



Volume Management for Pin-Constrained Continuous-Flow Microfluidic Biochips

Schneider, Alexander Rüdiger

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Schneider, A. R. (2018). Volume Management for Pin-Constrained Continuous-Flow Microfluidic Biochips. Technical University of Denmark. DTU Compute PHD-2018, Vol.. 479

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Volume Management for Pin-Constrained Continuous-Flow Microfluidic Biochips

Alexander Schneider



Kongens Lyngby 2018

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

Microfluidic Biochips integrate functional components for biochemical analysis and sample preparation on a single chip. The implementation of conventional biochemical laboratory equipment onto a chip at sub-millimetre scale offers several advantages. This thesis is focused on continuous-flow based biochip technology. Here, valves are used as basic building blocks to route reagents as desired along channels. Combining multiple valves allows to build more complex components to mimic the functionality of their conventional laboratory counterparts such as centrifuges, incubators or analysers.

Fabrication capabilities of biochips have seen a rapid growth in the last years, further miniaturizing valves to just few μm in size and thereby allowing thousands of components to be placed on a single chip. To keep up with the manufacturing technology, new design methodologies are required, and researchers have started to propose methods and tools for the physical design and programming of biochips. However, these tools have considered simplified models and unrealistic assumptions regarding other parts of the design process. To support the practical use of biochips, we have to consider the practical aspects that bridge the gap from research to functional verification and fabrication.

Even though thousands of valves can be integrated on a biochip, their functionality is limited due to the number of control-pins that can be connected to an external controller. Hence we have proposed a pin-count reduction technique to reduce the number of required external controls using shared valve actuations. Previous work has not considered the impact of such pin-count reduction on the binding, routing and scheduling, resulting in suboptimal results. Furthermore, the problem of fluid management is often ignored for microfluidic applications, assuming that all fluidic constraints are satisfied. However, dispensing additional fluid (overflow) can be expensive, while shortages of fluid (underflow) can interrupt the execution and even require manual replenishment. Therefore we provide a volume management approach

that satisfies all constraints while improving the efficiency of mixing operations. Furthermore, moving fluids within a biochip is simply referred to as fluid transport by researchers, ignoring the underlying complexity. Realistic biochip design has to offer the capabilities to provide specific volumes of fluid during the biochip's operation. We therefore propose a component that provides metering, alignment and missing fluid detection capabilities for fluid transport.

Current tools for physical design do not bridge the fabrication gap, which extends into lacking functional verification. Instead, interdisciplinary specialist knowledge is required to verify, potentially adapt and then turn the synthesized design into working biochips. Hence, in this thesis we consider three-layer normally-closed valve biochips that use two micromilled Poly-methylmethacrylate (PMMA) layers sandwiching a sheet of Polydimethylsiloxane (PDMS), which are easier to fabricate. We have developed a software tool towards a more automated biochip design process, reducing the amount of manual and error-prone work required. Aiming to reduce the interdisciplinary knowledge required by the end-user to fabricate biochips, the tool generates G-Code from the created design usable for fabrication of the biochip's PMMA layers in any micromilling machine.

Summary (Danish)

Traditionelle biokemiske laboratorie analyser kan i dag integreres på en enkelt biochip ved hjælp af mikrofluidiske komponenter, hvilket giver flere fordele. Denne afhandling fokuserer på kontinuerlig flowbaseret biochip teknologi. Her anvendes ventiler som grundlæggende byggesten til at bestemme flowet af små volumener af væsker i små kanaler. Ved at kombinere flere ventiler kan der bygges mere komplekse komponenter som efterligner funktionaliteten af deres konventionelle laboratorie modstykker, så som centrifuger, inkubatorer og analyser. Fabrikation af biochips har oplevet en stigende vækst i de senere år og den fortsatte miniaturisering af ventiler til nogle få μm i størrelse, tillader at tusindvis af komponenter integreres på en enkelt chip. For at kunne følge med denne produktionsteknologi er der brug for nye designmetoder, og forskere er begyndt at foreslå metoder og værktøjer til fysisk design og programmering af biochips. Disse værktøjer har imidlertid overvejende anvendt forenkede modeller og urealistiske antagelser vedrørende andre, men centrale, dele af designprocessen. For at understøtte de praktiske anvendelser af biochips, skal vi også overveje de praktiske aspekter, der er nødvendige for at komme fra forskning til funktionel verifikation og fabrikation.

Selvom tusindvis af ventiler kan integreres på en biochip, er deres anvendelighed begrænset af hvor mange kontrol kanaler der kan tilsluttes en ekstern kontrol enhed. Vi foreslår derfor en reduktionsteknik, hvor kontrol input til ventiler deles mellem så mange ventiler som muligt for at reducere antallet af nødvendige eksterne kontrol kanaler. Tidligere arbejde har ikke taget højde for en sådan reduktionsteknik i forhold til resurse binding, rutning og tidsallokering, hvilket resulterer i suboptimale resultater. Endvidere ignoreres problemet med at organisere og styre væskemængder ofte i mikrofluidiske applikationer, idet det antages at alle fluidiske begrænsninger er opfyldt. Imidlertid kan dispensing af yderligere væske (overløb) være dyrt, mens mangel på væske (underløb) kan ødelægge udførelsen og endda kræve

manuel genopfyldning. Vi foreslår derfor en mængdehåndteringsmetode, der opfylder alle begrænsninger, samtidig med at effektiviteten af blandingsoperationer forbedres. Desuden refererer forskere typisk flytning af væskemængder som simple transporter, og dermed ignorerer den underliggende kompleksitet. Et realistisk biochip design skal kunne tilbyde at levere specifikke væskevolumener under hele procesafviklingen på biochipen. Vi foreslår derfor en ny komponent, der kan levere måling, justering og detektering af manglende væske for selve væsketransporten. Nuværende værktøjer til fysisk konstruktion af biochips, adresserer ikke fabrikationsgabet, som også omfatter manglende funktionel verifikation. I stedet kræves tværfaglig ekspertviden for at kunne verificere, potentielt tilpasse, og derefter syntetiserer en funktionsdygtig biochip. Vi anvender derfor i denne afhandling en biochip bestående af tre-lags normalt-lukkede ventiler, der bruger et ark polydimethylsiloxan (PDMS) som membran mellem to polymethylmethacrylat (PMMA) lag, i hvilke kannaler og ventiler er micro-fræset. Denne teknik letter fremstillingsprocessen. Vi har udviklet et softwareværktøj som automatiserer biochip designprocessen, hvorved mængden af manuelt og fejlbehæftet arbejde, reduceres. For at reducere mængden af specialviden omkring biochip fremstilling som kræves af slutbrugeren, genererer værktøjet direkte den G-Code, som bruges til fremstilling af biochippens PMMA-lag for enhver micro-fræse maskine.

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science, of the Technical University of Denmark in partial fulfilment of the requirements for acquiring the Ph.D. degree in engineering.

The presented work is part of the “IDEA4CPS: Foundations for Cyber-Physical Systems” project which is funded by the Danish National Research Foundation.

The thesis’ proposes are optimization techniques for both the design and fabrication of continuous-flow microfluidic biochips. The work has been supervised by Professor Paul Pop and co-supervised by Professor Jan Madsen.

Kongens Lyngby, June 2018

Alexander Schneider

Notations and Abbreviations

DBMB Droplet Based Microfluidic biochip

FBMB Flow Based Microfluidic biochip

FFU Functional Fluidic Unit

FP Flow Path

FPS Flow Path Segment

a Acceleration

t Time

F Force

PDMS Polydimethylsiloxane

PMMA Polymethyl methacrylate

Via Layer Change

VG Valve Group

IVD In-Vitro Diagnostics

MVR Minimum Volume Requirement

MHC Maximum Hardware Capacity

HTR Hardware Transport Resolution

FVA Fluid Volume Assignment

VEDV Volume Error Detection Vent

AV Alignment Vent

LOF Left Over Fluid

LoC Lab-on-a-Chip

BDT Biochip Designer Tool

Contents

Summary (English)	i
Summary (Danish)	iii
Notations and Abbreviations	vii
1 Introduction	1
1.1 Microfluidic Biochips	1
1.2 State-of-the-art and Commercialization	3
1.2.1 Biochip Design and Development	4
1.3 Motivation	5
1.4 Thesis Overview and Contributions	7
2 Biochips	11
2.1 Passive Biochips	11
2.2 Valve-based Biochips	12
2.2.1 Quake Valves	13
2.2.2 Grover Valves	14
2.3 Fluidic Components	15
2.4 Biochip Architecture Model	19
2.5 Biochip Application Model	21
2.6 External Hardware	22
2.6.1 Control Logic	24
3 Design Methodologies	27
3.1 Binding, Routing and Scheduling	29
3.2 Component allocation and schematic design	32
3.2.1 Microfluidic HDL	32
3.3 Faults and Fault Tolerance	33

3.4	Fabrication	37
3.4.1	Multi-layer milling and bonding	38
3.4.2	G-Code	41
3.5	Programmability	42
3.6	On-Chip Control	44
3.6.1	Layer Changes	45
4	Pin-Count Reduction	47
4.1	Introduction	47
4.2	Problem Formulation	49
4.3	Proposed Method	52
4.3.1	Reducing the Pin-Count	54
4.4	Experimental Results	60
4.4.1	Actuation of multiple valves	62
4.5	Conclusions	63
5	Waste-Aware Volume Management	65
5.1	Introduction	65
5.2	Problem Formulation	68
5.3	Mixing	69
5.4	Volume Management	73
5.4.1	Optimizing the mixing process	76
5.4.2	Deriving mixing trees for 1:1 mixing hardware	78
5.5	Experimental Evaluation and Discussion	81
5.6	Conclusion and Future work	84
6	A Novel Metering Component for Volume Management	87
6.1	Introduction	87
6.2	Venting	88
6.3	Alignment and Metering	90
6.4	Detecting missing fluid	91
6.5	Experimental results	93
6.5.1	Setup	93
6.5.2	Vent Architecture	95
6.5.3	Fluid displacement	97
6.5.4	Increasing the throughput	99

<i>Contents</i>	xi
6.5.5 Detecting insufficient volumes	99
6.5.6 Arbitrary ratio mixing using vents	100
6.6 Conclusion	101
7 Discussion and Conclusions	103
7.1 Outlook and Future Work	106
A Biochip Designer Tool and Fabrication	107
A.1 Fabrication Example	111
A.2 PMMA preparation	113
Bibliography	117

List of Figures

Chapter 1

1.1	Number of publications related to microfluidics [90]	2
1.2	Two types of biochips	3
1.3	Electronic vs. Microfluidic VLSI design process [9]	6

Chapter 2

2.1	Passive mixing architectures	12
2.2	Orientation-based biochip [88]	13
2.3	Schematic design of a Stanford valve [32]	14
2.4	Schematic design of a Grover valve	15
2.5	Switches	15
2.6	5 stage pump actuation using 3 Grover valves. Yellow lines are flow channels. Light blue circles are displacement chambers. Dark blue circles are displacement chambers filled with fluid.	16
2.7	5-Phase equal ratio mixer	17
2.8	Schematic design for any component requiring a reaction chamber such as heaters and detectors	18
2.9	In- and outlets connected with flexible tubes	19
2.10	Storage component with four storage channels. The number of channels can vary to suits the architectures needs	19
2.11	Architecture model with three inputs and outputs, one heater, filter and detector and six switches to interconnect the channels leading to the components	20
2.12	Application model with two fluid sources, one mixing and two incubate and detect operations	22
2.13	Biochip to controller adapter. Tubes from the controller are connected to the adapter, into which a biochip can be placed (System from Microfluidic Innovations Inc, IN, USA)	23

2.14 Pressure controller capable of providing high- and vacuum-pressure through 36 connectors using solenoid valves (System from Microfluidic Innovations Inc, IN, USA)	24
2.15 Grid of 4 switches. Dark blue lines indicate fluid flow.	25
2.16 Schematic design of a 3-bit 8-way multiplexer	26
2.17 Pressure / Vacuum Latch-Valve [43]	26

Chapter 3

3.1 Overview of the design of FBMBs	28
3.2 Microfluidic components	29
3.3 Schedule example for the application from Fig. 2.12 running on the architecture from Fig. 2.11	30
3.4 Alternative solution to the schedule in Fig 3.3	31
3.5 Common Faults in FBMBs [50]	33
3.6 Fault Tolerance through redundant architecture design [51]	35
3.7 Fault Tolerant Components [29]	36
3.8 Minitex Machinery Mini-Mill/3 with PMMA piece aligned to a bracket	39
3.9 G-Code examples	40
3.10 Ambiguous assay description [7]	40
3.11 Code examples [6]	41
3.12 Biocoder code example [7]	42
3.13 Gates	43
3.14 PCR Biochip with two Mixers, eight capacity storage and two in- and outputs, optimized to reduce the number of required control-pins with on-chip control [48]	44
3.15 Schematic design of the use of Vias to avoid an intersection of two channels on the same layer	45

Chapter 4

4.1 The proposed pin-count reduction technique is implemented in the three highlighted design steps, control synthesis and the physical design on both layers	48
4.2 IVD Architecture	49
4.3 Typical IVD application that mixes various samples, reagents and buffers and analyses the results	49

4.4	Impact of pin-count reduction on IVD schedule	50
4.5	Design steps used by the proposed method	53
4.6	Valves	54
4.7	Valve Groups	55
4.8	Routes	55
4.9	Multi-purpose architecture	57
4.10	Multi-purpose architecture application	58
4.11	Trade-off for TC4 using PCM (blue, squares and line). CSO result reference for TC4 (red dot)	60
4.12	Mixer with three pairs of valves using shared pressure sources	63

Chapter 5

5.1	The proposed volume management optimization is implemented in the highlighted design step application graph generation	67
5.2	Architecture netlist including two mixers and detectors with a maximum hardware capacity (MHC) of 10 nl each.	67
5.3	Metering example using a metering chamber	68
5.4	Application graph showing the input requirements of the operations as ratios for mixing operations (red) and as discrete values for the other type of operations (green).	69
5.5	Example of achieving a 1:3 mixing ratio using 1:1 mixing hardware and the unavoidable byproduct of fluid (i.e. waste) not in the target ratio.	70
5.6	Comparison of mixing trees for a target ratio of 5:11 created by Min-Mix, REMIA and NFB	70
5.7	Comparison of mixing trees for a target ratio of 5:11 created by Min-Mix, REMIA and NFB	71
5.8	Application from Fig. 5.4 with fluid volume according to each operations MVR assigned, leading to shortage of fluid at O_6	71
5.9	Application from Fig. 5.4 with fluid volume according to each operations MVR assigned, leading to shortage of fluid at O_6	73
5.10	Application from Fig. 5.4 with optimal assignment of fluid according to the calculated FVAs.	75
5.11	Application from Fig. 5.4 with optimal assignment of fluid according to the calculated FVAs and leftover fluid from O_1 has been reused as input for O_2 . . .	76

5.12	Application graph for a glucose test. Black labels indicate the required ratios and discrete volumes, red labels indicate the FVAs.	77
5.13	Optimized application graph from Fig. 5.12, where left over fluid from O_2 and O_3 are used in O_4 . All values indicate FVAs.	78
5.14	Two mixing trees using the same input fluids to produce the same output fluids, but creating different LOFs due to different mixing order.	79
5.15	Mixing trees for target ratio 1:1 which can be precisely mixed and no approximation is necessary	79
5.16	Mixing trees for target ratio 1:2 which is approximated using a maximum of 4 operations to ratio 5:11	80
5.17	Mixing trees for target ratio 1:4 which is approximated using a maximum of 4 operations to ratio 3:13	81
5.18	Mixing trees for target ratio 1:8 which is approximated using a maximum of 4 operations to ratio 1:7	82

Chapter 6

6.1	The proposed component is added to the component library, allowing other design steps to make use of it	89
6.2	Visualization of Darcy's Law parameters applied to a PDMS membrane.	90
6.3	Metering channel containing an alignment vent. Fluid is pushed towards the closed valve (e.g. by an off-chip pressure source) and the trapped air is vented out of the chip through the PDMS membrane.	91
6.4	Metering using an Alignment Vent	92
6.5	Volume error detection enabled metering component. In addition to the alignment vent, a volume error detection vent is added at the opposite end of the metering channel. If too little fluid is available to fill the metering channel completely, air escapes though the VEDV which is detected by a pressure sensor. 93	
6.6	Pressure in the flow channel forces a grover valve open, with out any applied vacuum pressure. Air vented through the membrane at the valve can be detected by a pressure sensor.	94
6.7	Prototype of a metering chamber with a large vent (19 mm vent length, 22 mm chamber length) for fast venting. Fluid is introduced through In_1 or In_2 by opening V_1 . Alignment of the fluid in the chamber can be started by opening V_2 and applying vacuum pressure to the venting pin.	95

6.8	Vacuum pressure at the vent causing the PDMS membrane to deform into the control channel, onto the control layer, effectively reducing the surface area. .	96
6.9	Control-layer part of a groove vent. Small parts of PMMA are left in the vent, keeping the PDMS membrane from significantly deforming into the control channel when vacuum pressure is applied	96
6.10	Grooves are left in the vent, keeping the PDMS membrane from deforming with minimal impact of the surface area.	97
6.11	Metering using an Alignment Vent	100
6.12	Mixer with additional vent	101

Chapter A

A.1	GUI of the BDT	108
A.2	Biochip design process overview with yellow borders indicating the steps covered by the BDT	109
A.3	Low level design view	110
A.4	Component library used by the BDT	111
A.5	Abstract view of the architecture design showing a mixer and heater but hiding component details and control connections	112
A.6	BDT design for a vent testing architecture	113
A.7	Code snippet of the control-layer G-Code used in the example	114
A.8	Minitex Machinery Mini-Mill/3	115
A.9	Hydraulic bonding press with heated plates	116
A.10	Aligning the mill's Z-Axis to the PMMA using a sacrificial piece to determine the misalignment	116

List of Tables

2.1	All available FPSs for the architecture in Fig. 2.11	21
3.1	Examples of faults occurring in FBMBs	34
4.1	Partial actuation Table for IVD. 0: Open. 1: Closed. X: Don't Care	51
4.2	Valve Group Configuration	56
4.3	Valve Group combination examples: Examples 1-5 show the outcome of combining certain VGs regarding the architecture and routes from Fig. 4.8. Examples 6 and 7 show how different constraints can have varying effects on the schedule length. For these examples, 10, 8 and 6 time units are assumed for routes 1, 2 and 3 respectively.	59
4.4	Comparison between PCM and CSO. The percentage values in PCM indicate the reduction of the pin count and the increase in completion time compared to CSO respectively.	61
5.1	Comparison of fluid consumption, number of operations and execution time for MM, REMIA and NFB4 (NFB pruned to 4 vertices), at precision levels 8, 9 and 10. All results represent the average outcome of 10 test cases, 8 synthetic and 2 real world applications	82
5.2	Experimental results for optimization of a Colorimetric Cholesterol Assay (CCA) and a Proteasome Activity Assay (PAA).	84
6.1	Experimental results for vents using various vent designs and sizes. Vent areas marked (S) use basic, normal depth control channels at the vent, (D) indicated deeper (1 mm) control channels to allow increased membrane deflection and (G) marks vents using a groove design. * Marks the average flow while covering the vent from no cover to full cover. @ Marks tests filling a complete chamber.	98

Chapter 1

Introduction

This chapter contains contributions made in P1, P2, P3, P4 and P5, as stated in Section 1.4

Microfluidics refers to the control, manipulation and behaviour of small volumes of fluids, typically confined to space of sub-millimetre scale. After first attempts of dispensing nano- and pico-litre volumes in the 1950s, microfluidics made its first major breakthrough as the basis for the ink-jet technology still used today. This highly interdisciplinary field of physics, engineering, biochemistry and bio- and nano-technology has sparked increasing academic interest for decades. Starting with the first Lab-on-a-Chip device in 1979, microfluidic biochip research began serious growth in the 1990s with the development of micropumps, flowsensors and integrated fluid treatment analysis systems [66]. Within recent years this growth continued in both academia and industry with a growing publication count as shown in Fig. 1.1 and over 3000 patents referring to microfluidics and the USA and Europe, being issued in 2017 [91] [92].

1.1 Microfluidic Biochips

Microfluidic biochips, also called Lab-on-a-Chip (LoC) devices, integrate multiple biochemical analysis functionalities such as dispensers, filters, mixers and detectors on a single chip. While conventional analysers for macroscopic chemical and biological processes take up whole laboratories, biochips can integrate these at a sub-millimetre scale, and provide several

other advantages such as reduced sample volumes, ultra-sensitive detection and increased throughput [121]. Several biochemical applications have already been demonstrated on biochips such as Protein Crystallography, Amino Acid Analysis, Chemical Synthesis and High-Throughput Screening [72]. Microfluidic large scale integration provides additional advantages which can already be seen in advances in single cell, genomic and protein analysis [10]. The increasing degree of automation of biochips does not just allow for higher usability and throughput, but enables their use in harsh and even extraterrestrial environments [84].

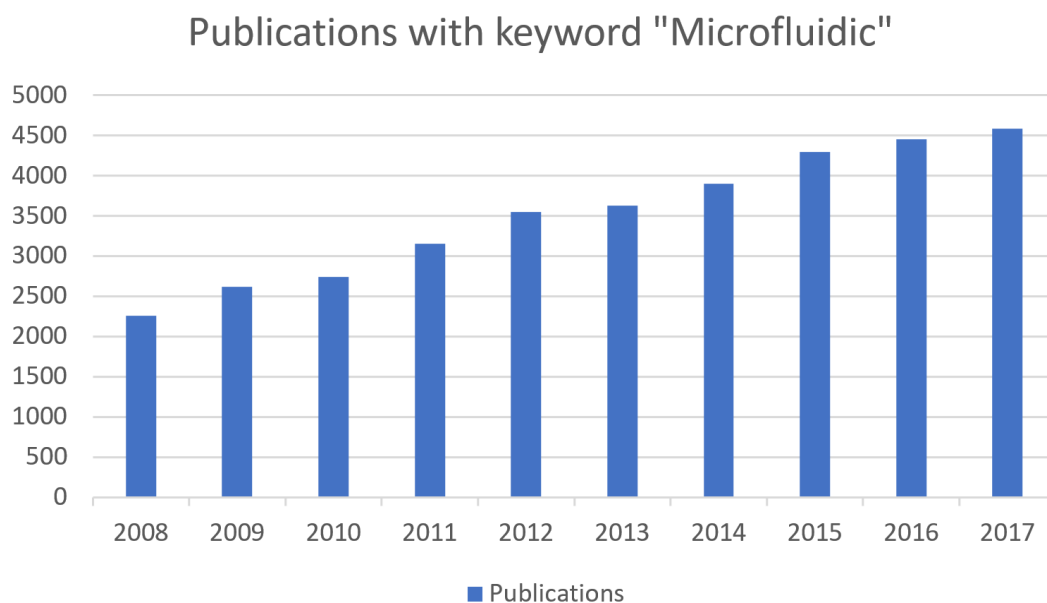


Figure 1.1: Number of publications related to microfluidics [90]

Multiple microfluidic platforms are available today which have previously been categorised into Capillary, Pressure Driven, Centrifugal, Electrokinetic and Acoustic, with further subcategorizations available [67]. Each category provides different characterizations within the used technologies and materials to achieve basic microfluidic functionality such as fluid-transport, metering, mixing or incubation.

However, based on how the fluid is manipulated on the biochip, they can be more broadly divided into two types. *Digital Microfluidic Biochips* (also called Droplet-Based Microfluidic Biochips (DBMB)) and *Continuous-Flow Biochips* (also called Flow-Based Microfluidic Biochip (FBMB)) as shown in Fig. 1.2. Digital Microfluidic Biochips manipulate liquid as discrete droplets on a two-dimensional array of electrodes [15]. A common technique to move droplets on such chip is ElectroWetting-On-Dielectric (EWOD). Droplets are dispensed on electrodes with a hydrophobic coating causing a high contact angle. By applying elec-

trical current, the droplet's surface energy increases, causing a reduced contact angle and therefore wets the surface. Using this process, multiple electrodes can be combined to form virtual components for transport, mixing or separation of droplets. The functionality of such biochips is further increased by additional components such as photodiodes for sensing.

Continuous-Flow Biochips transport a continuous flow of liquid in channels which are enclosed within the chip. This flow is manipulated using micromechanical valves. These valves are actuated using pressure sources, thereby controlling the flow of the fluid within the biochip as desired. The research conducted in this thesis is focused on such continuous-flow biochips, and the functionality, fabrication and application of which is described in detail in Chapter 2.

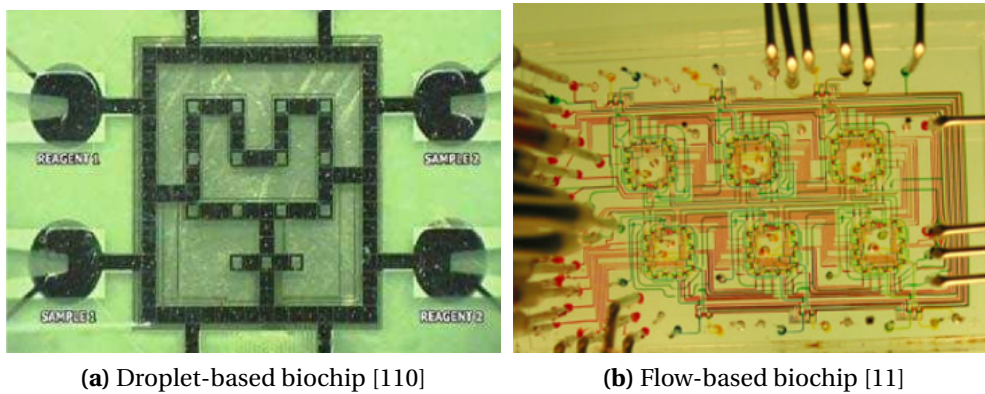


Figure 1.2: Two types of biochips

1.2 State-of-the-art and Commercialization

The ever growing interest in microfluidic biochips by the industry is evident. Previous market projections have been outperformed and new projections show a steep increase. For example, the global market for biochips was estimated to reach USD 4.6 Billion by 2017 [79] back in 2012. However, the market already surpassed this estimate with USD 7.63 Billion in 2015 and is now forecast to reach USD 17.75 Billion by 2020 [68]. An increasing number of companies are founded on the specific premise of fabricating biochips, or biochip related hardware such as Microfluidic Chipshop [17] and Elveflow [28]. Established companies in medical component and plastic engineering are also beginning to facilitate biochip production [60].

