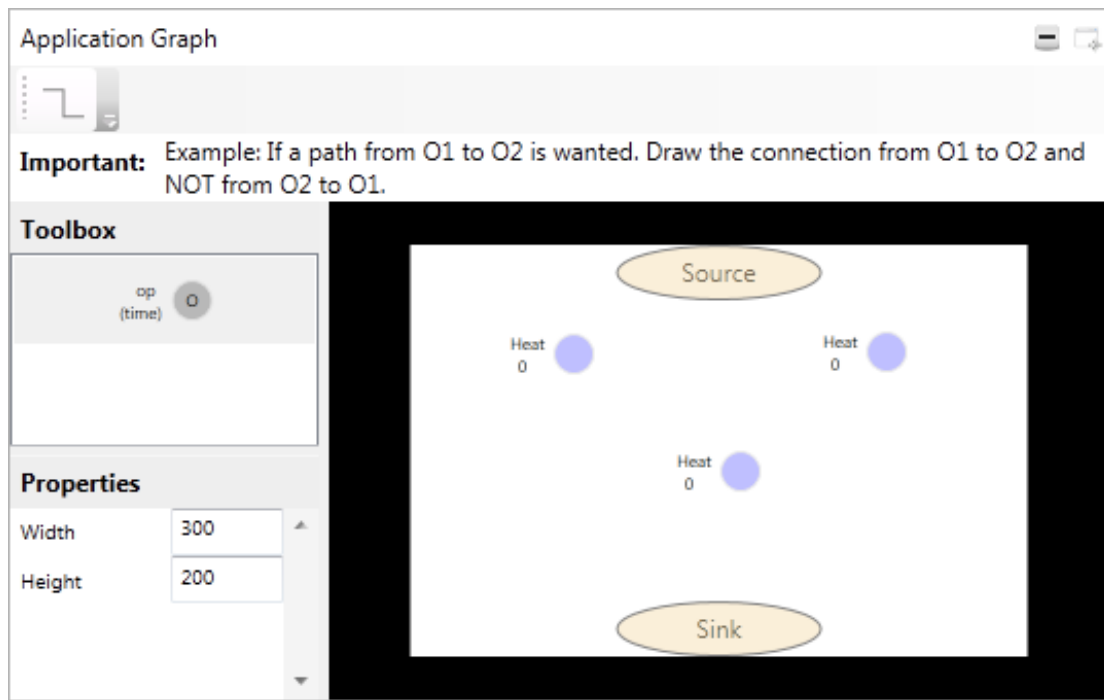
Start creating the chemical applications by clicking the  button in the toolbar. When the button has been click a new window will show up.



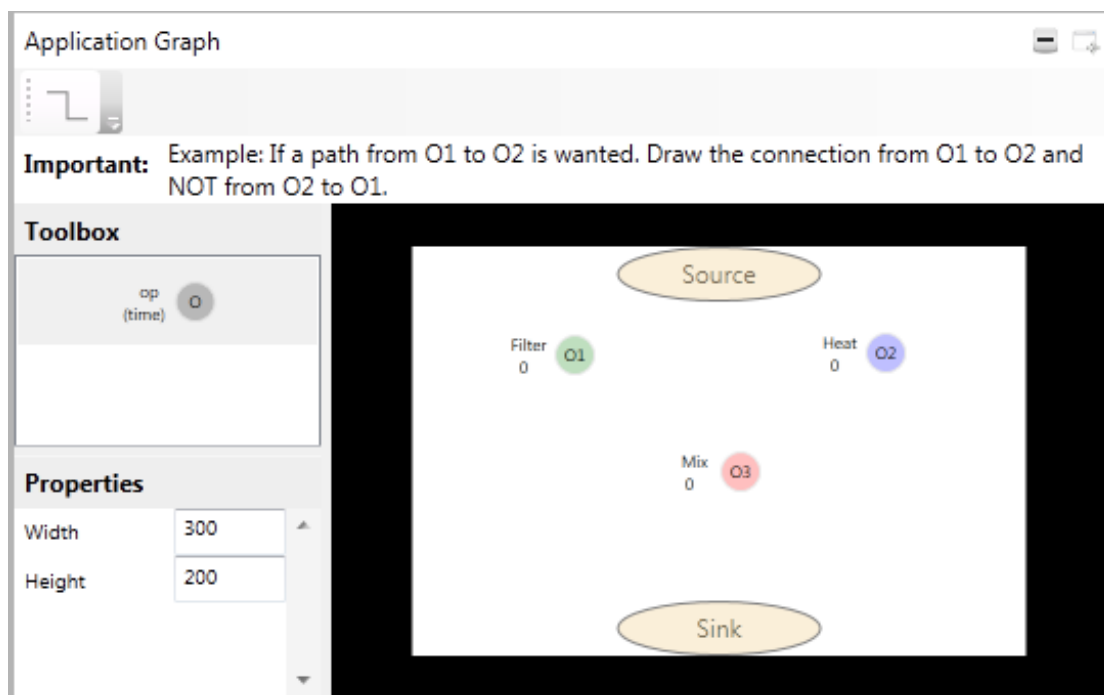
By clicking the drawing board in a property view will show up to the left. Here the width and height of the drawing board can be set. The property view is shown below.