Many biochip foundries for FBMBs [17, 76] specialize on passive biochips, which receive their name from the lack of active components built into them. Instead of pressure actuated valves, mechanical, manually operated (e.g. rotary) valves [12] are used to stop or direct the fluid flow. Accurate design of channels and chambers allows to mix various liquids by diffusion using only the pressure that drives the liquids through the chip. Such chips have become the focus of the industry due to their simplicity, both in fabrication and use. Tried and true techniques such as injection molding allows for cheap fabrication of large quantities of chips. The lack of active components ensures ease of use and reliability. However, while active biochips are significantly more complicated to design, fabricate and use, all of which will be discussed in detail in Chapter 2, they also provide a level of functionality that cannot be rivalled by passive biochips. Controlling the biochip during an applications execution allows for more automation, just-in-time feedback and response options and more general purpose use. While passive biochips are application-specific due to their fixed design, an active biochip's functionality can be used to serve many purposes. While commercial availability of active biochips is still limited, companies already provide tools and products for individual fabrication of chips and their use [28]. Current and near future uses of FBMBs include CDNA-synthesis [71], DNA assembly protocols for synthetic biology [107] and the Mars Organic Analyser [30].

1.2.1 Biochip Design and Development

Initially, microfluidics and the resulting biochip research was focused on specific advantages that arise from using very small volumes of reagents. Enhancing the analytical performance for pathogen sensing for example was one of the driving reasons for miniaturization. This quickly also presented other advantages such as reduced reagent consumption and integration of sensors and other components for ease of use [64]. At this stage, the design process of the biochips was still sidelined. Prototypes are build in a way that best highlights the advances in reagent consumption or detection accuracy. However, realizing that biochips can be extended to perform automated, complex procedures that would require advanced design methodologies to keep the portability, efficiency and low cost, an increasing amount of research is put into advancing the biochip design. At the present time, the design of biochips is still done manually to a large degree, with the limited aid of tools such as CAD software like AutoCAD [101]. Some early stage tools are available which offer a certain degree of design automation [51]. While this is feasible for many current designs, it does not only

require skills and knowledge in biochip design, but also means that applications are manually mapped to these designs, which can include binding of operations to specific components and determining correct valve actuation sequences. This level of exposure is comparable with manually programming the gates of integrated circuits for microelectronics and is extremely time-consuming and error prone. Ongoing advancements in integration density and component complexity will amplify these issues to a point where manual design is no longer feasible. The microelectronics industry has faced similar problems in the past to which solutions have been found that can be adapted to microfluidic biochips. Fig. 1.3 presents such a possibility, showing how microfluidics can follow in the footsteps of microelectronics. Very Large Scale Integration (VLSI) is the cornerstone of any modern electronics design and the same concept and similar methodologies, hence called microfluidic VLSI (mVLSI), can shape a similar future for microfluidics. Providing the same high integration that is considered the standard in present electronics would make any currently available microfluidic platform obsolete. Proof of concept for such integration is already available, with a biochip using more than 25000 valves and about one million features to run 9216 polymerase chain reactions in parallel build by Fluidigm [37]. Additional work also provides examples of how multiple functions can be integrated and run in parallel on a single chip, using mVLSI concepts [71].

1.3 Motivation

“The electronic VLSI circuits have benefited heavily from design automation and the electronic designers today work as conveniently on the billion transistor multi-core processors as they did on the Intel 4004 processor with only 2,250 transistors in the early days [81].” The ability to make similar statements about mVLSI in the future would surely be a clear indication of a successful design evolution. However, at this time many individual parts of the design process still need to be rid of unrealistic assumptions and limitations before a VLSI-like equivalent can truly take over. As contributions for this interdisciplinary topic are being made from many scientific fields, it is more likely that needs are misunderstood and unrealistic assumptions are being made. Contributions from the computer engineering field are no exception, as proposed designs and optimizations are often only tested on models as no prototype fabrication facilities are available. It is therefore crucial that gaps between what is proposed and what feasible are identified and closed, ensuring that these contributions remain relevant. Design automation is another crucial but so far underdeveloped part of

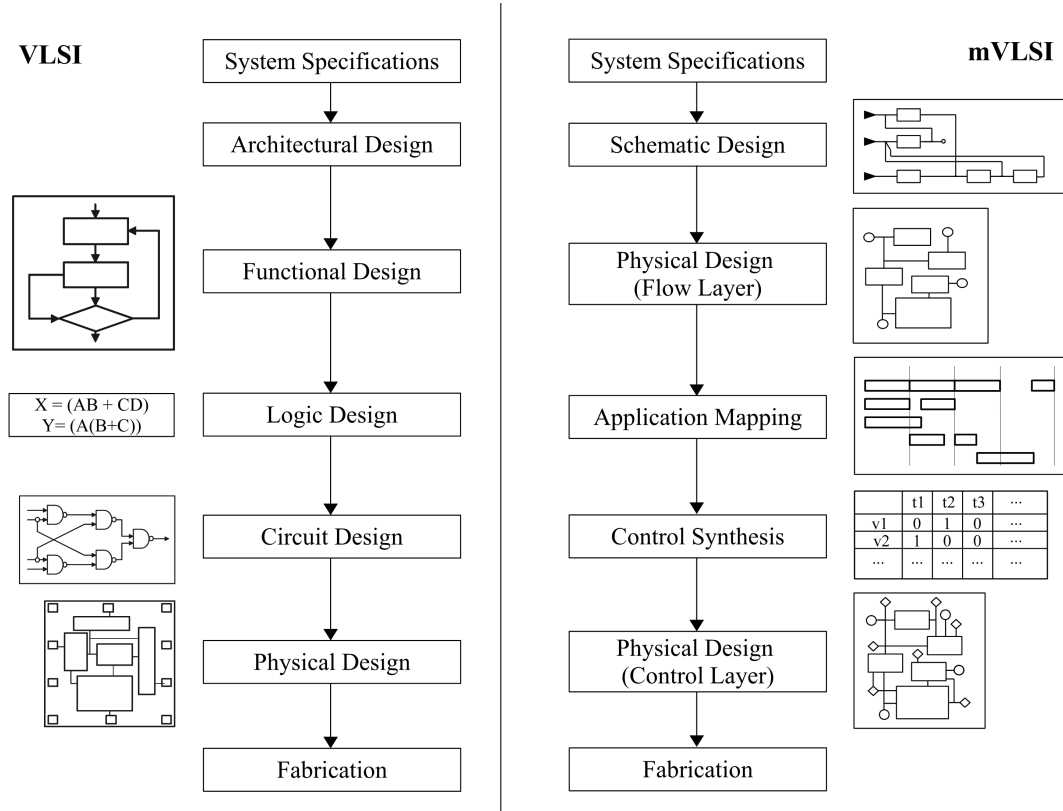


Figure 1.3: Electronic vs. Microfluidic VLSI design process [9]

this process. With custom and bottom-up methodologies requiring many manual steps remaining on the forefront, it is unrealistic to assume widespread use of biochips from end-users in fields such as biology or medicine which would require extensive training to take on the design process themselves. User-friendly tools incorporating algorithms that can solve design issues for the large variety of possible applications are therefore the logical next step to improve accessibility and growth of the technology. This also provides a platform from which further development such as towards mVLSI design has an easier access point.

The contributions presented in this thesis were made to specifically overcome such issues and limitations. The presented design-automation and -processes make integration of complex features feasible and allow for more biochip designs to evolve from mere models to being fully fabricatable and usable. With mVLSI looming on the horizon, the presented methods also address issues that arise from large scale integration and can ease the transition to such designs.

1.4 Thesis Overview and Contributions

The following publications represent a major part of my contributions made during my PhD course and therefore to this thesis. The published material is re-used in part or fully in this thesis, including results, primary material, data and interpretations. Chapters and sections using material from some or all of these publications indicate this, by referring to these publications as PN, with N being replaced by the number of the publication.

P1: A. Schneider, P. Pop and J. Madsen. A pin-count reduction algorithm for flow-based microfluidic biochips. In Proc. of Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP), 2016.

P2: Alexander Schneider, Paul Pop, and Jan Madsen. Pin-count reduction for continuous flow microfluidic biochips. volume 24, pages 483–494, Berlin, Heidelberg, January 2018. Springer-Verlag.

P3: A. Schneider, P. Pop, and J. Madsen. Waste-aware fluid volume assignment for flowbased microfluidic biochips. In 2017 Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP), May 2017.

P4: A. Schneider, P. Pop and J. Madsen, Volume management for fault-tolerant continuous-flow microfluidics, 2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Cambridge, 2017.

P5: A. Schneider, P. Pop, and J. Madsen. A Novel Metering Component for Volume Management in Flow-Based Microfluidic Biochips. In 2018 Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP), May 2018.

The rest of the thesis is structured as follows:

Chapter 2 provides a detailed introduction to biochips and their underlying technology. We discuss models that represent biochips and their functionality without restriction to a specific (e.g. valve-) technology. Furthermore, we provide an introduction to the additional hardware that is required to operate a biochip and the accompanying design constraints. This infor-

mation provides the background knowledge on which the contributions in Chapters 4-6 are based on. This chapter is based on the publications P1, P2, P3 and P5.

Chapter 3 introduces current design methodologies including fabrication, as well as the possibilities for dynamic interaction with biochips by making them programmable. Due to the complexity of the biochip design process, researchers have proposed solutions to isolated design tasks. Frequently, this leads to sub-optimal or even infeasible solutions as they cannot be integrated into the design process properly. In this chapter we introduce existing methodologies which in their core are essential to every biochip and along which we therefore implement our proposed methods and tools. This chapter is based on the publications P1, P2, P3, P4 and P5.

Chapter 4 covers my contribution on the topic of pin-count reduction, which allows to operate biochips using a reduced amount of controls. Even though thousands of valves can be integrated on a biochip, their functionality is limited due to the number of control-pins that can be connected to an external controller. Hence we have proposed a pin-count reduction technique which is integrated into the physical design and control synthesis tasks, to reduce the number of required external controls using shared valve actuations. Contrary to previous solutions, we consider the 2-way impact of binding, routing and scheduling with pin-count reduction. This uncovers superior solutions and furthermore enables to trade-off an increased application execution time for further pin-count reduction. This chapter is based on the publications P1 and P2.

Chapter 5 covers my contribution on waste-aware volume management, providing accurate control over the fluid volumes used in biochips. Related work has focused on optimizing mixing operations, aiming to reduce the number of required mixing steps and therefore the assay runtime. The potential increase in fluid consumption and the impact on volume management is often ignored. However, dispensing additional fluid (overflow) can be expensive, while shortages of fluid (underflow) can interrupt the execution and even require manual replenishment. Therefore we provide a volume management approach that satisfies all constraints while extending previous mixing optimizations. By allowing leftover fluids from operations to be reused in other mixing operations, the total volume of required fluids can be

further reduced. Due to the techniques low computational complexity, it is also feasible to be used during error recovery at runtime. This chapter is based on publication P3.

Chapter 6 introduces my contribution on a novel metering component for volume management. Moving fluids within a biochip is simply referred to as fluid transport by researchers, ignoring the underlying complexity. However, realistic biochip design has to offer the capability to provide specific volumes of fluid to precise locations in the architecture, which is not a trivial task. Our proposed component is capable of aligning and metering fluids, as well as detecting missing fluids during the biochip's operation. We make use of the gas permeability of the PDMS from which parts of biochips are made to introduce vents that allow for the proposed functionality. This chapter is based on the publication P5.

Chapter 7 contains the discussion and concludes this thesis.

Appendix A describes a biochip design tool that has been developed to design and fabricate our prototypes. Current tools for physical design do not bridge the fabrication gap, which extends into lacking functional verification. Instead, interdisciplinary specialist knowledge is required to verify, potentially adapt and then turn the synthesized design into working biochips. We have connected our physical-design methodology to a micromilling fabrication process for biochips using normally-closed valves. Such biochips are build from PDMS and PMMA that can be purchased off-the-shelf and are therefore easy to fabricate by non-expert end-users. The micromilling process to fabricate the PMMA layers is automated using the G-Code output from our tool.

Chapter 2

Biochips

This chapter contains contributions made in P1, P2, P3 and P5, as stated in Section 1.4.

Decades of research on biochips has led to a wide range of design methodologies, fabrication techniques, modelling and simulation tools, automation and optimization etc. In this chapter we will introduce those technologies and methodologies which provide the base for our work presented in the following chapters. The work presented in this thesis is based on Flow-Based Microfluidic Biochips (FBMB). While some analogies can be drawn between FBMBs and Droplet-Based Microfluidic Biochips (DBMB), adapting methodologies from one platform to the other still requires considerable work. Therefore, possible applications of this work for DBMBs or a direct comparison to existing DBMBs research is considered out of scope for this thesis. However, within the FBMB platform, no restrictions are made concerning the type of technology used.

2.1 Passive Biochips

We consider a biochip to be passive if it contains no active components such as valves or pumps which can direct the flow of liquids on the chip. Such chips are therefore limited in their functionality, but also simple regarding the design and fabrication. Fig. 2.1a illustrates a mixing channel example where two input channels are joint, causing the fluid from both to mix. However, the outcome quality of operations such as mixing depends heavily on the

component design and the physical properties of the used fluids [57]. This example uses simple diffusion to mix the input fluids, which is sufficient for some, but not all liquids, which marks the limits of design simplicity for passive biochips. More effective methods for this purpose have been developed, such as split-and-recombine, lamination, recirculation (recycle flow) [33] and chaotic advection [83]. Such methods result in more complex channel designs to provide adequate diffusion such as an example shows in Fig. 2.1b.

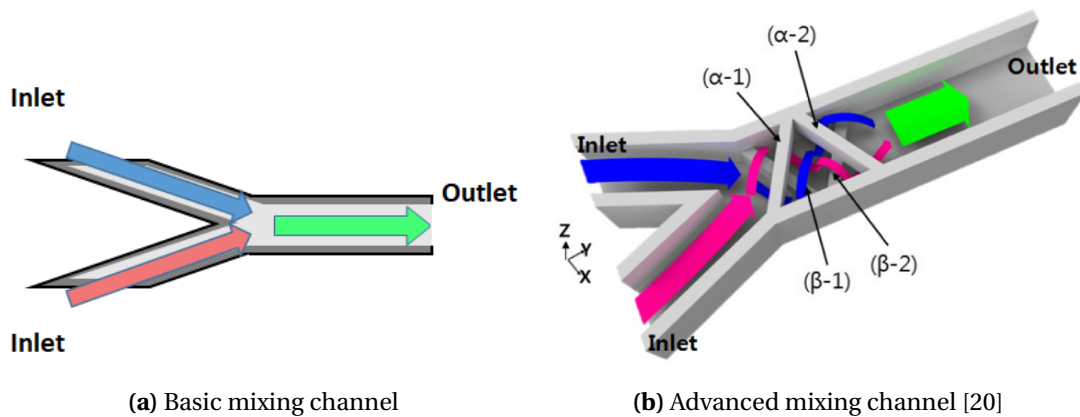


Figure 2.1: Passive mixing architectures

Passive biochips are currently the most common commercially available chips [17]. One of the main reasons for this is the easy usability, as no additional hardware is necessary to operate the chip. Reagents can be inserted into the chip directly, for example using a syringe. Furthermore can these chips be fabricated as one solid piece using injection-molding which is very efficient at high quantity production. Additional designs such as orientation-based biochips [88] have been proposed, indicating the wide range of design possibilities. As shown in Fig. 2.2, mixing of two fluids can be achieved in a simple channel. The desired mixing functionality is achieved by adjusting the orientation of the chip in regard to earth's gravity, causing fluids of different density to mix in certain ratios.

2.2 Valve-based Biochips

While passive biochips are popular due to their simplicity in fabrication, they also come with drawbacks. Passive biochips are for the most part application specific, meaning that most biochemical assays need an individual biochip design. Programmable biochips on

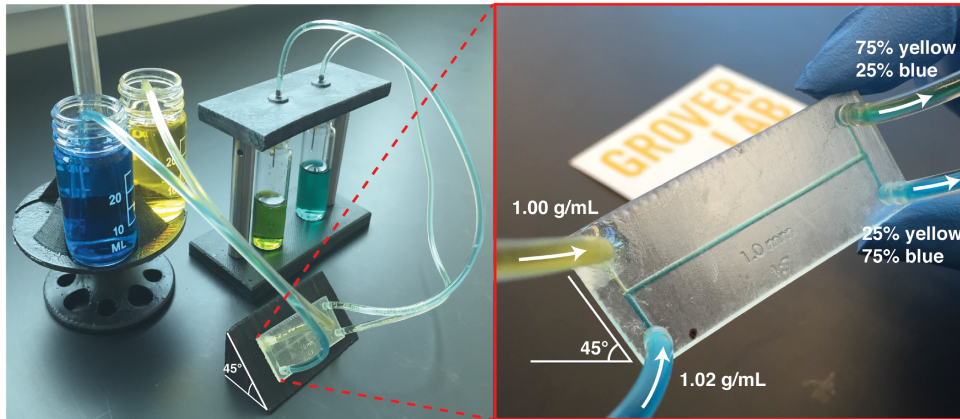


Figure 2.2: Orientation-based biochip [88]

the other hand can reconfigure the timing, order and routing of operations to allow single designs to operate several different applications. Furthermore do programmable chips provide more flexibility and increased automation. Feedback loops can be constructed which allow outcome based experiments to continue, without manual intervention. Additionally, programmable biochips provide the possibility of error recovery, which can save high value reagents or data from getting lost due to faults [117].

Active biochips are also called valve-based biochips as they obtain all their active functionality from basic valves. Valves in biochips are considered to only have two states: (1) Open and (2) Closed, which can either restrict or allow the flow of fluids within a channel in a biochip. This forms a strong analogy to transistors in electronics which allows for many methodologies to be adapted to microfluidics. Currently, two major valve technologies are being used, Quake valves and Grover valves which as introduced in the next two sections.

2.2.1 Quake Valves

The microfluidic valve technology developed at Stanford (also known as Stanford valve) [32] is designed for biochips, built from PolyDiMethylSiloxane (PDMS) which is a flexible and durable material (more details on fabrication in Sect. 3.4). To introduce valve functionality, the chip is divided into two layers, the Flow- and Control-Layer. The Flow-Layer contains channels used to transport fluids, while the channels in the Control-Layer distribute pressure. Fig. 2.3 shows a schematic design of a Stanford valve and indicates the separation of the chip in two layers. To close this valve, pressure is applied to the channel in the Control-Layer

(control channel) which pinches the channel in the Flow-Layer (flow channel), interrupting it and prohibiting fluid from passing. Such valves have been shown to have a very fast response time up to 100 Hz [32]. Furthermore can they be densely integrated as functional valves at $6 \times 6 \mu\text{m}$ have been fabricated, although at a slower response time [8]. In its idle state, i.e. no pressure is applied to the control channel, this valve remains open, hence it is also called a “normally-open valve”. To allow better sealing of the flow channel, the control channels are build rectangularly, while the flow channels are rounded on one side. Furthermore, the control channel can be placed above or below the flow channel [32].

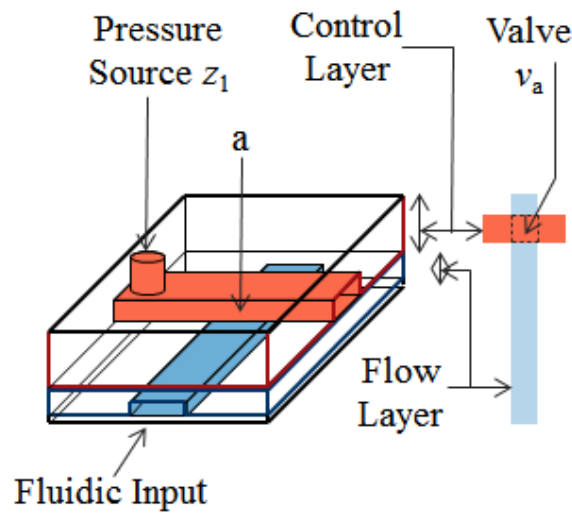


Figure 2.3: Schematic design of a Stanford valve [32]

2.2.2 Grover Valves

An alternative valve technology developed at Berkeley (also known as Berkeley valve) [42] is designed for biochips, build from three separate parts. Contrary to Quake valves the two outer layers are made from PMMA, Glass or a similar material and are separated by only a thin (typically 100-250 μm thick) membrane of PDMS. The other layers however are also referred to as Flow- and Control-Layer, into which the channels are fabricated. The thin layer of PDMS is required to enable this valve technology. As shown in Fig. 2.4, a valve is fully integrated into the layers, by leaving a notch in the flow channel during fabrication, which interrupts the flow of fluid through this channel. Simultaneously, a deflection chamber is built into the Control-Layer at the same location. Such cambers typically occupy at least 1×1

mm^2 . In its idle state, i.e. only atmospheric pressure is applied to the control channel, this valve remains closed, hence it is also called a “normally-closed valve”. In Fig. 2.4b the valve is shown in its open state, caused by vacuum pressure being applied to the deflection chamber in the Control-Layer. This pressure deflects the flexible PDMS membrane into the deflection chamber, creating a connection between the previously separated parts of the flow channel.

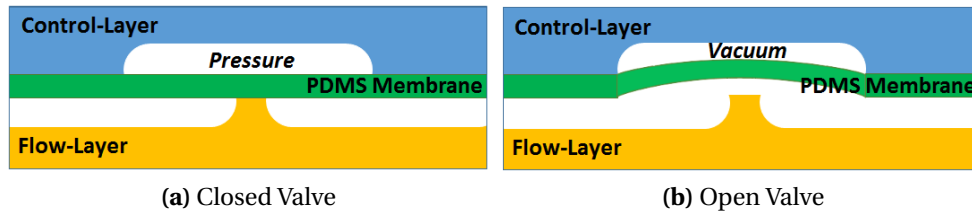


Figure 2.4: Schematic design of a Grover valve

2.3 Fluidic Components

Combining multiple valves allows for the fabrication of more complex components such as active mixers, sealed heating and detection chambers, switches or storage components. We model such components schematically as shown in the upcoming figures, which indicate the flow channels in yellow and valve positions in blue. Control channels that connect the valves with control-pins are omitted.

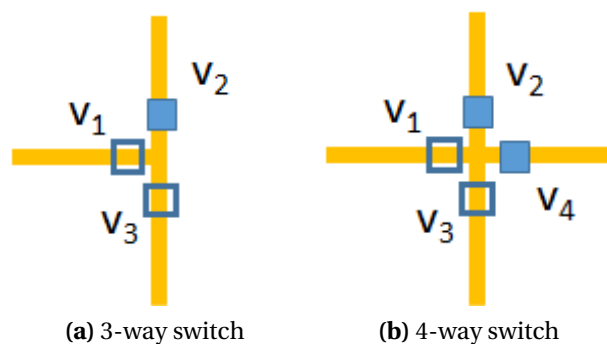


Figure 2.5: Switches

Switches enable the creation of complex architecture designs, by allowing overlapping channels and guiding the flow of the liquid by restricting access to certain channels. A switch is typically an intersection of three or four channels, where each channel can be blocked

by a valve, located close to the intersection as shown in Fig. 2.5. However, there is no hard restriction on the number of intersecting channels to form a switch, or the angle in which channels connect to the switch. While unlikely to affect the majority of applications, the switch design can affect properties such as the fluid's Reynolds Number [103] or heat transfer [65]. However, depending on the valve type and fabrication method, the number of valves that can be placed closely and at the desired angle to a switch is far more likely to be the limiting factor, resulting in the common use of 4-way and 90° angle switches.

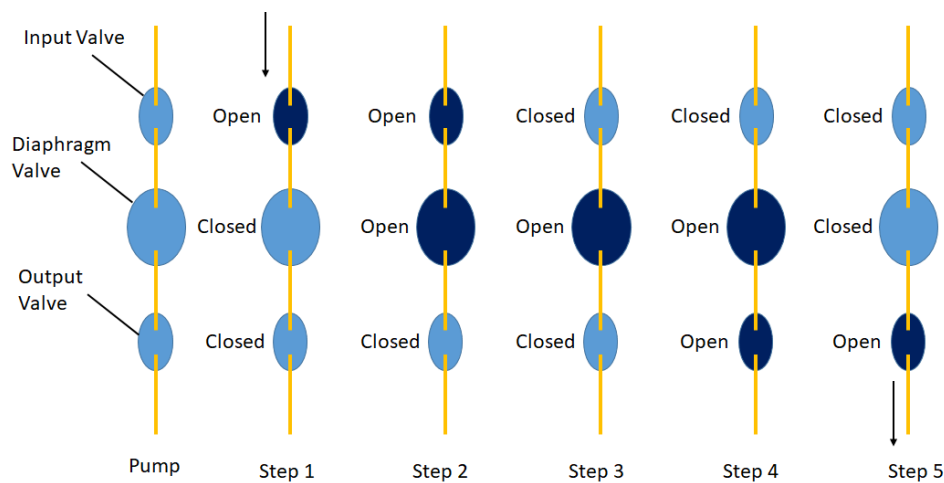


Figure 2.6: 5 stage pump actuation using 3 Grover valves. Yellow lines are flow channels. Light blue circles are displacement chambers. Dark blue circles are displacement chambers filled with fluid.

Pumps allow fluid to be moved within the chip without any outside pressure source. By placing three valves in sequence in a channel, these valves can be actuated to drive the liquid in this channel forward. Pumping fundamentally works the same using either Quake (normally-open) and Grover (normally-closed) valves. The speed by which the liquid is displaced can be controlled by the actuation speed of the pumping valves. The volume that is displaced with each actuation of the pump depends on the channel and valve sizes. Quake valves can be built at a very small scale and have been shown to allow fluid displacement of volumes as small as 100 pl [32]. The displace volume using Grover valves depends on the size of the displacement chamber of the diaphragm valve, which has been shown to have volumes down to the nanolitre range [41]. To create a one way motion of the fluid, the valves have to be actuated in a specific sequence as illustrated in Fig. 2.6 on Grover valves. The opening of a valve, i.e. deflecting the membrane into the displacement chamber, creates a vacuum at the valve. To equalize the pressure differential, contents of the flow channel (e.g. fluid) are sucked into the displacement chamber. Closed adjacent valves control from which side the

pressure differential is equalized. For example, in step 1 in Fig. 2.6 the vacuum created by opening the input valve can only be equalized from the top, as the closed diaphragm valve does not allow any flow towards the input valve from the bottom.

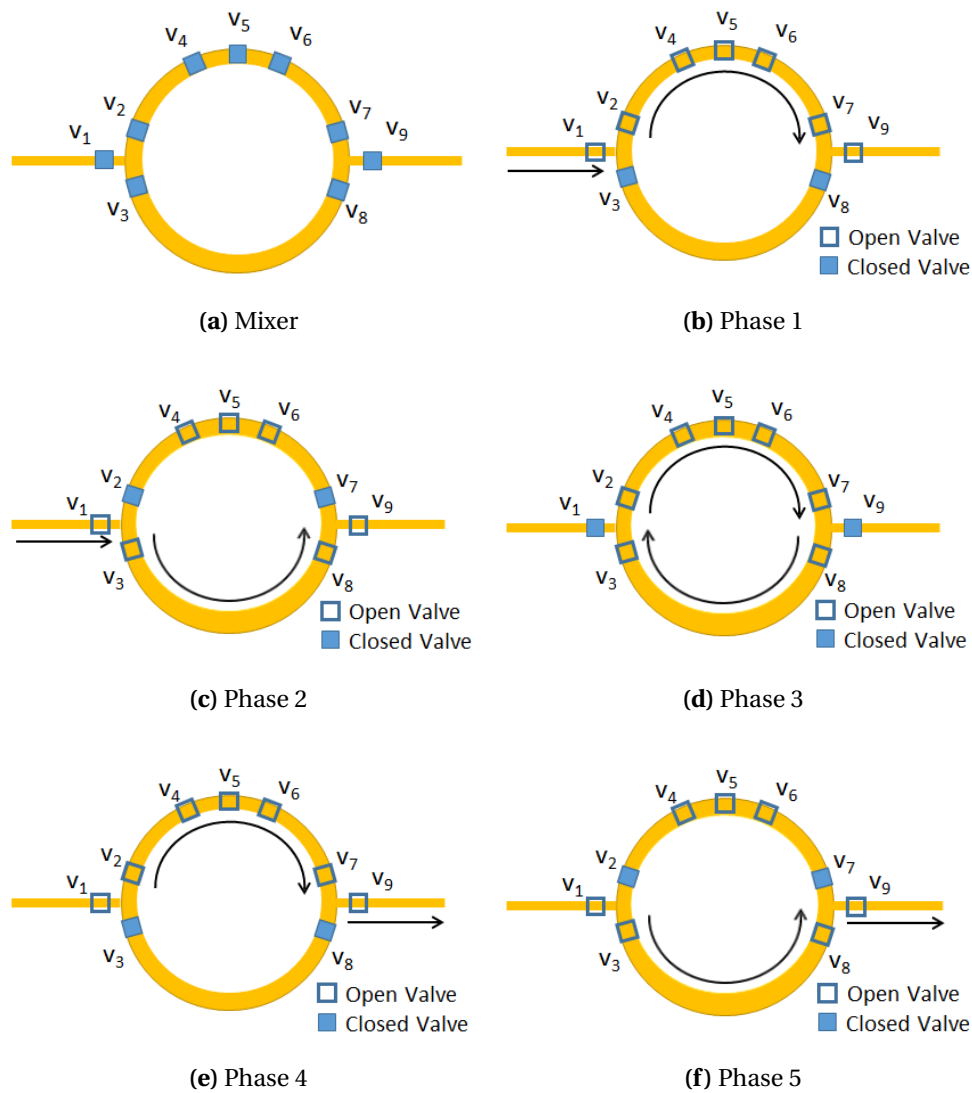


Figure 2.7: 5-Phase equal ratio mixer

Mixers provide the same functionality as centrifuges or vortex mixers in a conventional lab. While passive chips require fluids to mix by diffusion within various designs of diffusion chambers, active mixers can achieve this in an enclosed area, with no restriction on mixing time [18]. A model of such a mixer is shown in Fig. 2.7, which consists of a round mixing channel and nine valves. The input and output channel, along with the valves 1-3 and 7-9 are placed so that the mixer is separated into two equal sized chambers. This allows to fill

the mixers with two equal volumes of fluids, resulting in an equal mixing ratio. Such mixers are therefore also called 1:1 mixers. Valves 4-6 form a pneumatic pump which is used to operate the mixer and mix the loaded liquids. The mixing operation consists of five phases as shown in Figs. 2.7b to 2.7f. The first two phases represent the loading of the upper and lower chamber of the mixer with the desired reagents. In phase three, the mixer is sealed and the pump actuated, causing motion within the rounded channel. The last two phases remove the mixed liquid from the upper and lower chamber. Other ratios can be achieved by performing multiple mixing operations in sequence or using arbitrary ratio mixers, which is covered in detail in Chapter 5.



Figure 2.8: Schematic design for any component requiring a reaction chamber such as heaters and detectors

Components such as **Heaters** or **Detectors** are represented in a simple manner by an enclosed part of a channel as shown in Fig. 2.8. From a schematic view, these components only need to be able to hold fluids in place while the operation is performed. This can be in form of outside components such as a microscope, or combined with internal components such as magnetic beads [115]. The shape of such chambers is not limited to rectangles, but can be fabricated in other shapes to aid the desired process. Common practices for incubation are heating plates or infra-red light sources which allow for very quick changes in the fluids temperature due to the small volume [39]. On-chip detection methods have seen limited use so far as many conventional techniques such as agarose gels are difficult to implement within a biochip and optical detection is hindered by the chip's outer layers, but still possible [109]. However, advancements in the miniaturization of electrochemical sensors will greatly benefit the range of detection methods for biochips [13, 35] and bioanalysers using microfluidics provide a range of detection options as well [89]. Nevertheless, manual and external detection measures are still common to date, such as analysing the fluid on-chip using a microscope, or moving the fluid out of the chip into separate detection hardware.

Input and **Output** interfaces are typically holes in the flow-layer, connecting a flow channel to the outside of the chip. These interfaces can then be connected to fluid sources or waste containers using tubes as shown in Fig. 2.9. Such interfaces are identical and can therefore be assigned as required. Alternatively, an input can form a reservoir, allowing the fluid to be placed on the chip, removing the need for a fluid source. The fluid can then be moved

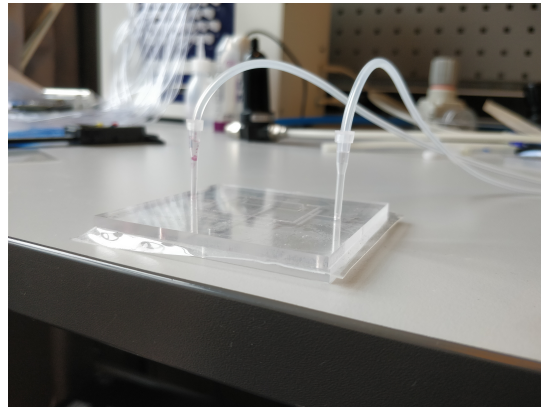


Figure 2.9: In- and outlets connected with flexible tubes

from the reservoir using on-chip pumps. For ease of use, such reservoirs can also be filled (e.g. with buffer) and sealed during fabrication, reducing the amount of fluids that need to be introduced to the chip by the user.



Figure 2.10: Storage component with four storage channels. The number of channels can vary to suits the architectures needs

Storages are required to store reagents for later use, without occupying functional components. A storage component consists of multiple channels that can hold a specific volume of liquid, for example as much as one mixing chamber. As Fig. 2.10 shows, a series of valves allows to access each storage cell individually.

2.4 Biochip Architecture Model

We use a system-level architecture model based on a topology graph to capture the biochip architecture [97, Chapter 3]. In such models, all components that provide functionality to execute a fluidic operation, such as mixers, heaters, detectors and storages are part of the same group of components. We call these components Functional Fluidic Units (FFU) and

they are represented as rectangles in our models as shown in Fig. 2.11. The same model shows switches as black nodes, in- and outputs as white nodes and channels as lines. This non-technology specific architecture model determines the number and type of components in the architecture and how they are connected to each other, but does not determine the size, position or mechanical properties of the components or how they are connected to pressure sources. This allows a wide range of technologies (e.g. valve or mixing) to be implemented using the same model. While our example architecture in Fig. 2.11 has preassigned in- and outputs, it is important to note that for most architectures the physical properties of such interfaces do not differ and the assignment is therefore interchangeable. The assignment of interfaces as in- or outputs would then be made in the routing phase (see Sect. 3.1). However, for simplicity reasons we choose to have our architecture's interfaces preassigned.

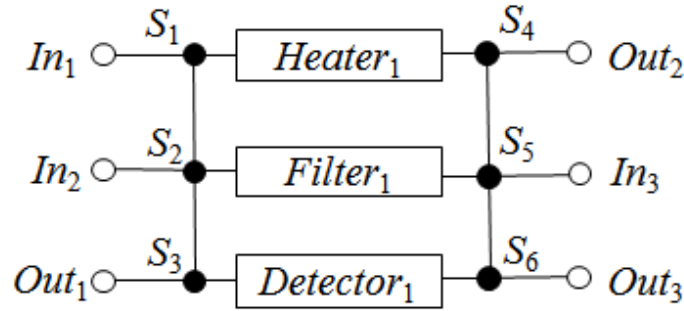


Figure 2.11: Architecture model with three inputs and outputs, one heater, filter and detector and six switches to interconnect the channels leading to the components

Furthermore, this model allows to determine all available routes or *Flow Paths (FP)* on the chip. Since fluids are moved by pressure, a complete FP must follow one of two rules. (1) The start and end of the FP are separate in- or outputs, one through which the pressure is applied and one through which the pressure can be released from the chip. (2) Alternatively, pressure can be generated by on-chip pumps, making circular FPs without connection to any in- or outputs possible. FPs can be broken down into *Flow Path Segments (FPS)*, each of which represents a directed connection between two FFUs. To connect the two FFUs, a FPS can contain any number of switches, but no additional FFUs, e.g. (In_1 , S_1 , S_2 , $Mixer_2$) is a valid FPS while (In_1 , S_1 , $Mixer_1$, S_3 , S_4 , $Mixer_2$) is not. FPSs are mutually exclusive, meaning they cannot be used to transport fluids at the same time, if they share at least one vertex. Table 2.1 shows a list of all available FPSs for the architecture in Fig. 2.11. As FPSs represent every available connection between FFUs, a FP can be described as a set of FPSs, which follow one of the complete FP rules set forth before.

FPS ₁ = (In ₁ , S ₁ , Heater ₁)
FPS ₂ = (In ₁ , S ₁ , S ₂ , Filter ₁)
FPS ₃ = (In ₁ , S ₁ , S ₂ , S ₃ , Detector ₁)
FPS ₄ = (In ₂ , S ₂ , S ₁ , Heater ₁)
FPS ₅ = (In ₂ , S ₂ , Filter ₁)
FPS ₆ = (In ₂ , S ₂ , S ₃ , Detector ₁)
FPS ₇ = (In ₃ , S ₅ , S ₄ , Heater ₁)
FPS ₈ = (In ₃ , S ₅ , Filter ₁)
FPS ₉ = (In ₃ , S ₅ , S ₆ , Detector ₁)
FPS ₁₀ = (Heater ₁ , S ₁ , S ₂ , S ₃ , Out ₁)
FPS ₁₁ = (Heater ₁ , S ₁ , S ₂ , Filter ₁)
FPS ₁₂ = (Heater ₁ , S ₁ , S ₂ , S ₃ , Detector ₁)
FPS ₁₃ = (Heater ₁ , S ₄ , Out ₂)
FPS ₁₄ = (Heater ₁ , S ₄ , S ₅ , Filter ₁)
FPS ₁₅ = (Heater ₁ , S ₄ , S ₅ , S ₆ , Out ₃)
FPS ₁₆ = (Heater ₁ , S ₄ , S ₅ , S ₆ , Detector ₁)
FPS ₁₇ = (Detector ₁ , S ₃ , Out ₁)
FPS ₁₈ = (Detector ₁ , S ₃ , S ₂ , Filter ₁)
FPS ₁₉ = (Detector ₁ , S ₃ , S ₂ , S ₁ , Heater ₁)
FPS ₂₀ = (Detector ₁ , S ₆ , Out ₃)
FPS ₂₁ = (Detector ₁ , S ₆ , S ₅ , S ₄ , Out ₂)
FPS ₂₂ = (Filter ₁ , S ₂ , S ₃ , Out ₁)
FPS ₂₃ = (Filter ₁ , S ₂ , S ₃ , Detector ₁)
FPS ₂₄ = (Filter ₁ , S ₂ , S ₁ , Heater ₁)
FPS ₂₅ = (Filter ₁ , S ₅ , S ₄ , Out ₂)
FPS ₂₆ = (Filter ₁ , S ₅ , S ₆ , Out ₃)
FPS ₂₇ = (Filter ₁ , S ₅ , S ₆ , Detector ₁)

Table 2.1: All available FPSs for the architecture in Fig. 2.11

2.5 Biochip Application Model

To model a biochemical application, we use a directed, acyclic and polar sequencing graph model [80]. An example of such an application graph can be seen in Fig. 2.12. A node in this graph is an operation that runs on a FFU. The edges denote dependencies between operations that require fluid transport. Each operation O_i has an execution time C_i when running on a FFU j . Some execution times are given by the application, such as an incubation time along with a temperature. Others are dependent on the technology used in the architecture, e.g. the mixing time might vary depending on the used mixing technology. Finally, some operations do not have a precise execution time, such as optical detection which is done

manually. Application models are not architecture specific. Therefore, before an application can be run on a biochip, the application has to be mapped to the chip. This includes binding operations to the functional units (where the operation should be executed), the routing of fluids (which channels should be used for transport) and scheduling (at which times the fluids are transported and the operations are executed) which are discussed in Sect. 3.1.

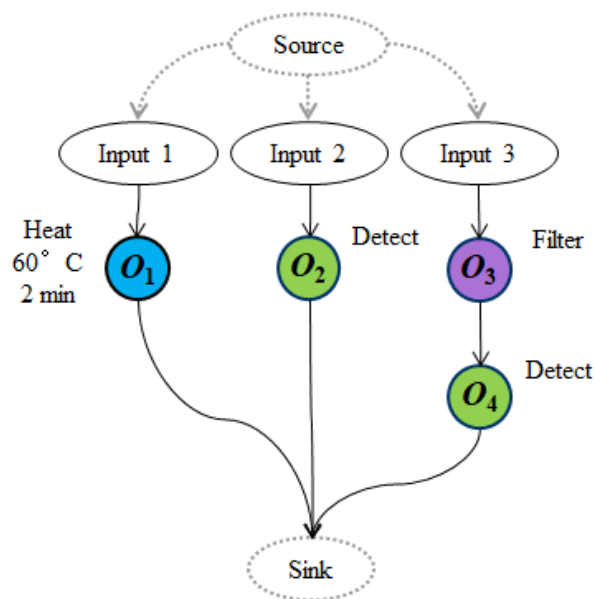


Figure 2.12: Application model with two fluid sources, one mixing and two incubate and detect operations

2.6 External Hardware

To actuate valves in a biochip, a source of pressure, either higher or lower than atmospheric pressure (depending on the valve type) is required. Furthermore, this pressure needs to be distributed to multiple valves and turned on and off independently throughout the use of the biochip. Control channels are therefore connected to the outside of the chip through an interface called a control-pin, to which a pressure source can be connected, for example through a rubber tube as shown in Fig. 2.9. More complex interfaces for larger numbers of control-pins can for example also be connected using a connection adapter as shown in Fig. 2.13. In this example, a biochip is placed between two parts of the adapter which are then held together using screws. The elasticity of a full-PDMS chip as used here aids in sealing the connection between the chip and the adapter, but adapters with integrated sealing can be

manufactured as well. The adapter redirects the pressure from the tubes to a specific area on the chip's surface, therefore determining where on the chip the control-pins have to be placed. While the predetermined control-pin locations put some constraints on the chip design, connections to the control-pins can usually be found if they are distributed evenly on the adapter. Having the chip fabricated according to the adapter design allows for rapid interchanging of biochips.

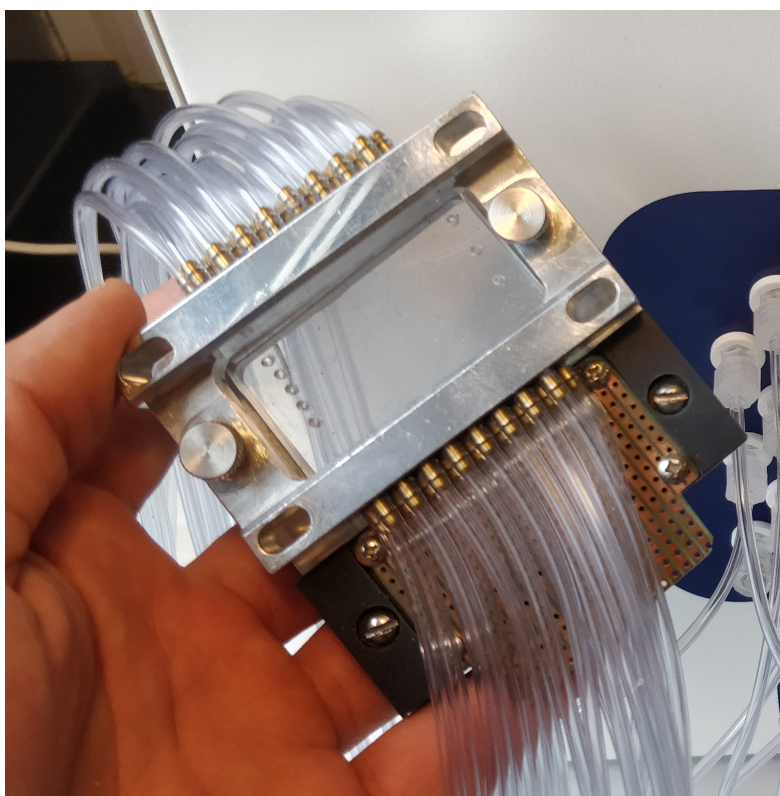


Figure 2.13: Biochip to controller adapter. Tubes from the controller are connected to the adapter, into which a biochip can be placed (System from Microfluidic Innovations Inc, IN, USA)

These pressure sources leading to an adapter or directly into the chip are fundamental components of programmable microfluidic biochips, as they control the valves that enable the integrated functionality. Therefore, pressure has to be supplied to all valves and be individually controllable to provide dynamic valve control. Furthermore, depending on the chip technology, external pressure sources may have to provide both high and low (vacuum) pressure at every connection. This is accomplished by control hardware that distributes a single pressure source into many control-pin connections and allows individual, programmable control over each connection using solenoid valves. The market for such hardware is constantly increasing and various solutions are already available to date [28, 74, 75]. While the

size, cost, weight and pressure ranges can vary significantly between different control hardware, each is considerably larger and heavier than any biochip, as clearly visible in Fig. 2.14. These measures increase with the complexity (number of valves) of the biochip that need to be controlled. It is therefore imperative to optimize the use of valves on biochips to reduce the size of external hardware that is required.

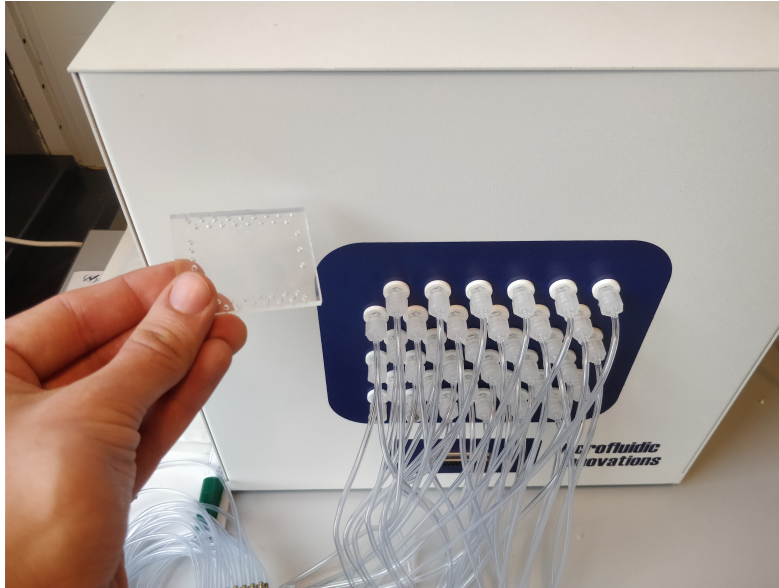


Figure 2.14: Pressure controller capable of providing high- and vacuum-pressure through 36 connectors using solenoid valves (System from Microfluidic Innovations Inc, IN, USA)

2.6.1 Control Logic

The order in which the chip's valves have to be actuated, can be extrapolated from an application's schedule [78], which in turn is created from the chip's architecture and the order of operations defined in the application. Translating the execution of the schedule into signals for valves reveals that each valve has three logical states it can reside in. Beside the two physical states, *open* and *closed*, valves can logically also be in a *don't care* state. This is the case whenever the physical state of the valve, has no impact on the execution whatsoever, as demonstrated in Fig. 2.15. Here, fluid is being passed through a grid of 4-way switches. At every switch that the fluid is routed through, 2 valves need to be open while the remaining 2 are closed to ensure a directed flow. However, the state of the valves in the unused switch is irrelevant at this point and they are considered to be in a don't care state. This logical state

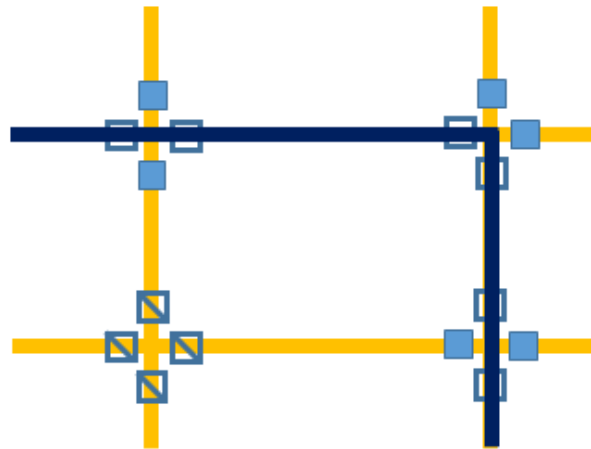


Figure 2.15: Grid of 4 switches. Dark blue lines indicate fluid flow.

plays a significant role in optimization strategies such as pin-count reduction which will be explained in detail in Chapter 4.

Such optimization strategies are necessary as typical control hardware only provides about 8-40 (depending on hardware size) individual pressure connections, requires a power source and is significantly larger and heavier than the biochip. This makes the external hardware necessary to operate the biochip a limiting factor. As the number of integrated valves can far exceed the number of reasonably available individual pressure connections, components and methods such as multiplexers [113], latching valves [43] and pin-count reduction techniques have to be implemented to allow more complex chips to be operated. Multiplexers as shown in Fig. 2.16 enable the control of multiple individual valves with each pressure source. This leads to 2^n output options being controlled using n input “bits”, or $2 * \log_2(n)$ pressure sources. Each of these bits controls two sets of valves which are in opposing states, depending on the bit’s value. A latching valve as shown in Fig. 2.17 enables more complex valve behaviour without additional pressure sources. A latching valve can be opened and closed with a short pressure impulse (< 1 second) and will retain this state for a prolonged time (> 1 minute), without the pressure source remaining active [43].