When a width and height has been set. Operation components can be dragged into the drawing board. When the three operation has been placed the window should look as shown below.



By clicking the components the property view changes and allows the operations to be changed. It is also possible to change the identity and execution time. When this has been done the window should look as shown below.



Now the connections or predecessors can be set by clicking the  button in the toolbar. Dragging the mouse between the operation components while holding down the mouse button sets the predecessors. When all predecessor has been set the graph should look as shown below.

Application Graph

Important: Example: If a path from O1 to O2 is wanted, Draw the connection from O1 to O2 and NOT from O2 to O1.

Toolbox

op (time) ○


Properties

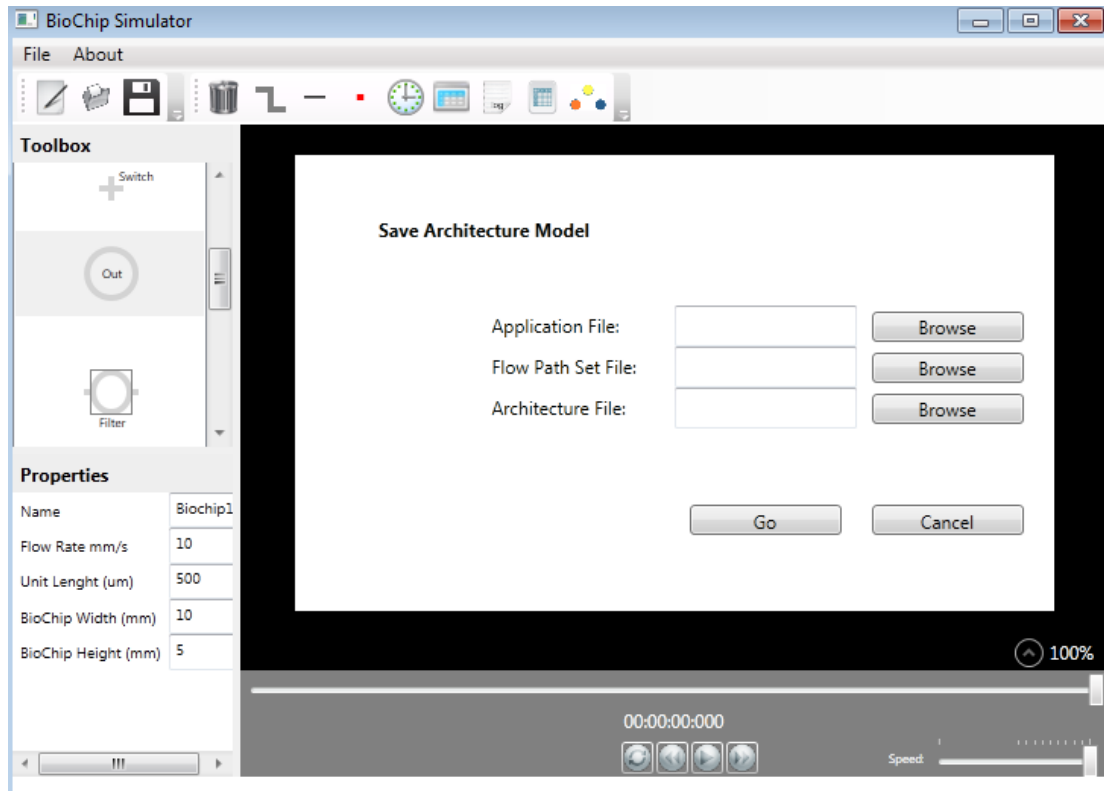
Width 300

Height 200

```
graph TD; Source([Source]) --> O1((O1)); Source --> O2((O2)); O1 -- Filter 2 --> O2; O2 -- Heat 3 --> O3((O3)); O1 -- Mix 5 --> O3; O3 -- Mix 5 --> Sink([Sink]);
```

5 LOAD AND SAVE

Biochip architectures and chemical applications can be saved by clicking the  button. When the save button is clicked a view in section 1 will appear. The window will look as shown below.

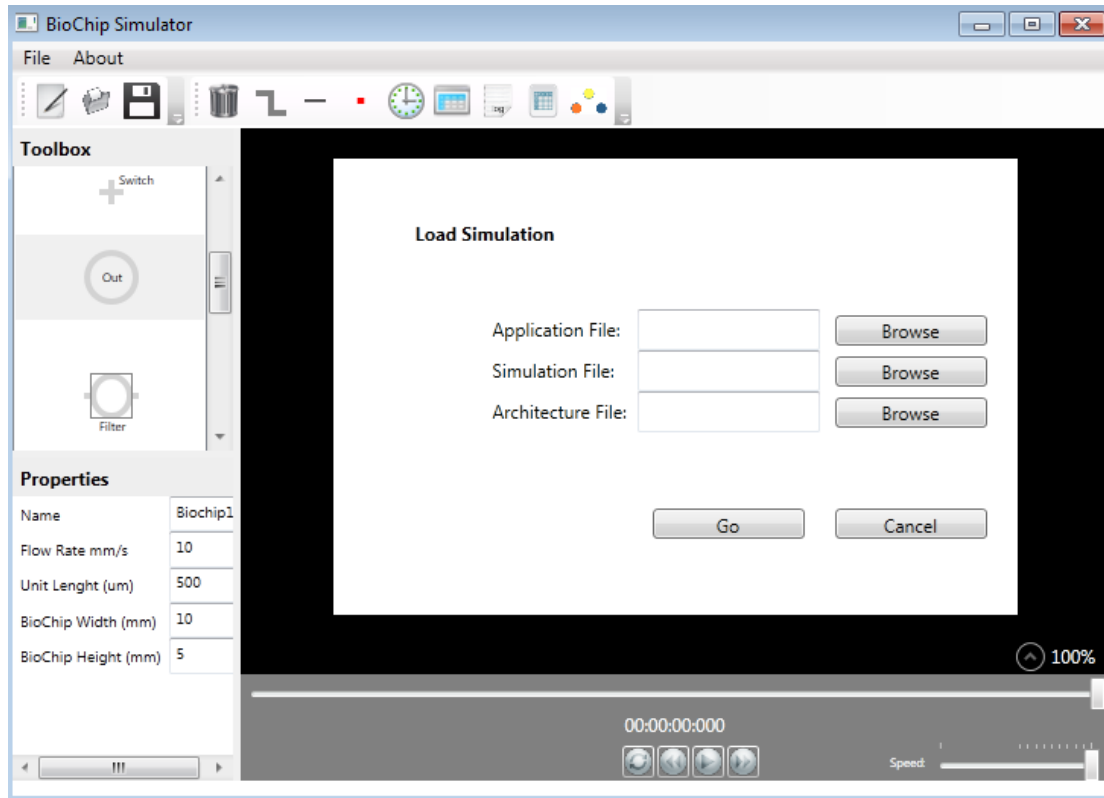


Clicking the browse buttons will allow saving in a location of the user's choice. The save view allows saving three kinds of files.

- Chemical or biochemical applications
- Biochip architectures
- Flow path set file

The flow path set file is a file containing all flow possibilities from a biochip architecture.

Loading files is done by clicking the  button. When the load button is clicked a new view in section 1 will appear. The window will look as shown below.



Clicking the browse buttons will allow loading from a location of the user's choice. The load view allows loading of three kinds of files.

- Application file
- Simulation file
- Architecture file

The simulation file is a file that is produced by a scheduler or another software tool that obeys the standards of the BioChip Simulator.

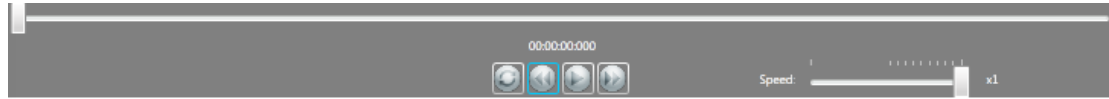
6 SIMULATION FEATURES

When a biochip architecture, chemical or biochemical application and a simulation is loaded into the simulator or created within the simulator. The simulator has the following features, which can be activated from the toolbar in section 4.

Feature	Icon/click	Description	Placement
Biochip properties	No icon	In the lower left corner the properties will be shown. (Flow Rate, Unit Size)	Click the white space in the biochip
Flow Path Sets		Show the flow paths available on the specified chip.	Button is available in the toolbar
Log		Shows what happens on the chip while simulating	Button is available in the toolbar
Valves		Shows the valves states for the whole simulation	Button is available in the toolbar
Application Graph		Shows the application graph for the simulation	Button is available in the toolbar
Execution time		Shows the time that a fluid flows in the component. The information is shown on the biochip components.	Button is available in the toolbar
Control Layer		Shows the control layer while simulating.	Button is available in the toolbar
Sink Layer		Shows the sink layer while simulating	Button is available in the toolbar

7 PLAY MENU

When a simulation, application and architecture file has been loaded and the features are selected. The simulation can start. The simulation is controlled from the Play menu available in section 2. The play menu is shown below.



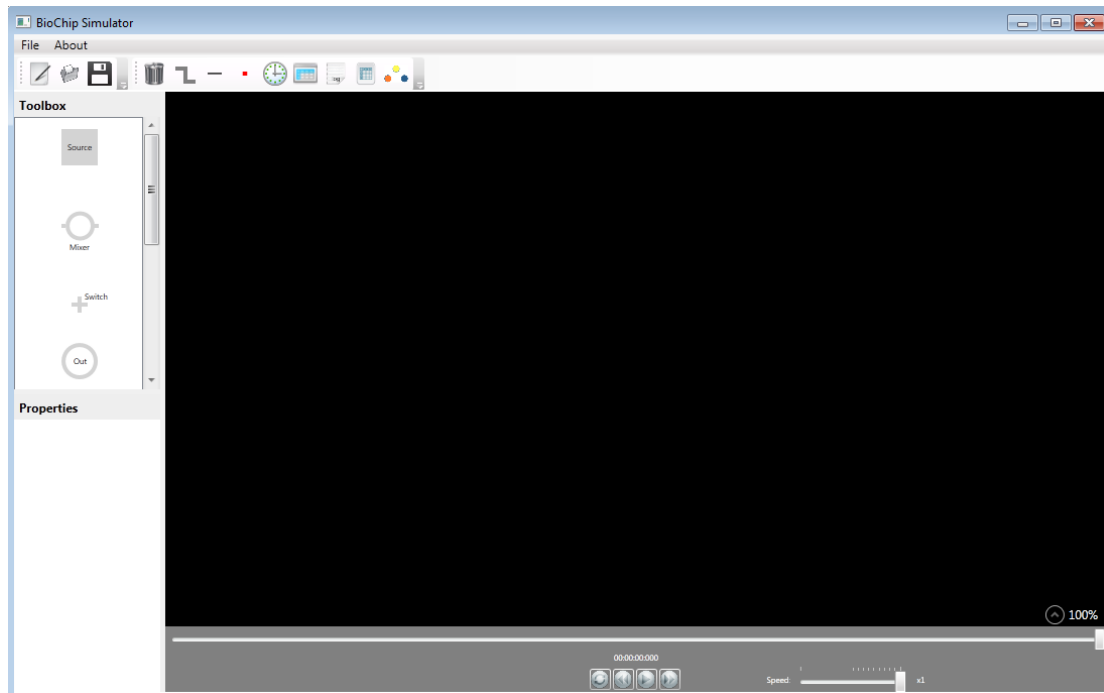
The play menu allows the simulation to be played in slow motion by set the speed using the slider. Clicking the refresh button will reset the simulation. It is also possible to step trough the simulation using the step buttons, on each side of the play button.