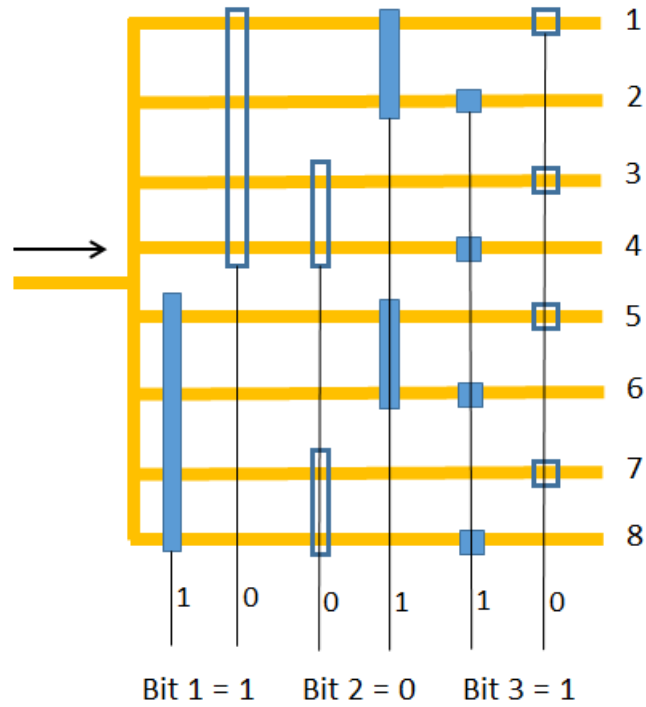


Figure 2.16: Schematic design of a 3-bit 8-way multiplexer

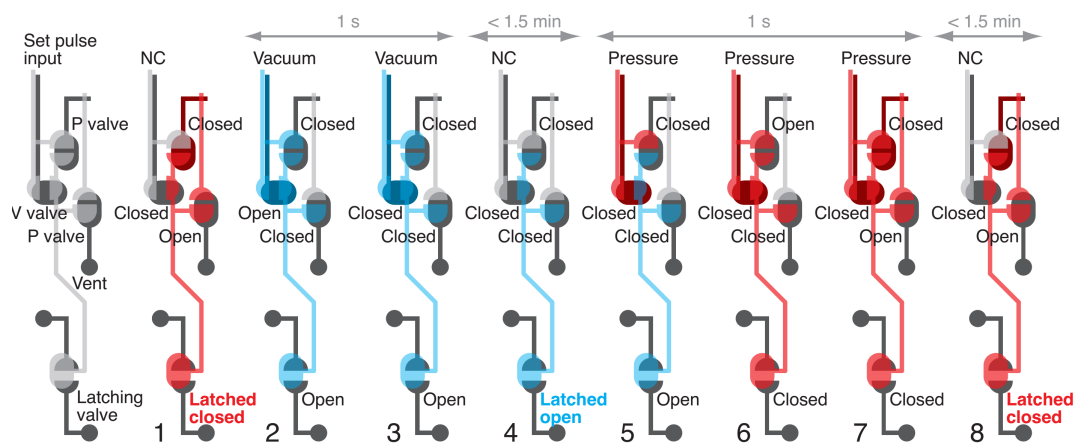


Figure 2.17: Pressure / Vacuum Latch-Valve [43]

Chapter 3

Design Methodologies

This chapter contains contributions made in P1, P2, P3 and P5, as stated in Section 1.4.

As biochips become capable of increasingly complex tasks, new design methodologies are necessary to keep the design process feasible. However, no single technology or set of components has been identified yet around which methodologies can be build. Instead, a flexible approach is required that does not only allow for a wide range of applications as input, but can also incorporate evolving technologies.

Fig. 3.1 shows an overview of our design process for FBMBs, resulting in both a manufacturable, physical design of a biochip as well as its logic counterpart that manages the active parts of the chip during the application's execution. In this process, information about the used technologies and components is acquired by every design step individually as required. This does not only allow for more flexibility and easier comparability between technologies, but also enables the creation of partial designs, without having determined a technology to be used yet. This, still general, overview indicates the underlying complexity of microfluidic biochips and therefore the need for automation and optimization of the processes as presented in this thesis.

Over the past years, first attempts have been made to automate individual parts of this progress such as binding, routing and scheduling [24, 82], physical design [70], control synthesis [78] and on-chip control logic [22], which are in part presented in this chapter.

In a sequential process of this design, results of previous steps would be used in future steps. However, many parts of the design process predict an output of a later stage such as a control synthesis prediction during physical synthesis of the flow-layer. Such predictions are necessary in order to find a solution in reasonable time or to increase the quality of the overall design. This does however lead to optimization loops. Every design step D that uses a prediction of a future step $S_{Predicted}$ as input, could potentially be revisited once the actual solution to step S is available as input. The revisited step $D_{Revisited}$ would in turn create another solution $S_{Revisited}$ which could be used as input, creating an optimization loop. Instead of an iterative approach, a “Co-Design” of these design steps is required [16]. Attempts have been made, but mostly result in an iterative design on a smaller scale [120].

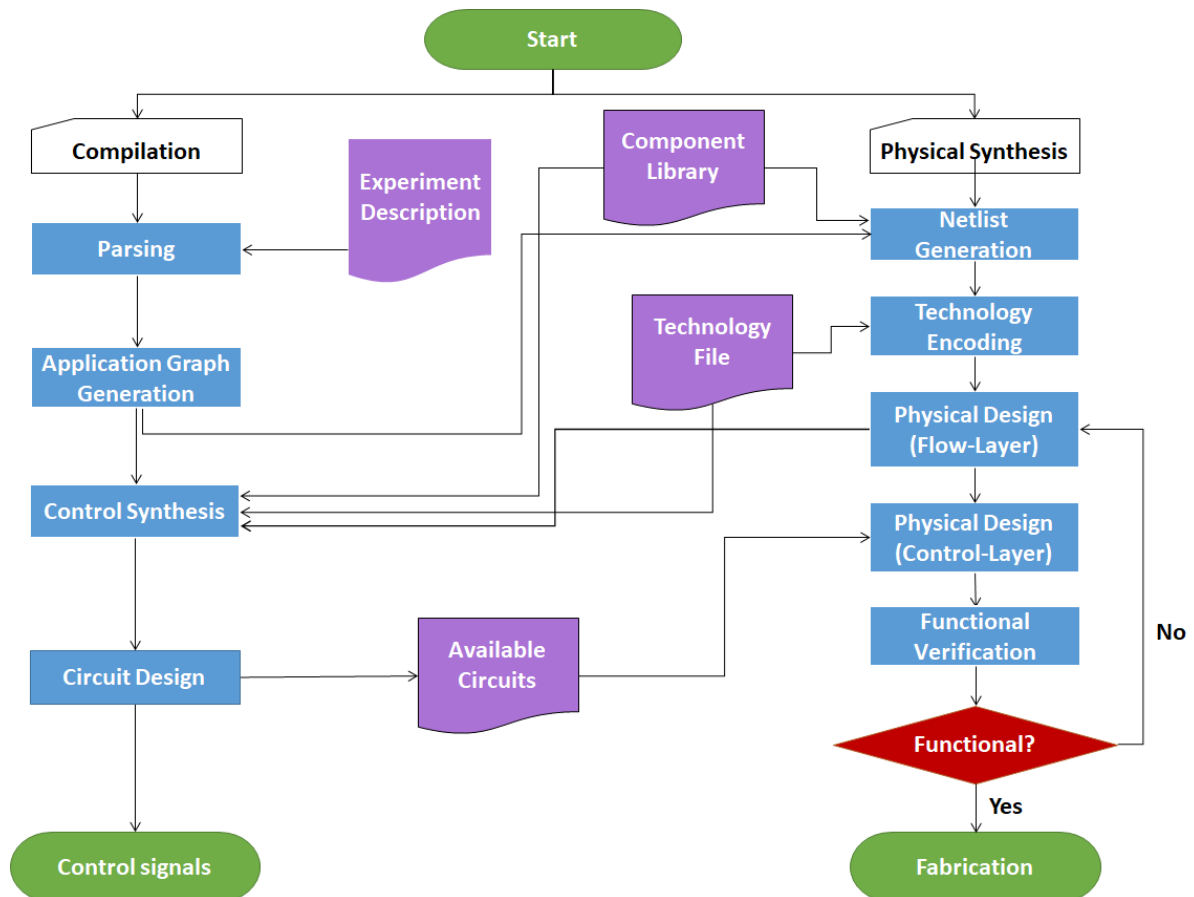


Figure 3.1: Overview of the design of FBMBs

3.1 Binding, Routing and Scheduling

The way in which a given application such as in Fig. 2.12 is mapped to an architecture as in Fig. 2.11 has a significant impact on the design steps of physical design and control synthesis from our design process overview in Fig. 3.1. The binding of operations determines which FFU will be used to execute any given operation. Routing determines the pathways that the fluid takes from one bound FFU to another. Scheduling determines the order in which the operations are executed. While some operations such as O_1 , O_2 and O_3 in Fig. 2.12 can be scheduled in any order, other operations such as O_3 and O_4 have to obey their sequential order. Binding of operations is very restricted for most architectures, as operations need to be bound to FFUs that can execute the desired functionality, e.g. an incubation operation has to be executed in an incubator component. Furthermore might other restrictions apply such as the maximum volume of a component which we have omitted for our basic example. However, if multiple components providing the same functionality are available, the binding of operations can have significant impact on the routing and scheduling of the operations. It is therefore beneficial to only determine all binding possibilities at a first stage and decide the specific binding to a single FFU during the routing process to avoid impractical routes to a previously bound FFU.

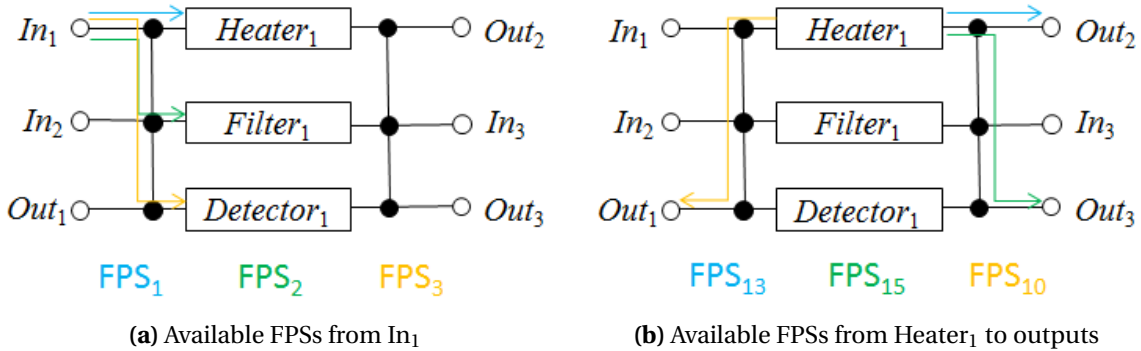


Figure 3.2: Microfluidic components

Some previous work has simplified the routing problem, by ignoring the need of pressure to move fluids, resulting in fewer components being occupied during fluid transport [79]. Our routing algorithm does not include any simplifications and determines functional routes without imposing additional constraints. We construct a Flow-Path (FP) by linking together multiple Flow-Path Segments (FPS) which are determined first, as explained in Sect. 2.4. We use the term “route” to refer to those FPs which have been chosen to be used by an operation,

distinguishing them from other available FPSs not used as routes. From this list of FPSs (which are stated in Table 2.1 for our example) we can determine FPSs to suit the operations of the application. We divide this process into three steps. (1) A connection between the FFU where the fluid is currently located and the target FFU (as determined by the binding process) has to be found. For example, operation O_1 from Fig. 2.12 requires fluid from In_1 to be transported to a heater. O_1 can execute on any heater, and our routing will search for a heater FFU to bind operation O_1 to (in this simple example there is only one option). We determine a path between these FFUs using one, or a set of connected FPSs. This is done using a Breadth-First Search algorithm, with all FPSs beginning at In_1 as starting points. Additional FPSs are linked to the established FPSs until a connection between the FFUs has been found. Fig. 3.2a shows all FPSs starting at In_1 . Since FPS_1 directly connects In_1 to $Heater_1$, it is chosen as part of the route for operation O_1 while the other options are discarded.

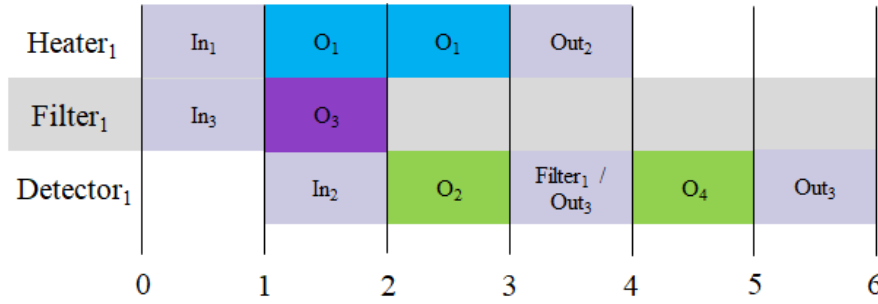


Figure 3.3: Schedule example for the application from Fig. 2.12 running on the architecture from Fig. 2.11

Two additional steps are necessary to form a valid FP. These steps have commonly been ignored in previous work to simplify the process, but are crucial for realistic behaviour of the model [80]. (2) The path determined in (1) has to be connected to an output. To the heating operation O_1 we bound $Heater_1$ and therefore need to connect our partial FP to an output from there. This is done using the same Breadth-First Search algorithm as in (1), only with $Heater_1$ as the starting point and any output as target. However, during this search conflicts with the previously determined path from In_1 to $Heater_1$ have to be omitted. Considering all available FPSs that lead to outputs as shown in Fig. 3.2b FPS_{10} cannot be used to connect $Heater_1$ with Out_1 , since this FPS overlaps with FPS_1 which is already in use. For operation O_1 , we therefore use FPS_{13} to connect $Heater_1$ to Out_2 without causing any overlap. (3) The path determined in (1) has to be connected to an input to be considered a valid FP. For operation O_1 this is already the case and no further action is required to complete the route. For operations that do not start at an input, an input has to be found in the same

fashion as an output in (2). Depending on the architecture, it is possible that multiple paths connecting the source and target FFU exist or multiple components of the same type to which an operation can be bound (e.g. Heater, Detector) are available, meaning we will end up with multiple options for all three routing steps, e.g. in Fig. 3.2b we could have also used FPS_{15} to connect $Heater_1$ with Out_3 . While choosing the shortest route is beneficial in many cases, other methods have been presented [2], and we discuss our own in more detail in Chapter 4. To schedule applications, prominent methods from computer science can be applied. Techniques such as List Scheduling [73] which optimizes the overall execution time of an application provide solutions applicable to our microfluidic system. For our scheduling, we allow parallel execution of operations if their routes do not overlap. Furthermore we assume that operations bound to a FFU (i.e. all operations that are not transport operations) do not use any part of the chip aside its bound FFU. For simplicity reasons, in this example, we assume that any operation (including transport) takes 1 minute, except for the heating operation for which the application states an execution time of 2 minutes. From these constraints the scheduling algorithm can determine a schedule as shown in Fig. 3.3 for our example architecture and application. As transport operations to both the filter and detector can't be executed at the same time, the transport operation to the detector was delayed until time step 1. In turn, once the filter operations is finished, the fluid can't be moved to the detector while it is still in use from O_2 and this transport operations is therefore delayed to time step 3. Here, two transport operations are combined, moving fluid from the filter to the detector while moving fluid from the detector to Out_3 , which is possible as both share the same route. Optimal (i.e. shortest overall execution time) schedules are frequently not unique, as the alternative schedule in Fig. 3.4 shows.

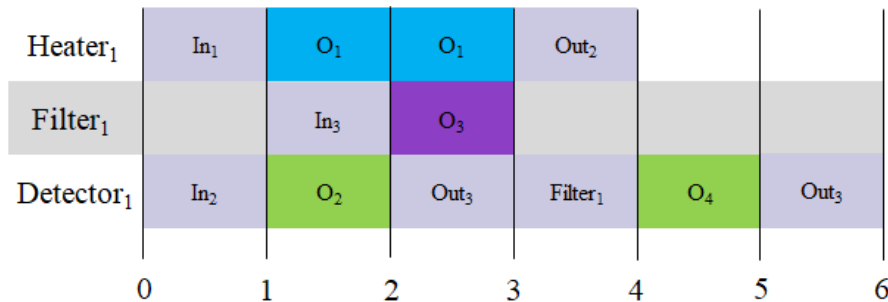


Figure 3.4: Alternative solution to the schedule in Fig 3.3

3.2 Component allocation and schematic design

A schematic design of a biochip (i.e. a netlist in Fig. 3.1) that follows our Biochip Architecture Model from Sect. 2.4 can be constructed from an application graph, using a given component library. To do so, the netlist is required to allocate components capable of executing all operations given by the application model. Furthermore, these components have to be connected in such a way, that the order of operations from the application model can be satisfied when executed on such a hardware. The allocation and schematic design of an architecture does not determine the location or size of components, or the length and route of channels connecting them, which is part of the Physical-Synthesis. A possible solution to the allocation and schematic design is presented in [96]. The component allocation is determined through a predicted schedule, created from the application graph. This preliminary schedule determined which operations can be executed in sequence and which have to be executed in parallel, due to given constraints. Parallel execution of, for example two heating operations, then requires the allocation of at least two heating components. The allocated components are then bound to the operations in the schedule, allowing to determine the necessary connections between these components to transport fluids to sequential operations. As this synthesis problem is NP-Complete [116], heuristic algorithms are applied to provide solutions in reasonable time, which are however not guaranteed to be optimal.

3.2.1 Microfluidic HDL

In efforts to alleviate the strain of biochip end-users on hardware design, a Microfluidic Hardware Description Language (MHDL) has been proposed in [69]. There, the language and accompanying compiler, verification tool and simulator are presented. Analogous to the electronic design automation leading to VHDL, the goal of MHDL is to separate the end-user from the hardware designer. MHDL allows specialized designers to take over the design task, including functional verification and performance estimation and provide the end-user with a suitable design.

3.3 Faults and Fault Tolerance

This subsection is based on the contribution P4 as described in Sect. 1.4.

Many potential use cases for biochips such as patient care, or on-site sampling require a high level of reliability from the devices that are used. Faulty devices can result in the loss of expensive or hard to obtain samples and reagents, while undetected errors can cause far reaching issues for example in case of misdiagnosis in healthcare. Before widespread use of biochips is possible, their reliability has to be increased and demonstrated to be at a level that at least rivals current methodologies.

Previous work has identified and classified faults in FBMBs into a *Fault Model* [50]. Such fault models allow to better distinguish between different potential faults and apply fault-detection and -tolerance methods more efficiently by adding additional abstraction layers. The fault model classifies faults in two main categories: **Permanent Faults** and **Transient Faults**.

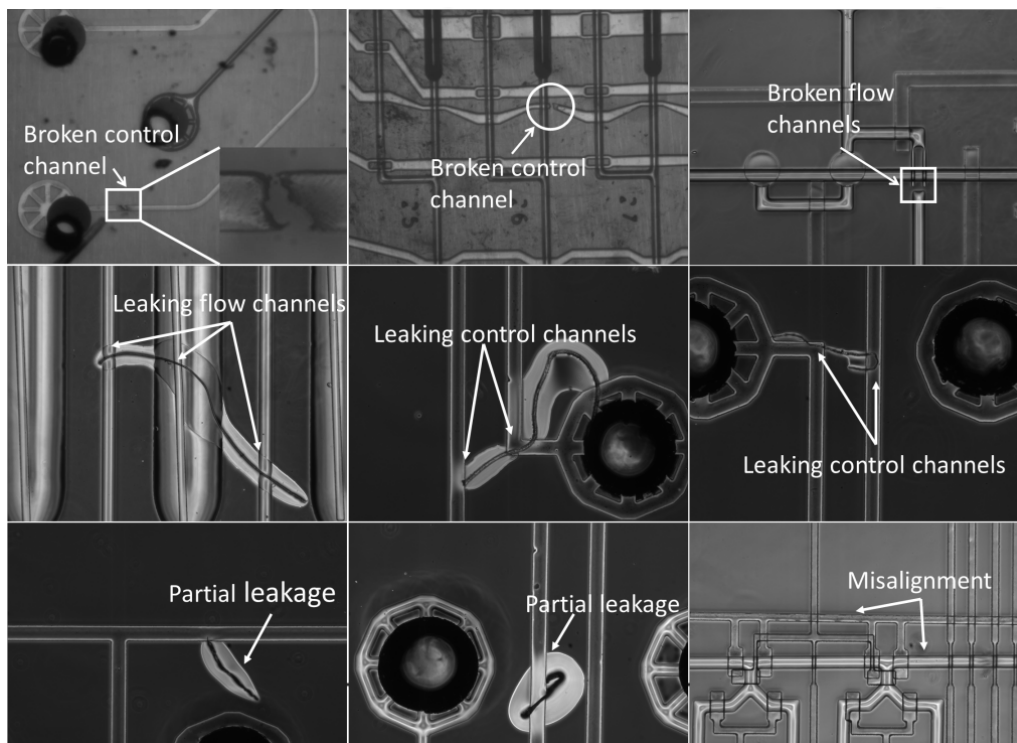


Figure 3.5: Common Faults in FBMBs [50]

Permanent Faults, examples of which are listed in Table 3.1, are caused by physical defects, usually introduced during the fabrication of the biochip or caused by worn out components.

Cause of Defect	Observable Error	Fault Model
Foreign particles, imperfect molding and hard baking	Transmission of pressure is interrupted	Block in flow channels or control channels
Defect in bonding between two independent channels or components	Cross-contamination	Leak between flow- or control- channels and components
Pumps fail to generate pressure	Missing transmission pressure	Block in flow channels or control channels
Membranes of valves lose flexibility or even get perforated	Valves cannot seal flow channels	Block in control channels

Table 3.1: Examples of faults occurring in FBMBs

Permanent faults prevent operations from executing correctly and can result in recoverable and irrecoverable faults. For example, a stuck close valve is blocking the access to a component, making the use of this component impossible. If this is the only components of its type, the fault is irrecoverable, otherwise the operations can also be executed on another component and the fault is considered recoverable. Chips with permanent faults are always discarded, even if they are considered recoverable as these fault will still have significant negative impact on the execution of the application. Recovery from permanent faults is therefore only of interest if discovered during the execution of an application, where error recovery can allow the application to finish successfully, avoiding long experimental times, wasting reagents and samples.

Transient Faults can occur during the execution of an operation, but are not permanent. This means that neither previous nor following operations will necessarily undergo the same fault. Such faults furthermore do not prevent the operation from executing, but will cause the result of the operation to deviate from its specified solution. All transient fault are therefore considered recoverable, for example by re-executing the operation from a previous, correct state. The main issue concerning transient faults lies therefore in their detection, as occurring faults might not be obvious, but can significantly alter the outcome of an application. Faults occurring in flow-based biochips are dominantly permanent faults as Table 3.1 shows. However transient faults might become more common due to newly introduced functionality such as on-chip metering.

Having identified common causes for faults, preventive and reactive measures can be taken to provide better reliability. To better distinguish between the application of each measure, Fault-Tolerance can be split into four major areas, each allowing to increase the overall reliability of biochips [50]. (1) **Offline Fault Detection** methods can determine the functionality of chips before use. Fabrication errors as well as long term use or transport can introduce faults to

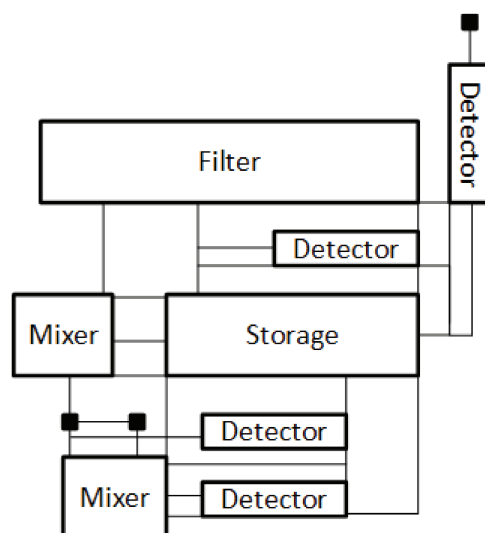


Figure 3.6: Fault Tolerance through redundant architecture design [51]

biochips. Test routines are run prior to their use allowing to discard potentially erroneous ones. (2) These methods furthermore need to be extended to allow **Online Fault Detection** during the use of the chip. Uncovering faults that occur spontaneously or that were not detectable before, assures that erroneous results are minimized. This also introduces the possibility of (3) **Error Recovery** which can allow to finish current operations despite of an occurring error, potentially saving reagents, data and time. Lastly (4) **Fault Tolerant Designs** introduce alternative functionality to the base architecture, that can be used in case of an occurring fault. This will aid to increase the yield of biochip fabrication, and furthermore also improve the chance of error recovery [51].

The majority of work for flow-based biochips is focused on fault tolerant design and offline fault-detection. While online fault-detection and error recovery options are available for DBMBs [62], equivalent solutions for FBMBs are very limited to date. Looking more closely into the four major areas of fault-tolerance in FBMBs we find that Fault-Tolerant Design in flow-based biochips can be simplified into a routing and binding problem. All four fault models in some way impact the routes and components on chip, and therefore risking erroneous execution of an operation. Current work in fault tolerant design therefore focuses on providing redundancy within the chip, which allows faults to be tolerated. In a naive solution, fault-tolerance is achieved by adding enough redundant channels and components to provide functional solutions for every possible occurring fault. This does however significantly increase the design, fabrication and operation cost of the biochip. Assessing the likelihood

of faults and the cost associated with fault tolerance is therefore important. One proposed solution is to change the physical design of components to make them more fault-tolerant instead of implementing additional redundant components [51]. These altered designs have the advantage that high-risk parts can be backed up with redundant parts, while others are not, which lessens the impact on the design, fabrication and operation cost. For example, as shown in Fig. 3.6 the pump of a mixer, which is the highest-risk part of a mixer, can be made fault-tolerant with the addition of a single valve, at low additional cost. If one of the pumping valves fails, the fourth valve can replace it.

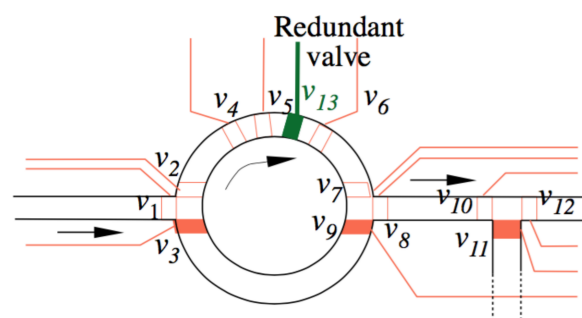


Figure 3.7: Fault Tolerant Components [29]

The possibility for offline fault detection has already been demonstrated by determining the fault model [49, 50]. There the integrity of routes can be tested by applying pressure to a flow-channel, attaching a pressure sensor at the predicted output and opening the valves along the route. This only determines that some part of the biochip is damaged, but does not precisely identify the source. However, as chips with any kind of fault are significantly less reliable, any fault warrens discarding the chip, making the specific source irrelevant.

Providing similar detection methods for online fault detection as proven to be more difficult, as the channels are occupied by fluids, pressure based detection is no longer a straightforward solution. Instead, optical detection using CCD cameras has been proposed to detect faults. Fluid reaching certain components can be captured and even the volume can be determined by analysing the length of the fluid occupied channel. Such measurements are however highly reliant on the chip architecture, such as the channel depth and width which might be too small to be accurately captured by the camera and therefore need to be provided. Additional variables have to be accurately described such as the camera's distance and angle to the chip. Further difficulties arise due to the fact that both the chips components as well as most reagents are transparent, resulting in very low contrast between them. So far, such optical detection has mostly been done manually using microscopes, which has also proven difficult

[45]. Alternatively, a pressure detection based fault detection using vents can potentially capture certain faults, which is part of the contribution presented in Chapter 6.

3.4 Fabrication

Fabrication of FBMBs can be done using several methods. Due to the vast differences in design and materials used for biochips, fabrication methods perform very differently depending on such parameters. Furthermore, while some techniques such as injection-molding have been virtually perfected, others such as 3D printing still benefit from ongoing advances leading to increasingly more efficient fabrication. The today most commonly used fabrication methods for FBMBs are Injection-molding [46], 3D printing [118], micro-milling [124] and laser-cutting [108].

Injection-molding has been used for decades not just for medical components, but anything from gameboys to bottlecaps using metals, glasses and thermoplastics such as PMMA. It uses masks (or moulds) commonly built from tool steels, into which the heated material is poured (or forced). The material hardens as it cools down and retains the shape given by the mask. While the design of the mask can be time consuming and the fabrication costly, once it is done the production of the final product is trivial. The process is therefore very optimized, allowing for cheap, high-quality and scalable fabrication. This technique is used dominantly for passive biochips (Sect. 2.1), as the whole chip can be poured from a mask. The nature of this process does however make it difficult to include moving parts in the chip. Some chips have been demonstrated that include components such as mechanical switches [12].

3D printing is in many ways similar to injection-molding as the result is a biochip consisting of a single piece. 3D printing makes it a lot easier to change design as it doesn't require a mask, but only a different design. However, other issues emerge, such as limitations in precision. While the layer resolution of 3D printers has vastly increased over the last years, it cannot yet rival the resolution of micromilling machines. Especially considering that the smallest layer resolutions can only be obtained using larger nozzles [99].

For better access to active components on a biochip, for example through the use of Grover valves (Sect. 2.2.2) a membrane needs to be introduced to the biochip. However, both injection-molding and 3D printing are not well suited for such multilayer, multi-material

designs. The most commonly used material for such membranes is PDMS due to its high elasticity and tear resistance. At its glass transition temperature of -45°C it can easily be formed into any desired shape, which is why it has also been used as material to build entire biochips from it. For multilayer design, thin, even sheets are required which are commercially available at various thickness, down to $20\text{ }\mu\text{m}$ [61]. As we have shown in Sect. 2.2.2, this membrane of PDMS is sandwiched between the two other layers, all three of which then need to be bound together to form a single device. This so far marks the limit of injection-molding and 3D printing. The PDMS is holding on to the other layers through Van der Waals forces [58]. For enough of these weak interaction to occur, the surfaces need to be extremely even. However, both injection-molding and 3D printing can only create surfaces with a roughness several times of that of PMMA sheets [52, 56]. The reduced contact surface of this rougher material make Van der Waals forces based bonds therefore very weak.

3.4.1 Multi-layer milling and bonding

Micromilling is an excellent choice for rapid prototyping and low quantity fabrication [44]. Since the design of the chip can be easily changed by adjusting the G-Code interpreted by the machine, there is no additional cost for masks or similar when changing the design. While the size and cost of micromilling machines can be high, smaller, tabletop version which also produce good results are available [114]. Scaling the production of biochips using micromilling machines is however difficult at this time, as significant manual intervention is still necessary. For most chip designs, multiple mills (of different size) are required and mills therefore need to be changed part way through the fabrication process. Moving the chip between multiple machines to avoid exchanging mills instead causes positioning issues. Position offsets caused by moving the chip to another mill of just a few μm can cause alignment issues. Therefore significant advancements that solve these issues still need to be made before micromilling can rival the cost efficiency of techniques such as injection-molding.

To validate the work carried out for this thesis, prototypes have been built to test the suggested features. All biochips fabricated for this thesis use Grover Valves (Sect. 2.2.2), therefore having individually fabricated flow- and control-layers which are separated by a thin layer of PDMS after bonding. For the flow- and control-layer we use cast plexiglass (PMMA) pieces [100] sized $70\times 70\times 3\text{ mm}$. While extruded plexiglass has some advantages such as better

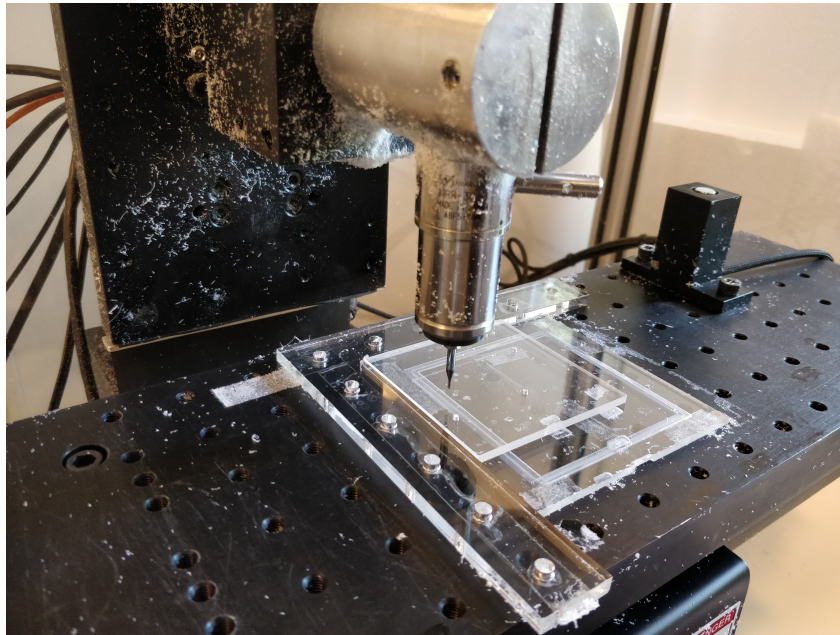


Figure 3.8: Minitech Machinery Mini-Mill/3 with PMMA piece aligned to a bracket

thickness tolerance and lower price, it also comes with various downsides. Lower resistance to chemicals as well as to physical force (scratches easily) can cause problems for biochips. However, most notably is the low melting point of about 80 °C which can easily cause parts of the extruded plexiglass piece to warp or even melt during fabrication. Cast plexiglass on the other hand has a melting point of about 160 °C and can sustain the heat created during the fabrication process [100]. The poor thickness tolerance of cast plexiglass ($\pm 15\%$ for 3mm thick sheets) can be fixed by using a fly-cutter [21]. These layers are separated by a layer of PDMS [61] of different thickness, typically 100 μm to 250 μm . Channels and components are milled into the PMMA with a Minitech Machinery Mini-Mill/3 Micromilling machine [63], using flat-end mills at 20000 RPM. Through holes are milled using a 3 mm diameter mill. Channels are milled using a 800 μm mill at varying depths. For more precise milling (e.g. vent groves in Chapter 6) a 200 μm mill is used. Fig. 3.8 shows the micromill with a piece of PMMA aligned to a bracket for correct placement and a sacrificial piece underneath for through holes.

The PMMA layers are bound to the PDMS using a hydraulic press with heatable plates. To do so, the PDMS sheet is placed on top of either PMMA layer and the second layer is placed and aligned manually on top. This loose bonding is then strengthened in the heated hydraulic press. The applied pressure removes most of the trapped air as it is pushed towards a channel

```

1 G00 X5 Y5 Z5
2 M03
3 G00 X51.9552 Y56.6016 Z1.0
4 G01 Z-0.2 F700
5 X51.9552 Y23.7952 F700
6 Z1.0
7 G00 X11.264 Y11.4048 Z1.0
8 G01 Z-0.2 F700
9 X41.3952 Y11.4048 F700
10 Z1.0

```

(a) Basic G-Code example

```

1 %
2 O00001G103 P1 ;
3
4 #800= 100 N100
5 IF [ #801 GE #800 ] GOTO200
6 #801= #801 + 1
7 IF [ #801 LE #800 ] GOTO100 N200
8 #801= 0 M30 ;
9
10 %

```

(b) Example of G-Code with macros

Figure 3.9: G-Code examples


- 
3. Add 1.5 vol. CTAB to each MCT and vortex. Incubate at 65° C for 10-30 mins.
 4. Add 1 vol. phenol:chloroform:isoamylalcohol (48:48:4) and vortex thoroughly.
 5. Centrifuge at 13000g at room temperature for 5 mins.
 6. Transfer aqueous (upper) layer to clean MCT and repeat the extraction using chloroform:isoamylalcohol (96:4).

Figure 3.10: Ambiguous assay description [7]

or the edge of the chip. The heating of the material aids this process as the material softens and the air expands, making it more likely for air bubbles between the layers to be pushed out of the chip. We heat the plates of the hydraulic press to 70 °C during the bonding process. While the PMMA generally has a much higher melting point at 160 °C, the material already softens significantly at temperatures well below. We found that even at 100 °C, the material becomes soft enough that warping can occur under the pressure of the press. The duration of the bonding process varies depending on the thickness of the PDMS and the PMMA, which determines the time it takes for the material and the trapped air to warm up. Using PMMA pieces of 3 mm thickness and PDMS membranes of 250 μm , thickness, a bonding time of 10 minutes has been sufficient. We have found that the applied pressure should not be higher than about 8 kN. Applying more pressure causes the PDMS membrane to be squeezed into channels and other open areas in the chip. While this does not cause the membrane to tear, the additional material that was forced into the channel pushes back as soon as the pressure is released, weakening the bonding. Furthermore should the chip undergo a cool-down process while still under pressure. The sudden change in temperature of the material and left over trapped air when exposed to room temperature can otherwise cause the bonding

to fail immediately. We found that cooling the chip to about 45 °C while also decreasing the pressure during the cooling phase resolves this issue.

<pre> 1 ASSAY Glucose START 2 fluid Glucose, Reagent, Sample; 3 fluid a, b, c, d, e; 4 VARResult[5]; 5 input Glucose 50; 6 input Reagent; 7 input Sample 30; 8 conflict Sample FOLLOWS Glucose WASH water; 9 a=MIX Glucose AND Reagent IN RATIOS1 : 1 FOR 10; 10 SENSE OPTICAL it INTOResult[1]; 11 b=MIX Glucose AND Reagent IN RATIOS1 : 2 FOR 10; 12 SENSEOPTICAL it INTOResult[2]; 13 c=MIX Glucose AND Reagent IN RATIOS1 : 4 FOR 10; 14 SENSEOPTICAL it INTOResult[3]; 15 d=MIX Glucose AND Reagent IN RATIOS1 : 8 FOR 10; 16 SENSE OPTICAL it INTOResult[4]; 17 e=MIX Sample AND Reagent INRATIOS1 : 1 FOR 10; 18 SENSE OPTICAL it INTOResult[5]; 19 END </pre>	<pre> 1 Glucose{ 2 --input s3,ip3 ;Glucose; 3 --input s4,ip4 ;Reagent; 4 --input s5,ip5 ;Sample; 5 --washport w1 ; 6 move mixer1, s3,4.47; 7 move mixer1, s4,4.47; 8 mix mixer1, 10; 9 move sensor2, mixer1; 10 sense.OD sensor2, Result(1); 11 move mixer1, s3,2.98; 12 move mixer1, s4,5.96; 13 mix mixer1, 10; 14 move sensor2, mixer1; 15 sense.OD sensor2, Result(2); 16 move mixer1, s3,1.79; 17 move mixer1, s4,7.15; 18 mix mixer1, 10; 19 move sensor2, mixer1; 20 sense.OD sensor2, Result(3); 21 move mixer1, s3,0.99; 22 move mixer1, s4,7.95; 23 mix mixer1, 10; 24 move sensor2, mixer1; 25 sense.OD sensor2, Result(4); 26 wash w1; 27 move mixer1, s5,4.47 28 move mixer1, s4,4.47; 29 mix mixer1, 10; 30 move sensor2, mixer1; 31 sense.OD sensor2, Result(5); } </pre>
(a) Aqua code example	(b) AIS code exmaple

Figure 3.11: Code examples [6]

3.4.2 G-Code

The G-Code programming language is used to operate many computer aided fabrication tools such as micromilling machines or laser cutters. Newer implementations include macro language capabilities somewhat similar to those of a high-level programming languages. As shown in Fig. 3.9b, this includes setting of variables (#), if-statements and go-to commands. However, G-Code is also still commonly used in its more basic, sequential operation form as shown in Fig. 3.9a. This only includes G-Code commands (starting with G and M) and coordinates. While a significant number of G-Code commands are available (see a full list here [19]), the main functionality it provides is the linear and circular interpolation between two coordinates, allowing to create straight cuts or arcs. Where these coordinates are and at what speed and order they should be connected is completely left to the programmer. Due

to this low level exposure, it is impractical to write G-Code manually, unless for very simple projects. Tools such as our Biochip Designer Tool presented in Appendix A or other available tools [122] have therefore been developed which generate G-Code, for example using input from a graphical user interface.

```
// step 1
first_step("Preparation of cells");
inoculation(flask, bacteria, 37, time(12, HRS), 1);

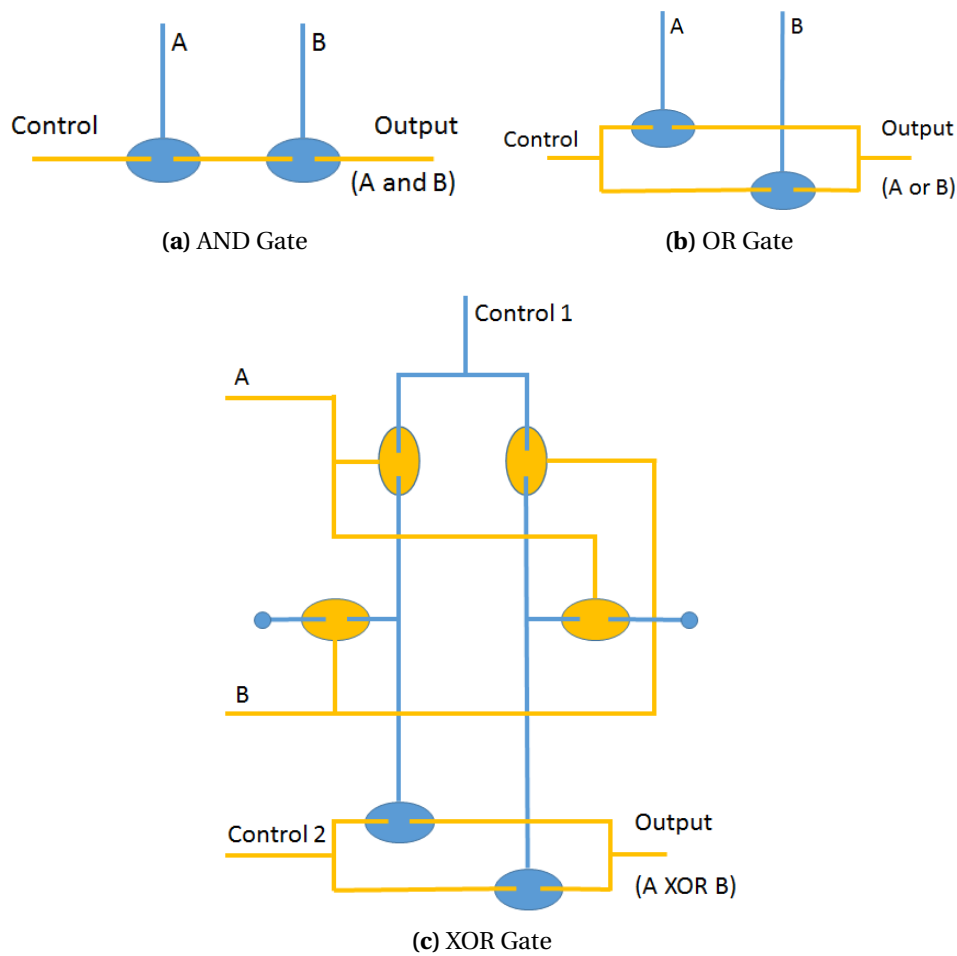
// step 2
next_step();
measure_fluid(flask, vol(1.5, ML), microfuge_tube);
centrifuge_pellet(microfuge_tube,
    speed(SPEED_MAX, RPM), 4, time(30, SECS));
comment("Leave the pellet as dry as possible.");
```

Figure 3.12: Biocoder code example [7]

3.5 Programmability

The continued advances in microfluidic technology allows us to recreate components from computer- and electrical engineering for use in biochips. Previous research has demonstrated the used of microfluidic logic gates, adders [54], shift-registers [22] and finite state machines [87] in the past. Such advancements don't only allow for more complex behaviour, but also for more general architectures. Instead of application specific design, multi- or general purpose biochips are being proposed, which can be programmed for many different uses. Similar to programming languages like C++ or Java for computer programs, a more abstract way to describe biochemical applications for biochips is necessary which hides the increasing, underlying complexity of the architecture.

Furthermore do such languages remove all ambiguity from protocols as the code can only be interpreted in a single way. This removes a source of possible errors as language based protocols such as the example in Fig. 3.10 can be interpreted in multiple ways. High-Level protocol Languages (HLL) such as Aqua [6] and BioCoder [7] have been proposed, to turn the abstract information of the application into an assembly language, such as the MHDL presented in 3.2.1. The Aqua language provides its own syntax and keywords from which

**Figure 3.13:** Gates

a source code can be constructed as shown in Fig. 3.11a. Creating such a domain specific language makes it more accessible to non programmers which represent the majority of biochip end-users. Besides the HLL, Aqua also provides its own instruction set called Aqua Instruction Set (AIS) shown in Fig. 3.11b into which the aqua source code can be compiled. Biocoder on the other hand is created as a library for C++ and therefore integrated into this well established programming language. This allows to use widely available IDEs, compilers and debuggers to create code in standard C++ syntax as shown in Fig. 3.12.

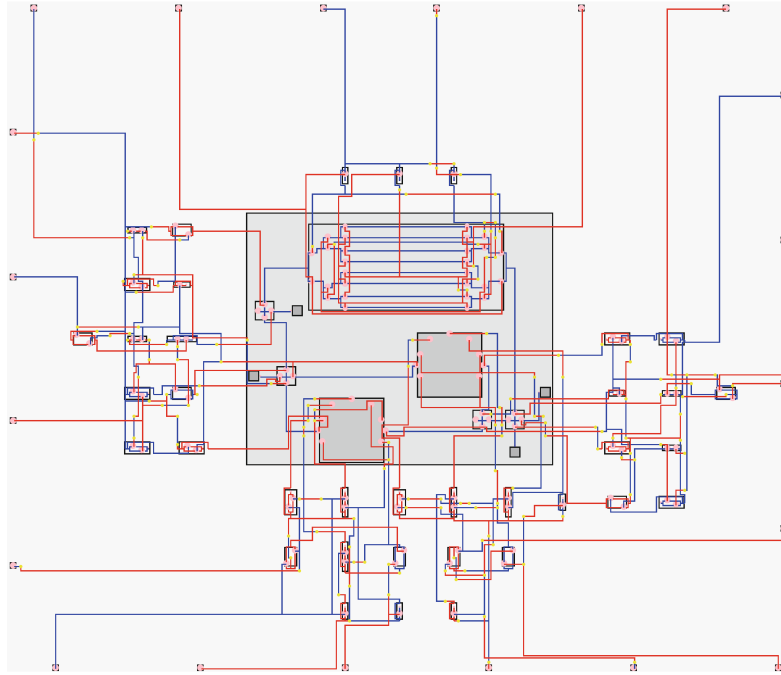


Figure 3.14: PCR Biochip with two Mixers, eight capacity storage and two in- and outputs, optimized to reduce the number of required control-pins with on-chip control [48]

3.6 On-Chip Control

The complexity of biochip architectures is significantly limited by the necessity of external hardware to actuate valves on-chip, and therefore provide the control logic. Recent work has however allowed to move some or even all of this control logic on-chip, drastically reducing the number of pressure sources required [86, 102]. Such control logic is build form logic gates as they are also used in electronics. Due to the analogy of pneumatic valves and electronic transistors, logic gates such as AND-Gates, OR-Gates, XOR-Gates, etc. can be build in a similar fashion for biochips as they are for electronic microchips [54]. Fig. 3.13 shows examples of such pneumatic logic gates. The distinction of Flow-Layer (yellow) and Control-Layer (blue) that previously indicated which channels would contain liquids and which air pressure, is not accurate for control gates. Here, both layers are used for air pressure transport, the distinction is merely kept to distinctly refer to the layers. This allows to restrict air flow into certain channels and obtain an output of either high pressure or low pressure, depending on the states of the gates. To build more complex control logic, multiple gates can be connected, allowing to encode specific circumstances in which flow controlling valves should be actuated.

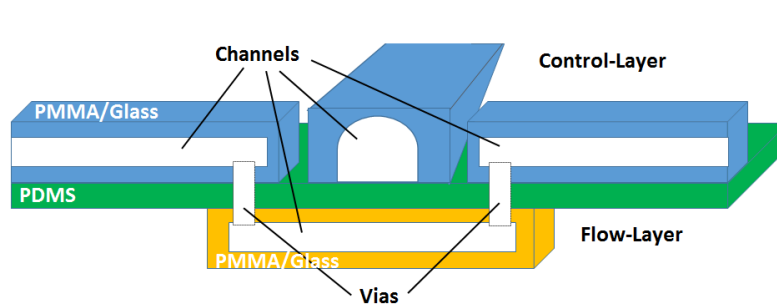


Figure 3.15: Schematic design of the use of Vias to avoid an intersection of two channels on the same layer

Fig. 3.14 shows a PCR biochip which has been extended with on-chip control logic. Through this, the 62 integrated valves (which otherwise would each need an independent connection) can be controlled using only 20 control-pins. The required control logic, shown in red (flow-layer) and blue (control-layer) far exceeds the biochip area (grey), which has to be considered as a trade-off.

3.6.1 Layer Changes

A *Layer Change* (also called *Via* or *Through Hole*) allows to connect channels across the separated layers. On biochips fabricated for Grover valves this means that the PDMS membrane is penetrated to connect overlapping channels on the flow- and control-layer as shown in Fig. 3.15. Fig. 3.14 in fact uses many such Vias, which are essential for successful and efficient routing. When integrating vias into an architecture, the distinct roles of the flow-layer to carry fluids and the control-layer to carry pressure to actuate valves overlap as both fluids and pressure can be passed to the other layer. This extends to the possibility to place control-pins as well as in- and outlets on either layer of the biochip.

For fabrication, vias simply represent holes in the PDMS membrane (and overlapping channels on both layers). Previous research has shown that it can be sufficient to punch the holes into the PDMS using a needle or similar. However, during our own experiments we encountered severely reduce air flow through vias punched using a needle, especially when alternating between high and low pressure. The same research suggests the use of laser cutters to burn holes into the PDMS [26]. As the material evaporates using this technique, it is less likely for the via to be obstructed during use.

Chapter 4

Pin-Count Reduction

This chapter is based on the contributions P1 and P2 as described in Sect. 1.4.

4.1 Introduction

Fluids in continuous flow biochips are moved by applying pressure to the flow channel containing the reagents. In passive biochips as introduced in Sect. 2.1, such a single pressure source is sufficient to drive the, be it limited, functionality of the biochip through intricate design of channels and chambers. To manipulate the flow of fluid within the biochip for more advanced functionalities, valves need to be actuated with additional pressure sources. Such pressure is introduced to the chip through control-pins as shown in Sect. 3.4 and typically generated by external pressure sources and controllers as explained in Sect. 2.6. The high cost, weight and size of such external hardware only remains practical until a certain point, which puts a limit on the complexity of biochips. We therefore propose a pin-count reduction by means of valve control sharing. On a logical level, valve control sharing removes the possibility for two or more valves to be in individual states (open, closed or don't care, see Sect. 2.6.1) and forces them to be in the same state at any time, forming a Valve Group (VG). On a physical level, the control channels leading to each of the valves in a VG are connected and all but one control-pin is removed, causing the pressure at every valve in the VG to be identical at any time.

Control-pin minimization using this general idea has been proposed before in [78, 102] based on application mapping and scheduling [81]. Here, control-pin minimization is based on the scheduled application. From the schedule, valves sharing the same state throughout the application can be identified. Such valves have no need for individual actuation and can form a valve group. However, with this approach, control-pin minimization is depended on a specific schedule. This prohibits the formation of certain VGs, for example if two valves V_1 and V_2 are in opposing states at any time, they can't be part of the same VG. However, a different schedule might exist where V_1 and V_2 are never in opposing states allowing them to form a VG.