8 SHORT DEMONSTRATION


The demonstration is divided into 4 steps and requires the simulator.zip file from the thesis home page <https://sites.google.com/site/biochipsimulator/Files> to be downloaded. When the files have been extracted you will be able to start step 1. The demonstration can of course also be read as a short introduction to the tool.

8.1 STEP 1 – START UP THE SIMULATOR

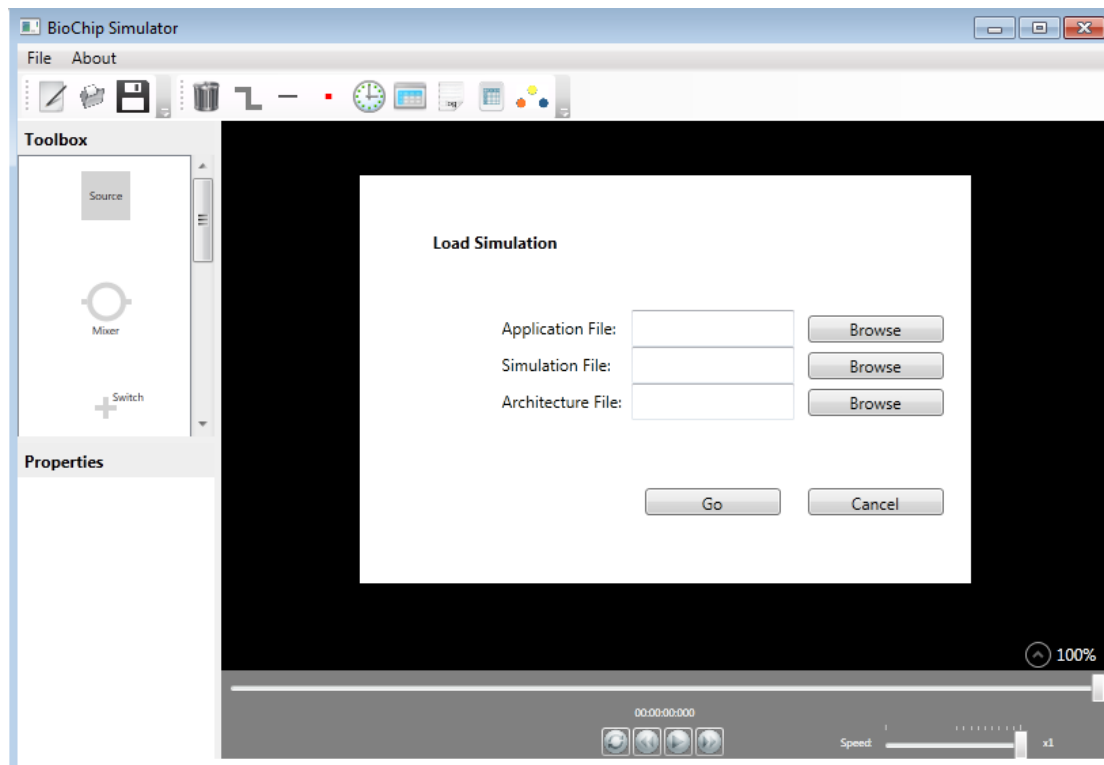
Start the simulator by executing the file named Simulator.exe. When the simulator is running correct you will see a screen as shown below.



8.2 STEP 2 – LOAD FILES

When the program is running click the load files button . The button is available in the toolbar at the upper left corner.

When the load button has been clicked a load screen will show up. See the screen dump below.