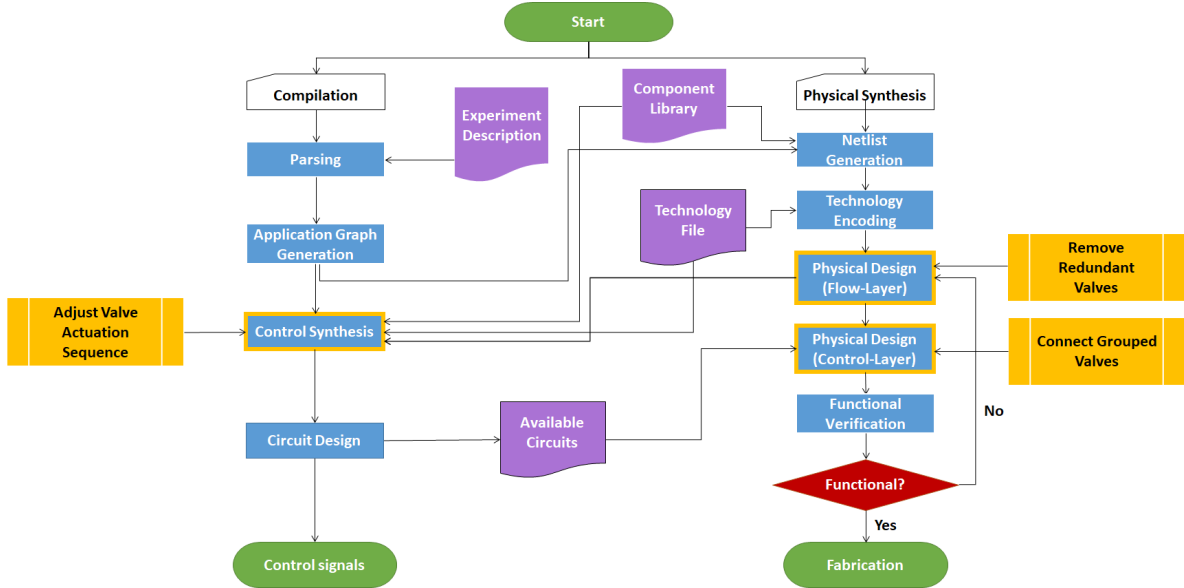


Figure 4.1: The proposed pin-count reduction technique is implemented in the three highlighted design steps, control synthesis and the physical design on both layers

Given a solution space θ of all possible schedules, a scheduler will find an optimal solution γ_i , i.e. a solution with the shortest possible execution time, for the given application. Iterating through the entire solution space θ to find all optimal solutions $\gamma \subseteq \theta$ and determine which allows for the most VGs is, however, computationally impractical. Therefore we base our pin-count reduction algorithm on an earlier stage in the design process, looking at the Flow Paths (FP) and Flow Path Segments (FPS) for the routes of the fluid through the biochip, to determine viable VGs. Similar to how a fixed schedule limits the options to form VGs, so do fixed VGs limit the options for viable schedules. For example, a VG consisting of V_1 and V_2 invalidates all schedules that would require these valves to be in opposing states at any

time. Therefore, every VG that is introduced to a given architecture, removes a subgroup M_i from θ for viable schedules. As more VGs are introduced it is likely that $(\sum_{x=1}^i M_x) \supseteq \gamma$ meaning that all optimal solutions are removed from θ . From this point on, the reduction of control-pins becomes a trade-off for execution time. As γ was removed from θ by $\sum_{x=1}^i M_x$, the new solution space θ_n has a new subgroup of optimal solutions γ_n which in turn can be removed by VGs that remove subgroups M_j for $(\sum_{x=1}^j M_x) \supseteq \gamma_n$. In this chapter we will explain in detail how such VGs can be found and what the effects of this trade-off on the schedule and execution time of applications are. In Fig. 4.1 we can see where this optimization affects our design process. The changes are fairly substantial as significant changes are made to the physical design as well as to the logical control of the valves.

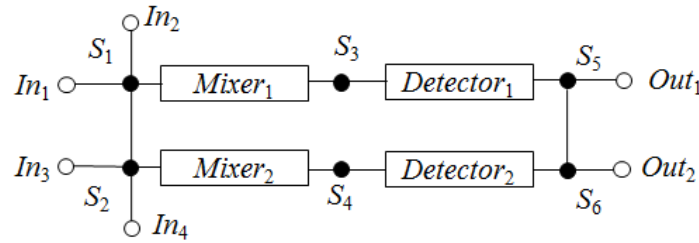


Figure 4.2: IVD Architecture

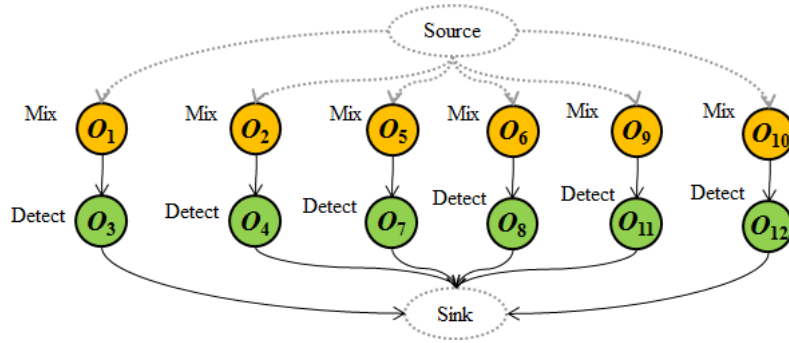
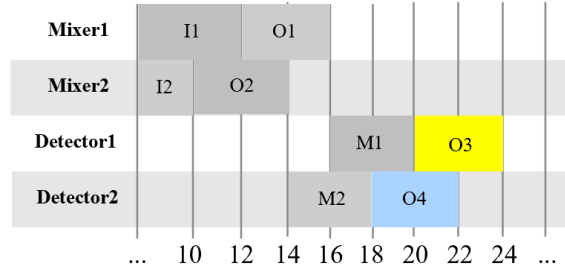


Figure 4.3: Typical IVD application that mixes various samples, reagents and buffers and analyses the results

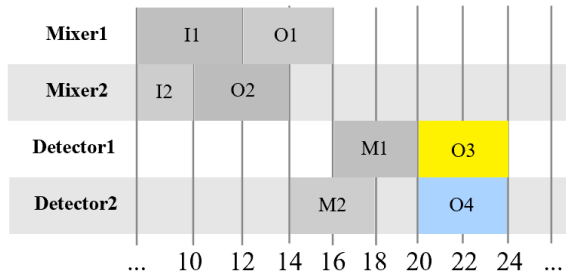
4.2 Problem Formulation

Let us illustrate our problem using the In-Vitro Diagnostics application (IVD) from Fig. 4.3 that has to be mapped to the architecture from Fig. 4.2. We indicate with O_i the i -th operation in the IVD. Let us assume that the binding of operations is as follows: O_1 , O_2 and O_5 are bound

to Mixer₁; O₆, O₉ and O₁₀ are bound to Mixer₂; O₃, O₄ and O₇ are bound to Detector₁ and O₈, O₁₁ and O₁₂ are bound to Detector₂. Furthermore, let us assume that we schedule this application such that the resulting schedule is the one shown in Fig. 4.4a leading to the valve actuation sequence shown in Table 4.1. The schedule shows the start times and the duration of operations as references on a timeline, and the four rows correspond to the FFUs in the architecture from Fig. 4.2. We denote with M_i the fluid transport operations. The time units are listed at the bottom, indicating the elapsed time since the start of the application. Note that for space reasons we only show partial schedules in Fig. 4.4 and a partial valve actuation table in Table 4.1. The rows of the table represent the valves and each column represents a time step in which certain valves need to be actuated. As mentioned, related work [78] performs control pin minimization after scheduling, thus using the data from Table 4.1 as a starting point. There, Valve₁ and Valve₃ for example, can be in the same state throughout the depicted time steps (at time step 18, Valve₃ is in a *don't care* state and can be opened to suit Valve₁) and their controls can therefore be combined.



(a) Partial IVD schedule before the pin-count reduction, all operations are executed as early as possible.



(b) Partial IVD schedule after pin-count reduction with scheduling constraints in place, leading to a deferred execution of O₄

Figure 4.4: Impact of pin-count reduction on IVD schedule

Note that in the IVD schedule in Fig. 4.4a, the detection operations O₃ and O₄ are waiting for the mixing operations O₁ and O₂ to finish respectively. The fluid transport M₂ and the follow-

ing detector operation O_4 are started as soon as possible, in order to minimize the application completion time. This however means, that the equivalent operations for Detector₁ (M_1 , O_1) are not executed at the same time as for Detector₂. This leads to the valves used by these four operations to be in opposing states in time step 18 as highlighted with the corresponding color coding in Table 4.1 and Fig. 4.4a, meaning their controls can't be combined.

Valve No. / TS	8	10	12	14	16	18	20	22
1	0	0	0	0	0	0	X	X
2	1	1	0	0	1	1	X	X
3	0	0	0	0	0	X	X	X
4	1	0	0	1	1	X	X	X
5	0	0	1	1	0	0	X	X
6	0	1	1	0	0	X	X	X
7	0	0	X	X	0	0	1	1
8	0	X	X	0	0	1	1	X
...
8A	0	X	X	0	0	X	1	1

Table 4.1: Partial actuation Table for IVD. 0: Open. 1: Closed. X: Don't Care

As we can see from this example, the scheduling step has introduced constraints that restrict the valve sharing options. Let us assume that we have decided to combine the valves used by Detector₁ and Detector₂ before scheduling the operations. This means that the two detectors can only operate simultaneously, which creates a scheduling constraint. We can enforce this constraint during scheduling, for example by postponing the execution of operation O_4 until operation O_3 can be executed as well, the resulting schedule is shown in Fig. 4.4b. The resulting change in the control logic for V_8 is stated as V_{8A} in Table 4.1, which is now compatible with V_7 . However, the introduced change in the schedule can affect the application execution time, as Detector₂ is now occupied by O_4 until time step 24 instead of time step 22, potentially postponing other operations by 2 time steps as well.

By determining the valve sharing before the scheduling step, we can reduce the number of control pins required. For the complete IVD application, which has an initial pin-count of 31 we can reduce the pin-count from 14 using our proposed technique, compared to a pin-count of 20 as determined by the technique from [78]. The scheduling constraints introduced by our technique do however delay the operations, increase the application completion time and possibly exceed the application deadline, which has to be satisfied for certain

biochemical protocols. For this example the reduction of required control pins to 14 increases the application execution time from 81 (as determined before the reduction) to 107 time steps. In addition, we have to make sure that the application can still be executed successfully as control pin combinations can force valves to be in states which are incompatible with certain routes, which is explained in detail in Sect. 4.3.1.

We define the problem we address in this chapter as follows: Given an architecture and application model, optionally including application deadline and available number of control-pins, we want to determine the optimal number of VGs to form. The optimal number of VGs is determined by the input. (1) If any deadline is given, the optimal number of VGs is the smallest number of VGs that still allows the application to finish within the given time, despite the scheduling constraints introduced by the VGs. (2) If an available number of control-pins is given, the optimal number of VGs is the given number of control-pins. (3) If neither deadline or available number of control-pins is given the optimal number of VGs is the smallest number of VGs that still allow for the application to successfully execute, regardless of the required time.

4.3 Proposed Method

We have indicated in Fig. 4.1 which parts of the design process we extend with our pin-count reduction technique. The underlying steps to solve the problem outlined in the previous section are also shown in Fig. 4.5.

In step 1, we bind and route the operations given by the application onto to the given architecture model, using an alternate routing technique presented in Sect. 4.3.1 which is based on the basic routing explained in Sect. 3.1. Step 2 performs a preliminary control synthesis which determines the valve states for every created route. Contrary to the complete control synthesis [78], this version does not yet contain any information about timing. The determined valve states are then used as input along with the application deadline for our proposed pin-count reduction algorithm from Sect. 4.3.1. The algorithm produces a grouping of valves that can share the same control pin, which introduces scheduling constraints for the next step. Since scheduling the whole application is time consuming, step 4 uses a prediction function presented in Sect. 4.3.1 to determine the impact of the constraints introduced on

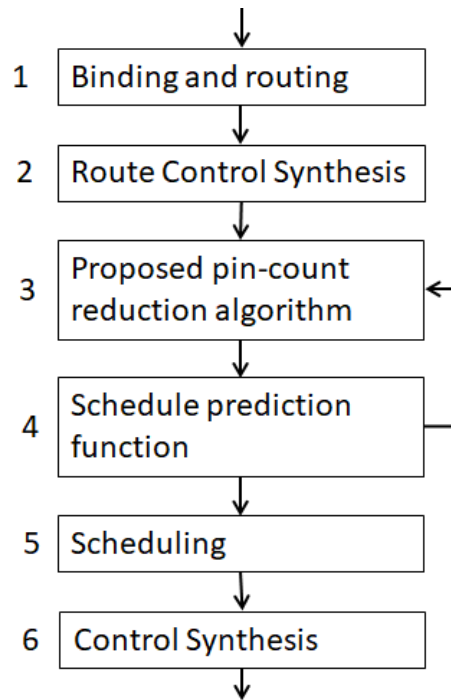


Figure 4.5: Design steps used by the proposed method

the schedule. The application is scheduled in step 5. Step 6 concludes this process with control synthesis, which in contrast to the one performed in step 2, does also incorporate the schedule. To schedule the application we have implemented a List-Scheduling algorithm [73] and extended it to take scheduling constraints into account. After the control synthesis is performed, we know how many control pins are required and how they have to be connected to valves. The physical design task for the control layer is responsible to determine the routing of the control channels in the architecture [47]. Note that our approach is iterative (the back arrow from step 4 to step 3) and allows the designer to control the trade-off between the number of control pins and the application completion time. If the end-user already has a control box with a given number of outputs, then this number can be given as an input, and our algorithm in step 3 will stop when this is satisfied. Our algorithm also stops before the application deadline (if given) would be exceeded. If none of these restrictions are given, a solution using the smallest number of control-pins is determined.

4.3.1 Reducing the Pin-Count

As shown in Fig. 4.5, binding and routing of operations has to be completed before our pin-count reduction technique can be applied. We use a routing algorithm based on the fundamental routing explained in Sect. 3.1. There we mentioned that, depending on the architecture, it is possible that multiple paths connecting the source and target FFU exist or multiple components of the same type to which an operation can be bound (e.g. Heater, Detector) are available, meaning multiple options for all three routing steps are available. For convenience, we use the same basic architecture from Fig. 2.11 in the upcoming Figs. 4.7 and 4.8. However, for optimal results of our pin-count reduction, we no longer necessarily choose the shortest FP over longer FPs if presented with multiple options. Our strategy is to choose the FP which blocks the smallest number of FFUs and switches on the chip, which is not necessarily the shortest FP. A component is considered blocked if it is actively used (e.g. routed through), or if it cannot be used by another FP, since an adjacent component is used. Take FPS_4 in Table 2.1 (In_2 , S_2 , S_1 , $Heater_1$) for example. Even though it does not use $Filter_1$, the valve leading towards it has to be closed and the filter is considered blocked, since it is not possible to form a FP that uses $Filter_1$, while FPS_4 is active. Choosing routes with the lowest number of blocked components therefore allows a higher flexibility while scheduling, since more components are still available to be used by another FP.

To operate our example architecture from Fig. 2.11, 32 valves are required. A partial enumerated example of where those valves are located in this architecture is shown in Fig. 4.6.

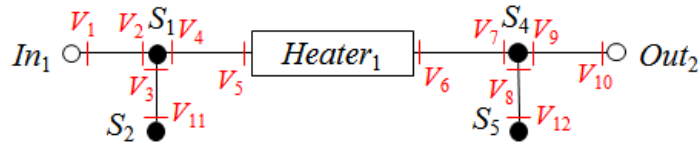


Figure 4.6: Valves

Our goal is to form VGs from these valves which incorporate as many individual valves as possible, which as a result are actuated synchronously from a single control-pin. We distinguish between two distinct sets of valve combinations into VGs, according to their effect on the schedule.

The first set of combinations performed differs from other combinations, as they will never introduce any constraints onto the schedule. Those combinations can be made, regardless of the architecture, for all valves which are located in a channel between two switches, an input

and a switch, or an output and a switch. Consider In_1 and S_1 in Fig. 4.6. They are connected by a channel which contains $Valve_1$ and $Valve_2$. Only two possibilities exist in which this channel can be used: Either the channel is routed through, in which case both valves have to be opened, or it is not routed through, in which case both valves can be closed. Therefore, the valves have no need to be actuated individually. The same principle applies to the valves which close off any given FFU, along with other valves that are located in this channel (e.g. valves 4 to 7 in Fig. 4.6). Therefore, these valves' controls can be combined and share a control pin, meaning that these valves will always work in unison as a VG. Fig. 4.7 shows the result of applying this to the presented architecture, where the previously mentioned $Valve_1$ and $Valve_2$ now form VG_1 , valves 4 to 7 form VG_2 and so on. Further combination of VGs means that all valves of both groups share a single control pin and are therefore activated in unison.

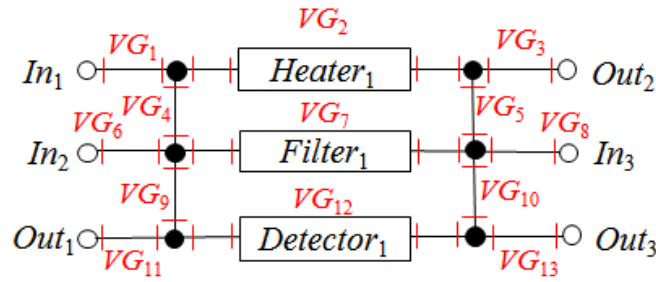


Figure 4.7: Valve Groups

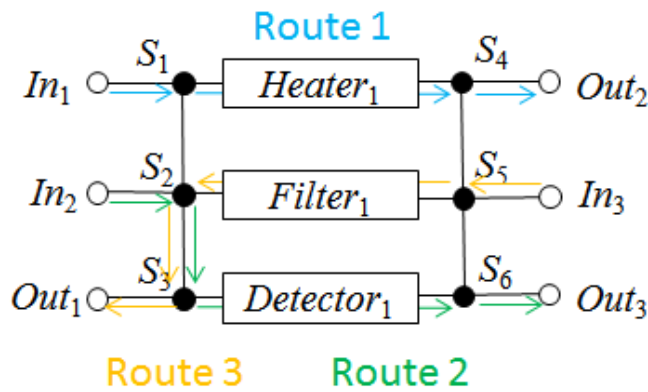


Figure 4.8: Routes

Additional combinations will introduce scheduling constraints, which may affect the application execution time negatively. To ensure that the application deadline will not be exceeded, it is therefore necessary to determine combinations that introduce constraints with minimal effect on the schedule. To clarify the effects of such constraints we map the application from

Fig. 2.12 to our example architecture in Fig. 2.11. The three resulting routes can be seen in Fig. 4.8. From these routes we can extract the valve states for each route, which is illustrated in Table 4.2. Each route has sets of VGs in the previously explained states *open*, *closed* and *don't care*.

Table 4.2: Valve Group Configuration

Route	Open VGs	Closed VGs	Don't Care VGs
1	1, 2, 3	4, 5	6, 7, 8, 9, 10, 11, 12, 13
2	6, 9, 12, 13	4, 7, 10, 11	1, 2, 3, 5, 8
3	7, 8, 9, 11	4, 5, 6, 10, 12	1, 2, 3, 13

Given this data, the application graph, the desired number of control pins and (if relevant) the maximum application execution time, Algorithm 1 can determine the potential effect on the schedule for each combination of VGs and hence choose the appropriate combinations that have the minimum impact on the application execution time.

Algorithm 1 Pin-count reduction

```

1: for each possible combination of VGs comb do
2:   if valves in comb have to be in opposite states for at least one route then
3:     Combination not possible without invalidating such routes
4:   else if all valves in comb are in the same state at any given time then
5:     Combination valid and no increase in schedule length
6:   else
7:     for all possible pairings of given routes do
8:       if combining the valves in comb can increase the schedule length then
9:         Use prediction function to determine by how much
10:      end if
11:      Store the prediction for this pair of routes, or 0 if no prediction was necessary
12:    end for
13:  end if
14: end for
15: Sort all combs according to the predicted increase in execution time
16: Combine VGs according to the sorted list until the application deadline is exceeded

```

Algorithm 1 starts with the assumption that all VGs can be combined, hence all permutations of combinations are iterated through in line 1. However, several combinations can be discarded right away as they would invalidate at least one of the given routes, meaning the application could no longer be correctly executed. This is the case whenever some of the valves in questions are in the *open* state, while another part is in the *closed* state for a single route, as shown in Example 1 in Table 4.3 and checked in line 2 in the algorithm.

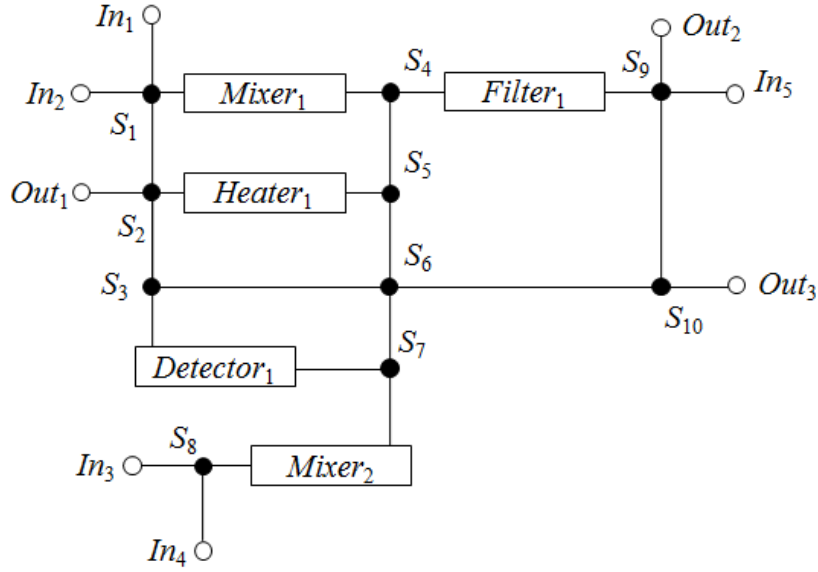


Figure 4.9: Multi-purpose architecture

If this is not the case, the algorithm continues in line 4, determining whether this combination introduces a scheduling constraint. This is similar to how valve groups are determined in the related work in [78]. However, the related work only determines whether the combination is valid for an already given schedule. We determine if the combination is valid for **any** possible schedule, meaning no scheduling constraint is introduced. This is the case if all potentially combined valves can be in the same state for all routes, i.e. all valves are either in the *open* or *don't care* state, or in the *closed* or *don't care* state, for all routes, as it is the case in Example 2. If a combination of valves does introduce a scheduling constraint, the algorithm continues on to determine the impact on the application execution time for this constraint in lines 7-12. A scheduling constraint prohibits two or more routes to be executed in parallel, because parallel execution would cause unintentional mixing of fluids, leading to a potential increase in application execution time as these routes have to be executed in sequence. Hence we iterate through all pairs of routes in line 7 in order to determine the effect on them. First, some routes were never able to be executed in parallel anyway, since they partially overlap, as demonstrated in Example 3. In this case, which is checked in line 8, the scheduling constraint does not have any effect on this pair of routes. In other cases such as in Example 4, the scheduling constraint does impact the parallelism of the application as routes 1 and 3 can no longer be executed at the same time. When such a case arises, the algorithm determines how much this scheduling constraint affects the application execution time in line 9. Exact results can however only be determined by scheduling the application and comparing the

execution times. Since this is too time consuming, we propose a cost function to predict the effect on the schedule. This prediction is based on how many parallel executions of routes are prohibited by a scheduling constraint. E.g., Example 5 prohibits route 1 from being executed at the same time as either route 2 or 3. Example 4 on the other hand only prohibits parallel execution of routes 1 and 3, leaving the possibility to execute routes 1 and 2 at the same time, leading to a prediction of a smaller increase in execution time than for Example 5. Additionally the execution time of each route (how long it takes to transport the fluid along the FP) is taken into account. The cost function predicts that constraints affecting routes with long execution times have a larger impact on the schedule. This can be seen in Examples 6 and 7 where we assume that routes 1-3 have an execution time of 10, 8 and 6 time steps respectively. Both examples prohibit one pair of routes from parallel execution, yet Example 7 results in a shorter execution time, since the longest routes are not affected by constraints.

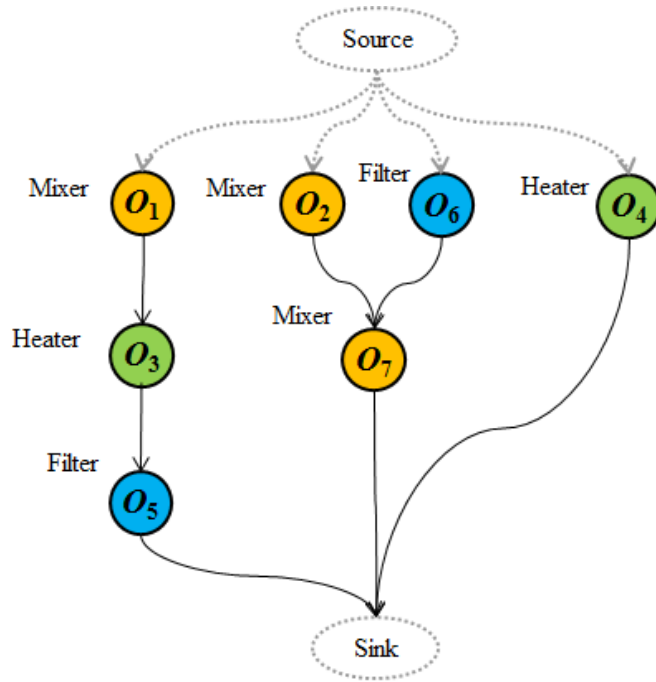


Figure 4.10: Multi-purpose architecture application

Once the impact on the application execution time has been predicted for all combinations of valves, these combinations are sorted from smallest to largest impact in line 15 and then applied to the architecture until a stopping criteria is reached. The end user can specify the stopping criteria: Stop the pin-count reduction when a given application deadline is reached, or stop when a given target number of control pins is reached. By varying the stopping

criteria, an end-user can trade-off the number of control pins and the application completion times, as discussed in the experimental results.

Example	Combined VGs	Result	Reason
1	VG ₁ and VG ₄	Combination not valid	Route 1 is invalidated, since VG ₁ has to be open to allow fluid transport while VG ₄ has to be closed to prevent leakage
2	VG ₁ and VG ₂	Combination valid and no increase in schedule length	VG ₁ and VG ₂ are <i>open</i> for Route 1 and in a <i>don't care</i> state for Routes 2 and 3. Therefore no matter which routes are executed in parallel, VG ₁ and VG ₂ can always be open
3	VG ₆ and VG ₁₃	Combination valid and no increase in schedule length	Even though this combination does prohibit routes 2 and 3 from being executed in parallel, no increase in schedule length will occur since these routes were not able to run in parallel in the first place, due to their partially overlapping FPs
4	VG ₁ and VG ₁₂	Combination valid and possible increase in schedule length	VG ₁ has to be open for Route 1, while VG ₁₂ has to be closed for Route 3, prohibiting parallel execution of these routes
5	VG ₁ and VG ₁₀	Combination valid and possible increase in schedule length	VG ₁ has to be open for Route 1, while VG ₁₀ has to be closed for routes 2 and 3, prohibiting parallel execution of these routes
Example	Constraint	Result	Exec. time
6	Routes 1 and 2 restricted from being executed at the same time	Routes 1 and 3 are executed in parallel Route 2 is executed as soon as Route 1 finishes	18
7	Routes 1 and 3 restricted from being executed at the same time	Routes 1 and 2 are executed in parallel Route 3 is executed as soon as Route 1 finishes	16

Table 4.3: Valve Group combination examples: Examples 1-5 show the outcome of combining certain VGs regarding the architecture and routes from Fig. 4.8. Examples 6 and 7 show how different constraints can have varying effects on the schedule length. For these examples, 10, 8 and 6 time units are assumed for routes 1, 2 and 3 respectively.

4.4 Experimental Results

We were interested to evaluate the proposed Pin-Count Minimization (PCM) strategy. The evaluation has been performed for 4 biochips with the corresponding biochemical application. We have compared our PCM approach with the Control Synthesis Optimization (CSO) from [78], which performs pin-count reduction after the scheduling step, using a Graph-Coloring algorithm. We have used the following test cases for the evaluation:

Test Case 1 (TC1): We use an architecture capable of IVD (In-Vitro Diagnostics), which has various real life applications [25].

TC2: We have used the processing part of the MOA (Mars Organic Analyser) [84], and for TC3 we have modified it to an alternate design for the same application.

TC4: We have created a Multi-Purpose (MP) architecture to test the effects of our pin-count reduction technique. The corresponding architecture- and application model for our MP-Architecture can be seen in Figs. 4.9 and 4.10. The number of FFUs and valves in the architectures as well as the number of operations in the applications are given in columns 2-4 in Table 5.2. The architecture and application models are available as supplemental materials.

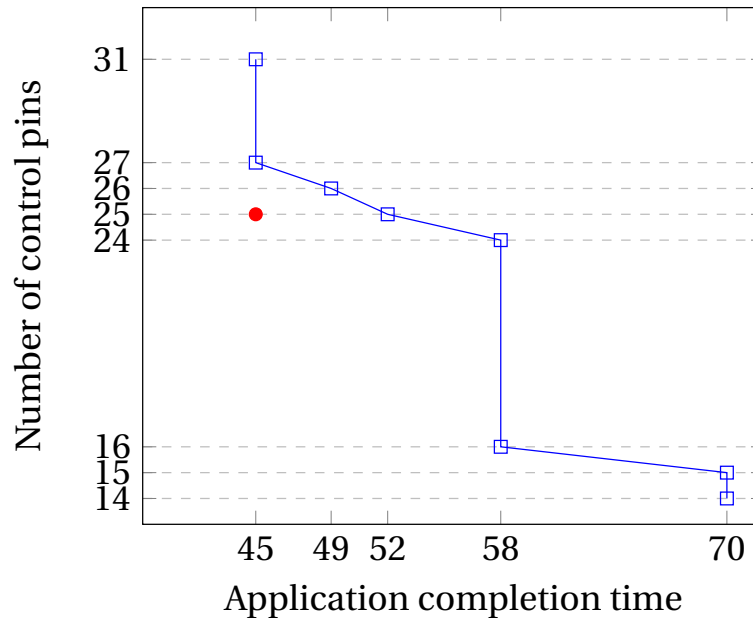


Figure 4.11: Trade-off for TC4 using PCM (blue, squares and line). CSO result reference for TC4 (red dot)

For all experiments we assume that it takes 1 time step for fluid to pass through any given channel (e.g. from one switch to another) and 4 time steps for an operation (e.g. mixing) to finish. Table 5.2 shows a direct comparison between the CSO presented in [78] and our PCM. The table contains the number of valves and FFUs in the architectures, the number of controls required and the corresponding execution times for both methods, for each of the tested architectures. As expected, the pin-count reduction comes at the expense of an increase in application completion time. However, for most cases, the minimum number of pins is obtained without a significant increase in completion time. The advantage of our approach is that it allows for a direct, stepwise trade-off between the number of control pins and the application completion time.

Test Case			PCM		CSO [78]	
Name	FFUs	Valves	Pin count	Completion time	Pin count	Completion time
TC1	8	46	14 (30%)	107 (32%)	20	81
TC2	13	84	17 (14%)	119 (0%)	21	119
TC3	15	74	14 (30%)	96 (13%)	20	85
TC4	13	66	14 (44%)	70 (56%)	25	45

Table 4.4: Comparison between PCM and CSO. The percentage values in PCM indicate the reduction of the pin count and the increase in completion time compared to CSO respectively.

Hence, in our second experiment, we were interested to evaluate the ability of our proposed PCM approach to support such a trade-off. Thus using TC4 (TC1-3 are available in the supplemental material) we show in Fig. 4.11 how PCM enables the trade-off between the number of required control pins and the application completion time. The figure shows the number of control pins, starting at 31, which is the initial number of VGs created, on the Y-axis and the completion time on the X-axis. As we can see from the figure, the end-user can choose which trade-off is most appropriate, considering the constraints of their control solution in terms of number of control pins available versus the application completion time, which may be constrained by a deadline for certain applications. Such a trade-off is non-trivial, since the impact of a certain valve-sharing solution on the schedule is not easy to determine. For example, there are combinations which do not affect the schedule length, for reasons as shown in Examples 2 and 3 in Table 4.3. This is however only true for this particular order in which the combinations have been made. E.g., the combinations that reduced the control count from 24 to 16 do not generally have no effect on the schedule length. Instead, the scheduling constraints introduced by the combinations before, create a scenario similar to Example 3 for the following combinations. Especially for applications

with few operations, such large jumps can occur frequently after multiple constraints have already been introduced. This is due to the fact that most of these few operations are already executed in sequence after a small number of constraints is applied, providing more options for combinations such as Example 3 in Table 4.3, which constrain the schedule no further. Additional combinations bring the control pin count to 14, which is the minimum number of control pins required to run the application as determined by the pin-count reduction algorithm. We have also validated our method on the AquaFlux biochip controlled by a 36-controls box from Microfluidic Innovations, LLC. Our method was able to determine the same pin-count as the one currently used by the manually designed AquaFlux biochip.

4.4.1 Actuation of multiple valves

We have verified the possibility to actuate multiple valves using a single pressure source with prototypes. As shown in Fig. 4.12, we have built mixers with several valves that share pressure sources, as suggested in this chapter. Additional tests such as shown in the video named “SharedValves” in our video repository [105] further prove the functionality of shared valves. In this video, four valves which are positioned in different locations on the biochip and therefore varying distance to the control-pin, are actuated from a single pressure source. However, the number of valves connected to a pressure source directly impacts the required pressure to actuate them. Additionally, large distances and therefore long control channels between valves sharing a pressure source do not only further increase the required pressure, but can also introduce a latency in valve actuation. We use the architecture in our video as example: Providing -5 kPa, which has been sufficient to actuate a single valve in other chips, has no effect on the four connected valves. An increased pressure of -10 kPa will actuate the two valves close to the control-pin, but not the other two. Further increased pressure of -15 kPa will open all four valves, however, while the close-by valves actuate instantly, a noticeable latency (of about 1 second) occurs before the other two, further away valves actuate. Applying even more pressure of -20 kPa as in the video will open all four valves without any noticeable delay.

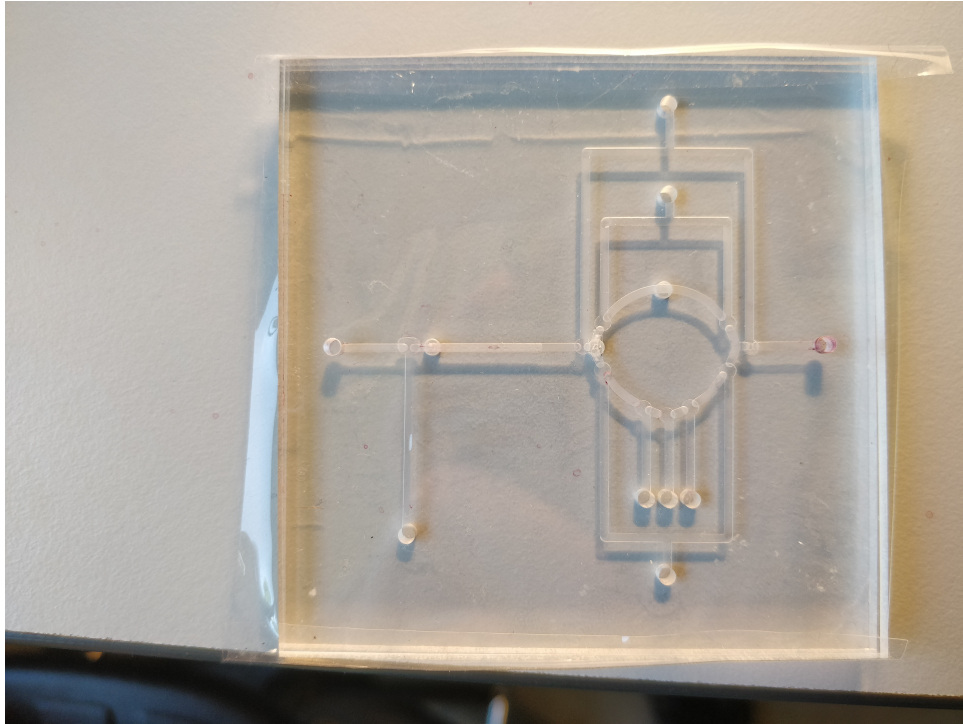


Figure 4.12: Mixer with three pairs of valves using shared pressure sources

4.5 Conclusions

With new advances in biochip fabrication, the number of valves integrated on a single chip is rising fast. The external hardware required to operate a biochip is however not as scalable, resulting in further increase in size and cost of this hardware, which is already magnitudes larger and more expensive than a biochip. In this chapter we have proposed a new technique to reduce the required pin-count for flow-based biochips, effectively reducing the complexity of external hardware required. Contrary to previous work, this method is able to trade off execution time to reduce the pin-count even further. Experimental results have shown that our algorithm is capable of reducing the pin-count significantly, while keeping the extension of the schedule length acceptable. To produce realistic results, the current state-of-the-art routing model has been extended.

Chapter 5

Waste-Aware Volume Management

This chapter is based on the contribution P3 as described in Sect. 1.4.

5.1 Introduction

The problem of fluid management is often ignored for microfluidic applications, assuming that all fluidic constraints are satisfied during the execution of an application [3]. However, dispensing more fluid than needed (overflow) can be expensive, while shortages of fluid (underflow) can interrupt the execution and even require manual replenishment. Furthermore, hardware restrictions can impose constraints on the amount of fluid that can be passed on from one operation to another, details of which are presented in Sect. 2.4. Additionally, non-deterministic events such as errors [62] or conditional execution [34] can further complicate fluid management. Our experimental results show how wasteful microfluidic operations are in many cases. While this issue is often masked by the fact that extremely small volumes can be used in biochips, it becomes very apparent when the output volume is predetermined, for example for master-mixes or as minimum volume for certain detection methods.

Research so far has provided multiple solutions for optimizing mixing operations, both for droplet and flow based biochips [23, 34, 59]. These techniques however are focused on optimizing mixing operations to require as few mixing steps as possible, which can increase fluid consumption and require high computational complexity. The waste-aware mixing

algorithm proposed in [112] allows to minimize the consumption of one input fluid, but is restricted to a mix of only two fluids and drastically increases the consumption of the other fluid. Furthermore, [112] focuses on optimizing single mixing operations as opposed to the application-wide fluid management problem we address in this paper. Previous research on automatic volume management has proposed a linear programming solution, which is capable of avoiding over- and underflow for systems with arbitrary mixing ratio hardware. However, the execution time is infeasible for large assays, hence a fast heuristic is proposed for an over-constrained version of the problem, which in turn does not guarantee to provide a valid solution [5]. Furthermore, the authors are interested in a solution that maximizes the sum of the output volumes and ignore the waste produced during the execution. We consider that the output volumes are given and we are interested in providing solutions that satisfy all constraints while minimizing the total fluid volume required. Wasted fluids are to be reused whenever possible to further decrease the fluid consumption. Furthermore, we are interested to design a low computational complexity algorithm, to allow for non-deterministic events such as conditional execution or errors to be solved during the execution of the biochemical application.

Fig. 5.1 shows that the proposed application graph optimizations are applied at an early stage of the design process during the application graph generation. This allows for significant changes to be made, without the need to reiterate through any other parts of the design process.

To apply our volume management technique to our design process, we extend some models introduced before, as well as introduce additional constraints that are present in any given FBMB. We extend our architecture model from Sect. 2.4 by a Maximum Hardware Capacity (MHC) parameter for every FFU as shown in Fig. 5.2. The FFUs might also be able to function with a smaller volume, but if a volume larger than the MHC is transported to an FFU, part of the fluid will remain outside the FFU, potentially blocking channels and other FFUs. Furthermore we introduce a Minimum Volume Requirement (MVR) which is the smallest volume of fluid a component requires to function properly, for example a detector requires a certain volume of fluid to allow for a reliable optical detection. An additional, architecture wide restriction on fluid volumes is the Hardware Transport Resolution (HTR). This describes the smallest volume of fluid that can be handled reliably by the architecture. For many architectures this describes the smallest volume necessary to fill the channel depth and width

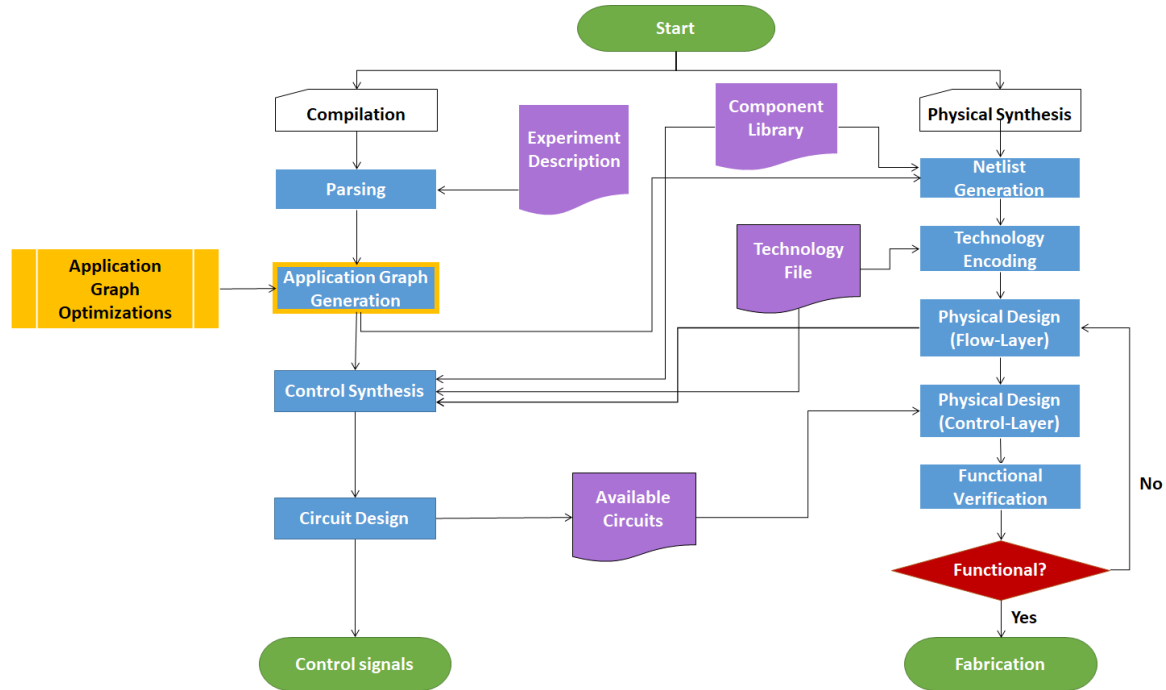


Figure 5.1: The proposed volume management optimization is implemented in the highlighted design step application graph generation

for a sufficient length to avoid faults, occurring due to air bypassing fluid in a channel due to its small volume. The HTR can however also be imposed by functional components, for example the smallest volume that can be accurately metered.

Fig. 5.3 shows an example using a metering chamber, used to obtain a certain volume of fluid from the input reservoir. While it is possible to fill the metering chamber multiple times and combine the fluid to create larger volumes, it is not possible to determine whether the chamber is filled to a certain degree, making it impossible to reliably meter volumes smaller than the size of the metering chamber. To assure correct functionality of the chip, all volumes

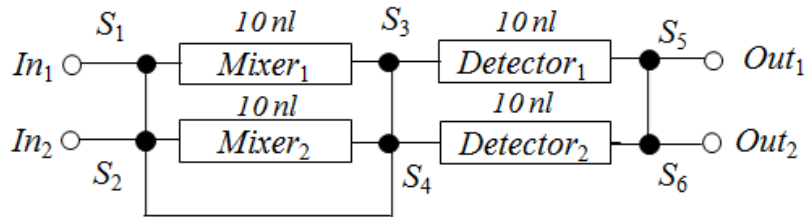


Figure 5.2: Architecture netlist including two mixers and detectors with a maximum hardware capacity (MHC) of 10 nl each.

used therefore have to be multiples of the HTR.

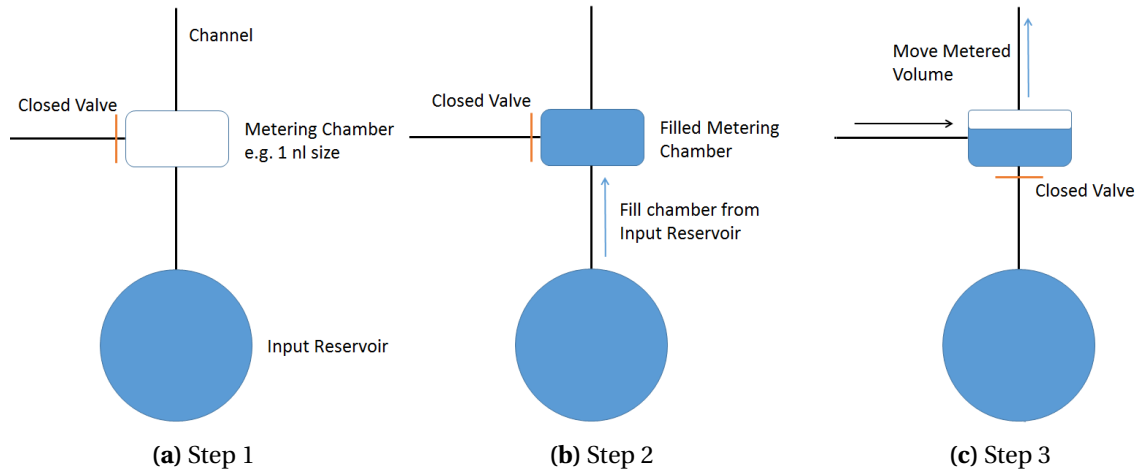


Figure 5.3: Metering example using a metering chamber

We extend the use of the application model introduced in Sect. 2.5 by adding discrete and fractional volumes to the edges, indicating the minimal amount or ratio of fluids that have to be transported to the following operation. An example of such an application is shown in Fig. 5.4, where red operations indicate mixing operations which do not require a specific volume of fluid (regarding the application, architecture restrictions still apply), but rather a certain ratio of multiple fluids. Green operations indicate other operations such as heating, detection or filter, which have a MVR.

5.2 Problem Formulation

Given the application from Fig. 5.4 as input, we are interested in determining the **Fluid Volume Assignment (FVA)** for each operation of the application such that all the fluidic constraints are satisfied and the total fluid volume is minimized. This is achieved by advanced mixing algorithms and reuse of waste, generated by the operations. In addition to the application graph, we use the component library and technology file as inputs as shown in Fig. 5.1. These provide information about the available hardware (e.g. 1:1 or arbitrary ratio mixers) as well as the MHC and MVRs of the component and the HTR of the architecture. The architecture from Fig. 5.2 is not required as an input, but only used to clarify the process.

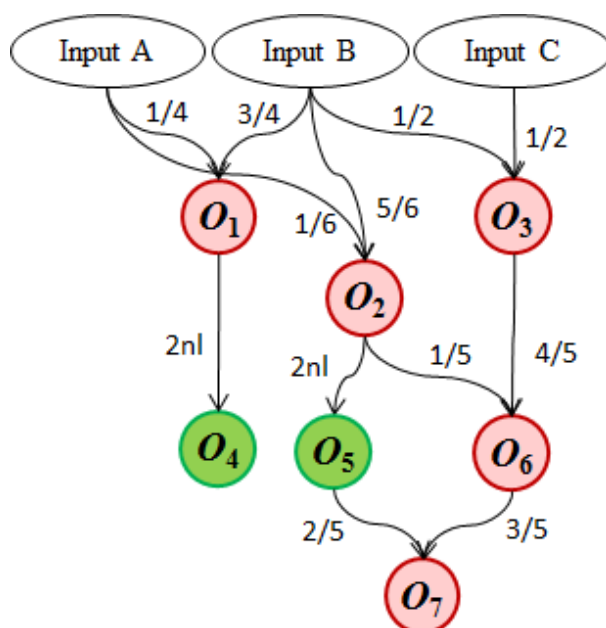


Figure 5.4: Application graph showing the input requirements of the operations as ratios for mixing operations (red) and as discrete values for the other type of operations (green).

5.3 Mixing

For many biochemical applications the most common operation is mixing, be it in preparation of an experiment to produce a Master-Mix, preparation for detection requiring to add dye to the samples or long term experiments requiring additional reagents at various times. At the same time, mixing operations are the main source of fluid waste in continuous-flow biochips as mixing operations are prone to produce excess fluid.

The 1:1 mixing architecture as shown in Fig. 2.7a mixes two equal volumes of fluid. To achieve ratios other than 1:1, multiple sequential mixing steps are necessary. Fig. 5.5 shows how a 1:3 mixing ratio can be achieved by adding a second mixing operation which mixes the result of the first operation with additional fluid from the input. However, the architecture of 1:1 mixers makes sequential mixing such as this especially wasteful. The output volume of a mixing operation is the sum of both input volumes, therefore if O_1 receives 1 nl from each input, it produces an output volume of 2 nl. Assuming the same size of mixer for all operations, only 1 nl can be passed on to O_2 to be mixed with more fluid from the input, leaving the other 1 nl to be discarded. Using mixers of increasing sizes is infeasible. While using a mixer that takes two inputs of 2 nl for O_2 would allow O_1 to pass on all its output volume, it would

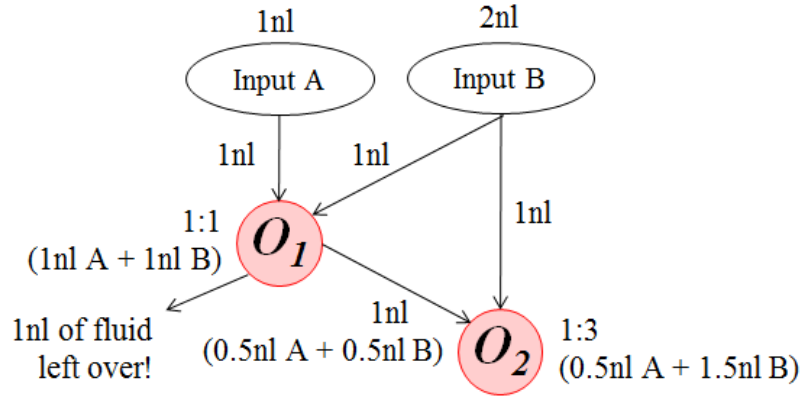


Figure 5.5: Example of achieving a 1:3 mixing ratio using 1:1 mixing hardware and the unavoidable byproduct of fluid (i.e. waste) not in the target ratio.

also require Input B to dispense 2 nl instead of 1 nl to this operation, increasing the overall fluid consumption and furthermore just passing on the problem to a potential next mixing operation which would then require an even larger mixer.