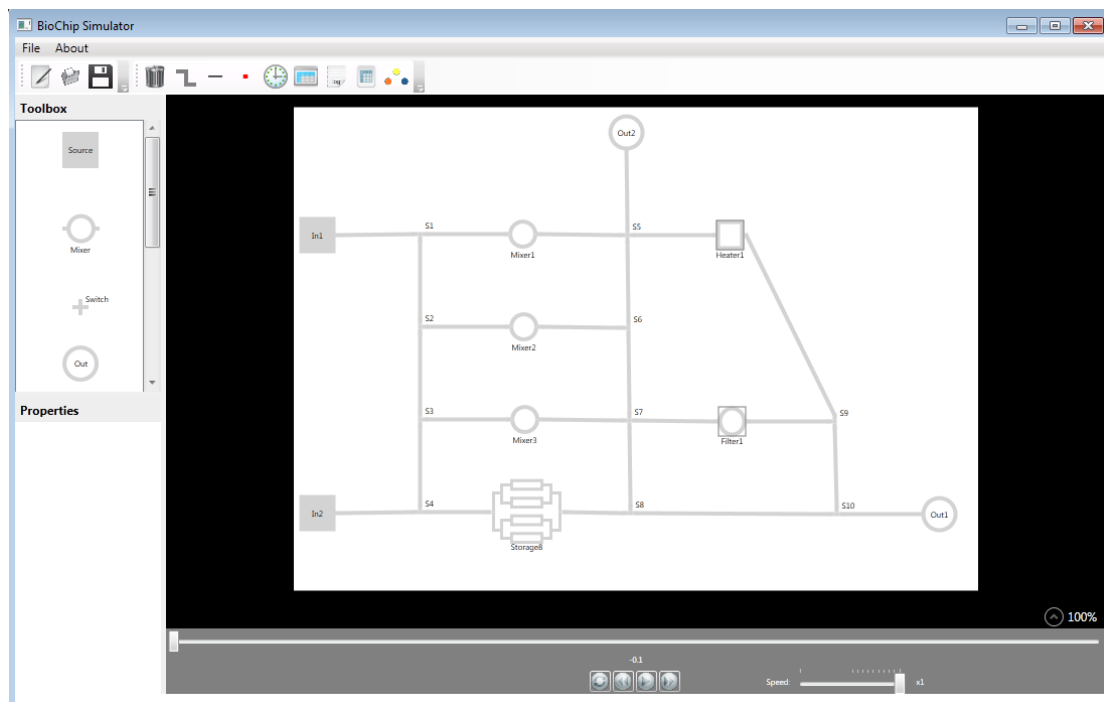


Now load the three files listed below into the simulator.

- Application File: ArchitectureDemo.arc
- Simulation File: ApplicationDemo.app
- Architecture File: SimulationDemo.sim

Search for the files, by clicking the browse buttons available on the load screen. When all files have been selected, click the Go button.

The simulator will now load all the files and show the architecture of the biochip.



8.3 STEP 3 - FEATURES

It is now possible to show all kinds of information about the biochip, application and simulation.

Below is a table that shows and describes some of the features available in the simulator.

Feature	Icon/click	Description	Placement
Biochip properties	No icon	In the lower left corner the properties will be shown. (Flow Rate, Unit Size)	Click the white space in the biochip
Flow Path Sets		Show the flow paths available on the specified chip.	Button is available in the toolbar
Log		Shows what happens on the chip while simulating	Button is available in the toolbar
Valves		Shows the valves states for the whole simulation	Button is available in the toolbar
Application Graph		Shows the application graph for the simulation	Button is available in the toolbar
Execution time		Shows the time that a fluid flows in the component. The information is shown on the biochip components.	Button is available in the toolbar
Control Layer		Shows the control layer while simulating.	Button is available in the toolbar
Sink Layer		Shows the sink layer while simulating	Button is available in the toolbar

8.4 STEP 4 – VISUALIZING SIMULATION

Now when the simulation, application and architecture files are loaded and the features are selected. The simulation can start. The simulation is controlled from the Play menu available in lower central area. A screen dump of the play menu is shown below.



Here you are able to click play, step forward and step backwards (Backwards does not work properly yet, demixing has not been implemented). From the play menu the speed of the simulation can be adjusted and it is also possible to restart the simulation. The features can be adjusted as wanted while simulating. A screen dump from a successful simulation is shown below.