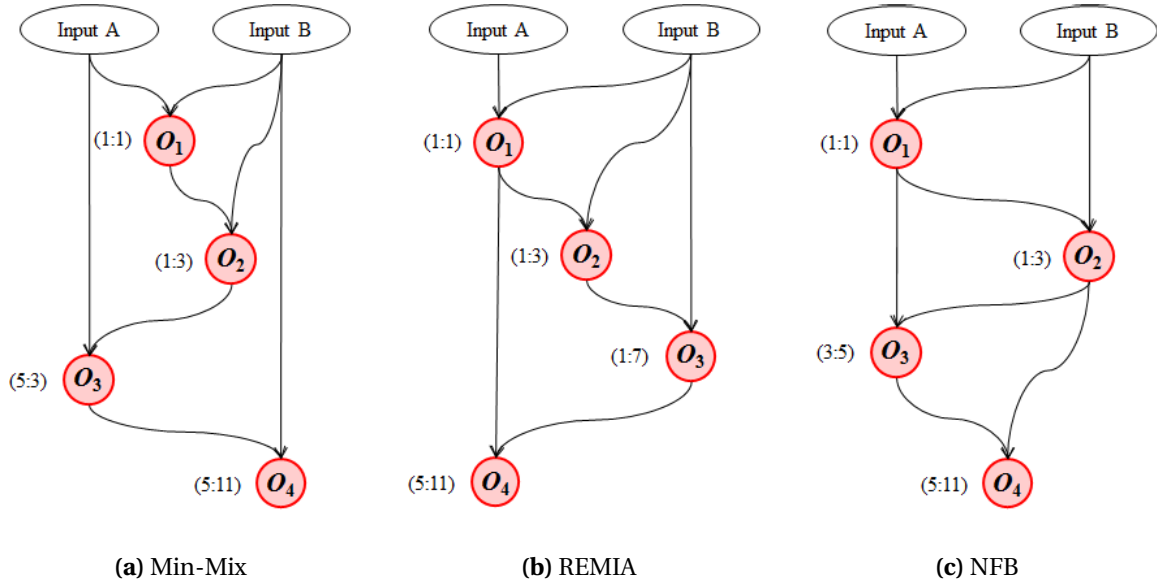


Figure 5.6: Comparison of mixing trees for a target ratio of 5:11 created by Min-Mix, REMIA and NFB

Furthermore, not all mixing ratios are obtainable by chaining multiple 1:1 mixing operations together. Ratios which are unreachable using 1:1 mixing hardware are approximated and the quality of the approximation can be controlled as it is directly linked to the number of mixing operations performed, also called precision level [23]. As mentioned before, sequential mixing is wasteful and should therefore be done as efficiently as possible. Multiple solutions

have been proposed to find optimized mixing trees, such as Min-Mix (MM) [111], REMIA [59] and the Network Flow Based (NFB) algorithm [23]. We have compared and evaluated each of the techniques and a basic example outcome for each can be seen in Fig. 5.6 which also provides a direct comparison. While MM and NFB focus on minimizing the operation count, REMIA also considers the difference in cost of the inputs (e.g. sample and buffer) and minimizes the use of the expensive fluid. As Fig. 5.6 already indicates, NFB performs superior regarding fluid consumption compared to MM and REMIA, as it solves an Integer Linear Programming (ILP) problem optimally, which can be visualized with network graphs such as shown in Fig. 5.7. Solving an ILP program does however require significantly more computation time than MM or REMIA. This can be tolerated to some extent, since the application graph optimization is generated during the design phase as shown in Fig. 5.1, but large applications or very high approximation precision levels can still lead to infeasible calculation times.

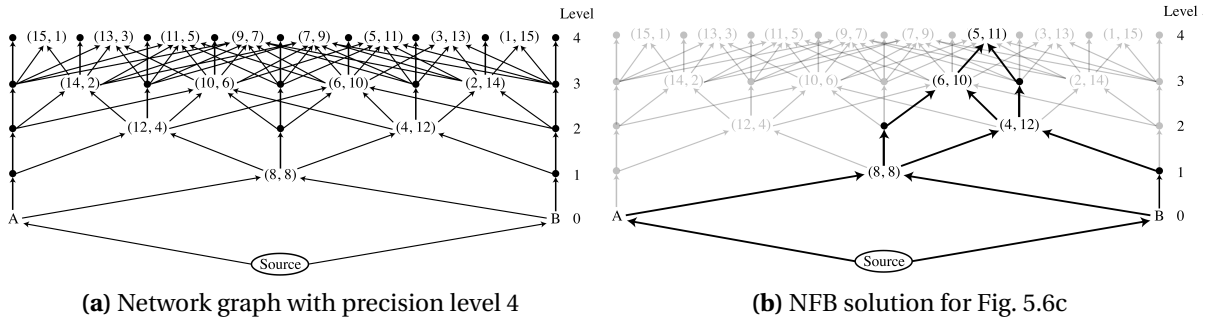


Figure 5.7: Comparison of mixing trees for a target ratio of 5:11 created by Min-Mix, REMIA and NFB

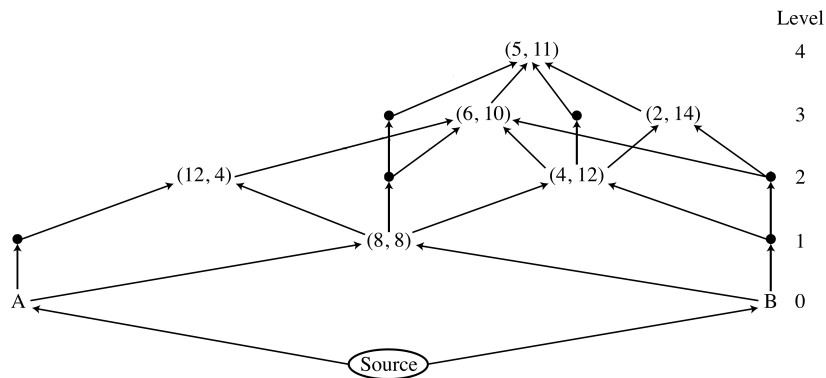


Figure 5.8: Application from Fig. 5.4 with fluid volume according to each operations MVR assigned, leading to shortage of fluid at O_6 .

We therefore apply a pruning technique to the Network Graph, which removes a significant amount of vertices which are unlikely to be in the optimal solution. The resulting Network Graph is shown in Fig. 5.8. The applied restriction is that no more than four vertices at any level X can be used as input for any vertex at level $X + 1$. While this alteration no longer guarantees an optimal outcome, the results are still significantly superior to both MM and REMIA, and keeps the execution time feasible even for high precision levels as discussed in Sect. 5.5.

The computational complexity of NFB does therefore not pose an issue and we consider that the NFB4 (NFB pruned to 4 vertices) version of NFB provides the best solution to the mixing problem for our case and the mixing trees of our following examples have been created using this technique.

An alternative to 1:1 mixing architecture are arbitrary ratio mixers as proposed in [4]. Such mixers can drastically reduce the amount of excess fluid created during sequential mixing operations since most ratios can be mixed in a single operation. This is possible because the mixer does not require to be completely filled with fluid, but can be partially filled with air. Air which is picked up by the fluid during the mixing process can then be expelled through vents [31]. A ratio of 1:3 as shown in Fig. 5.5 can be mixed in a single operation as any volume of fluid (up to the mixers maximum capacity) can be dispensed into the mixer. While this eliminates the wasteful in-between step which is necessary when using 1:1 mixers, the outcome is not necessarily more efficient. Due to the MHC as explained in Sect. 5.1 larger volumes may have to be dispensed. In the example in Fig. 5.5 a total of 1 nl Input A and 2 nl Input B is required to achieve the 1:3 mixing ratio and results in 2 nl of the target 1:3 ratio and 1 nl of byproduct in 1:1 ratio. Considering the MHC to be 1 nl, then using an arbitrary ratio mixer results in the mix of 1 nl of Input A and 3 nl of Input B. The overall fluid consumption is therefore higher, however 4 nl of the target 1:3 ratio is produced. In addition to the prolonged exposure of the reagents to air, arbitrary ratio mixers come with other downsides. Longer execution times (due to slow venting), more complex fabrication and the possibility of air bubbles getting trapped in the fluid make this technology infeasible for some assays. This leads to the conclusion that no mixing architecture is inherently superior, but rather the context of the application and how much fluid is required for further operations determines which architecture is more efficient.

Our automatic volume management algorithm presented in this paper can therefore be applied to architectures using either mixing technology, examples for both are shown in Sect. 5.4.

5.4 Volume Management

In this section we will present our automatic volume management algorithm in detail and provide examples using arbitrary mixing ratio architectures. Our algorithm can be applied to architectures with 1:1 mixing hardware as well, which however provides some additional challenges which are presented in Sect. 5.4.2.

Let us consider the application from Fig. 5.4 to be executed on the biochip from Fig. 5.2. We assume a MHC of 10 nl for all FFUs and a HTR of 1 nl. We propose a method that calculates the minimal FVA for every operation and then optimize the input fluid consumption by reusing waste. This method is implemented in Algorithm 2. Applying this algorithm to the application in Fig. 5.4 results in FVAs as shown in Fig. 5.9. In lines 1-12 the algorithm determines the minimal FVA for every operation in the application in reverse topological order.

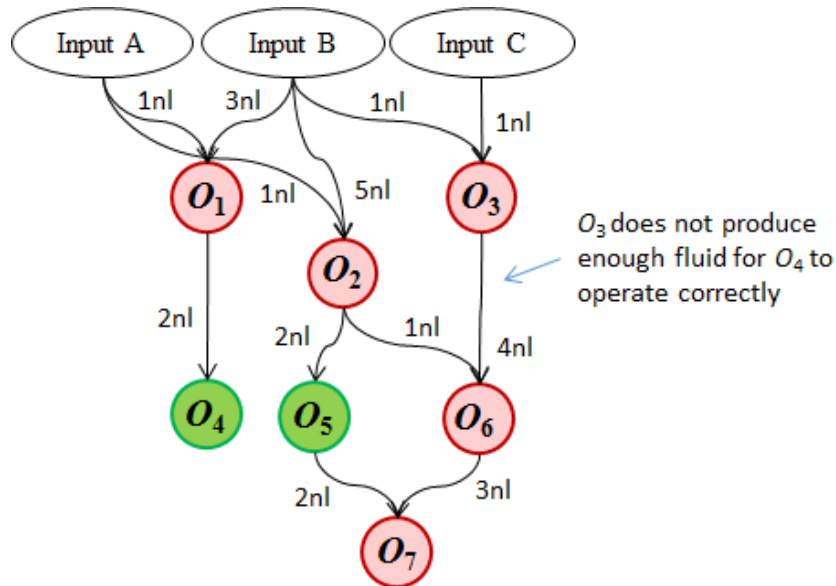


Figure 5.9: Application from Fig. 5.4 with fluid volume according to each operations MVR assigned, leading to shortage of fluid at O₆.

To determine the minimal FVAs, every operations MVR has to be known. A MVR represents a local minimal volume assignment as it is the smallest amount of volume required to successfully execute a single operation. While MVRs of FFUs such as detectors are already discrete amounts that can be dispensed, for mixing operations the MVRs that obey the hardware restrictions have to be determined from the given mixing ratios first. We calculate the MVR for mixing operations as follows: The smallest amount of fluid that satisfies both the HTR as well as the target mixing ratio is the sum of the products of the HTR and the ratio numerators, e.g. for O_1 from Fig. 5.4 which requires 1/4 Input A and 3/4 Input B we calculate $HTR * Numerator A + HTR * Numerator B = 1nl * 1A + 1nl * 3B$ resulting in an MVR of 1 nl of Input A and 3 nl of Input B.

Algorithm 2 Fluid Volume Assignment

Input: Application graph, HTR, MHC, MVR for each FFU

```

1: for each node  $n$  of the application in rev. order do
2:   for each outgoing edge  $e$  of  $n$  do
3:      $RF += e.RequiredFluid$ 
4:   end for
5:    $FVAs = n.MVRs$ 
6:   for  $X = 1; FVAs < RF; X++$  do
7:      $FVAs = MVRs \diamond (HTR * X)$ 
8:   end for
9:   if  $FVA > MHC$  then
10:    Cascading or Static Replication
11:   end if
12: end for
13: for each node  $n$  of the application do
14:   if  $n.CombinedInput > n.CombinedOutput$  then
15:      $LeftOverFluids.Add(n.LeftOverFluids)$ 
16:   end if
17: end for
18: for each leftover fluid  $LOF$  in  $LeftOverFluids$  do
19:   if At least one input of another operation can be completely or partially replaced by the  $LOF$  then
20:     remove  $LOF$  and update volume assignments and application graph
21:   end if
22: end for

```

Output: Optimized application graph

The MVR does however not consider how much fluid other operations in the application will require to successfully execute. Fig. 5.9 shows the application from Fig. 5.4 with MVRs

assigned to each operation. This however leads to underflow at O_6 since this operation requires 4 nl of fluid from O_3 , but O_3 only has a combined input, and therefore maximum output, of 2 nl. To prevent such scenarios, Algorithm 2 first determines, for every operation, how much fluid they have to pass on to following operations in lines 2-4. The algorithm does so in reverse topological order, making a single pass over all operations sufficient. Leafs of the application, which are calculated first, never have to pass on any fluid. Therefore they receive FVAs which are identical to their MVRs. For example O_4 in Fig. 5.4 receives 2 nl from O_1 and similarly O_7 receives 2 nl from O_5 and 3 nl from O_6 which is the MVR calculated from the mixing ratios and shown in Fig. 5.9.

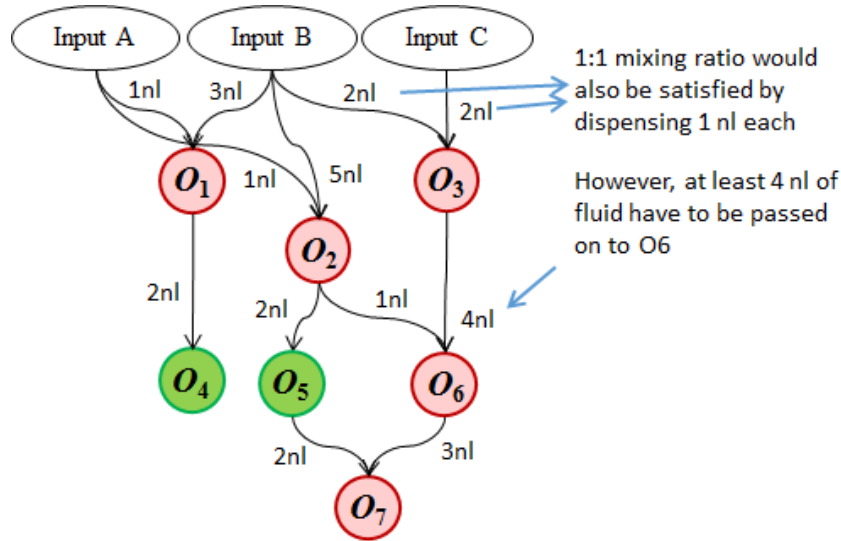


Figure 5.10: Application from Fig. 5.4 with optimal assignment of fluid according to the calculated FVAs.

All other operations have to assure that they process enough fluid to satisfy the following operations FVA which is checked in line 6-8. O_6 receives a total of 5 nl as input using its MVRs which is enough to satisfy the input of O_7 . O_3 however would only produce 2 nl of fluid using its MVRs of 1 nl from Input B and 1 nl from Input C as shown in Fig. 5.9. The algorithm then scales these inputs in line 7. We solve for x in the following equation: $RequiredOutput \leq \sum_{n=1}^{numMVRs} MVR_n \diamond (HTR * x)$, for $x \in \mathcal{N}$ where \diamond stands for multiplication for mixing operations, which scales the input while keeping the mixing ratio, and addition for other operations which determines the smallest multiple of HTR that satisfies the required output. For our example of O_3 we can see in Fig. 5.10 that both inputs have been scaled by a factor of two, keeping the mixing ratio as well as providing enough fluid to pass on

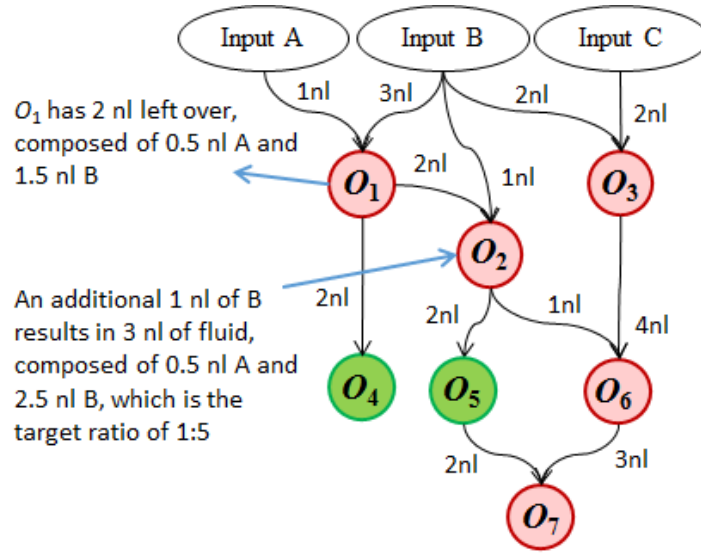


Figure 5.11: Application from Fig. 5.4 with optimal assignment of fluid according to the calculated FVAs and leftover fluid from O_1 has been reused as input for O_2 .

to O_6 . After determining the FVA the algorithm assures that no overflow will occur in lines 9-12. Overflow occurs whenever more fluid than the MHC has to be dispensed i.e. the FVA is larger than the MHC. This issue can be resolved through cascading or static replication as described in detail in [5].

5.4.1 Optimizing the mixing process

Accurately determining the FVA prevents dispensing of excess fluids to the inputs and thereby reducing waste, however fluids are also wasted during the applications execution, whenever an operation processes more fluid than it passes on. This is mostly the case for mixing operations, as large amounts of fluids may be needed to obtain the desired mixing ratio. 1:1 mixing hardware [98] has an inherent issue of producing waste for all ratios except for 1:1, as illustrated in Fig. 5.5. However, also variable mixing ratio hardware [4] can produce excess waste. Reusing these Left Over Fluids (LOF) does not only reduce the required volume of input fluids further, but can also remove the need to route such fluids to a waste port. Doing so can optimize our application even further as is shown in Fig. 5.11. O_1 has 2 nl of fluid left over after passing 2 nl on to O_4 , which is composed of 0.5 nl Input A and 1.5 nl Input B. By moving this LOF to O_2 and adding 1 nl of Input B, O_2 still satisfies its FVA and

mixing ratio while requiring less fluids from inputs. As this example shows, reusing LOFs can completely and/or partially replace previous inputs. LOFs can potentially be assigned to any operation that is not a predecessor of the operation that created it and that contains the same basic reagents. E.g. the LOF from O_1 in Fig. 5.11 can be reassigned to O_2 because both operations require reagents from Input A and Input B. O_3 would never be considered as a target for the same LOF since this operation is not supposed to contain any amount of Input A. To successfully reassign as much LOFs as possible, a thorough analysis of all given data is necessary as we will show using the example from Fig. 5.12. This example consists of an application graph of a glucose test where black labels indicate the MVR and the red labels the FVA. Here, O_2 , O_3 and O_4 produce LOFs. Our algorithm determines these LOFs in lines 13-17. The data required for optimal LOF reassignment consists of the LOF volume, the operation that creates it and the fluid composition regarding the applications original inputs, i.e. for the glucose example, how much Glucose, Reagent and Sample the LOF contains. O_3 from Fig. 5.12 for example, receiving a total of 5 nl as input and passing 2 nl on to O_8 , creates a LOF with a volume of 3 nl, containing 1.5 nl Glucose and 1.5 nl Reagent.

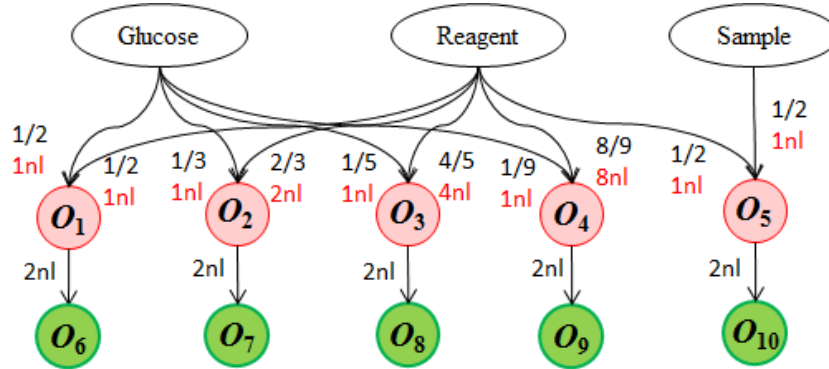


Figure 5.12: Application graph for a glucose test. Black labels indicate the required ratios and discrete volumes, red labels indicate the FVAs.

Our algorithm assigns the LOFs to operations in lines 18-22. As previously mentioned, a LOF can be used whenever it allows to partially or completely remove one or more incoming edges of an operation, if the resulting volume still obeys the target operations FVA and the ratios of the contained fluids remains correct. However, it is also possible to only use part of the LOF, or combine multiple LOFs in order to meet these requirements. The assignment of LOFs containing multiple fluids is made possible by a large extend due to acceptable error margins when mixing fluids [62]. One of these cases is shown for our example from Fig. 5.12. Using the initially calculated FVAs, operations O_2 , O_3 and O_4 produce LOFs. The algorithm iterated

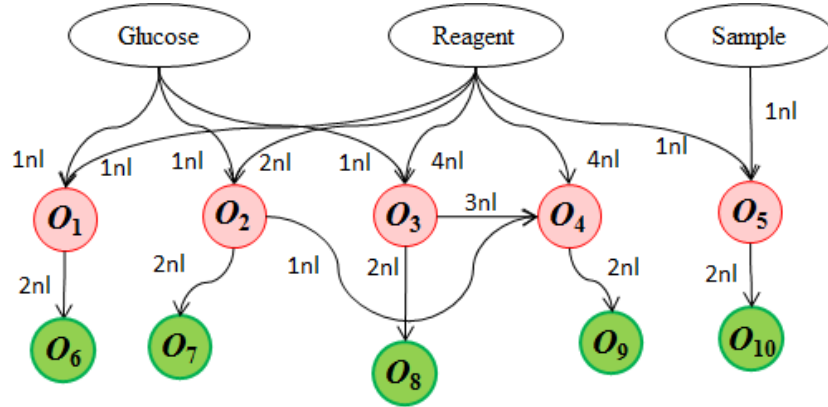
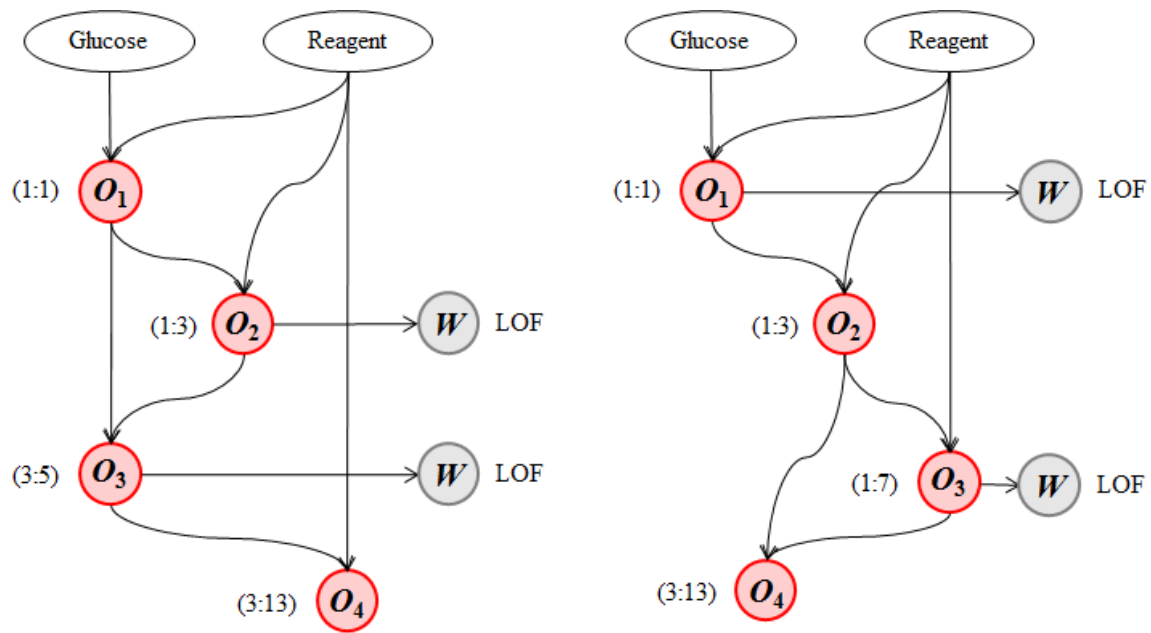


Figure 5.13: Optimized application graph from Fig. 5.12, where left over fluid from O₂ and O₃ are used in O₄. All values indicate FVAs.

through all possibilities to reassign these LOFs and determines that the LOF produced from operations O₂ and O₃ can be used as partial input for O₄, as shown in Fig. 5.13. Here, O₄ no longer receives 1 nl of Glucose (G) and 8 nl of Reagent (R) from inputs, but instead 0.33 nl G and 0.66 nl R from O₂, 0.6 nl G and 2.4 nl R from O₃ and 4 nl R from an input. This corresponds to a total of 0.93 nl G and 7.06 nl R, or a ratio of 0.93:7.06 which is 0.6% off the desired ratio of 1:8, which is within acceptable error margins. As a result O₄ consumes 1 nl G and 4 nl R less from inputs, reducing the overall volume requirements of the application as well as the need to route LOFs from O₂ and O₃ to waste ports.

5.4.2 Deriving mixing trees for 1:1 mixing hardware

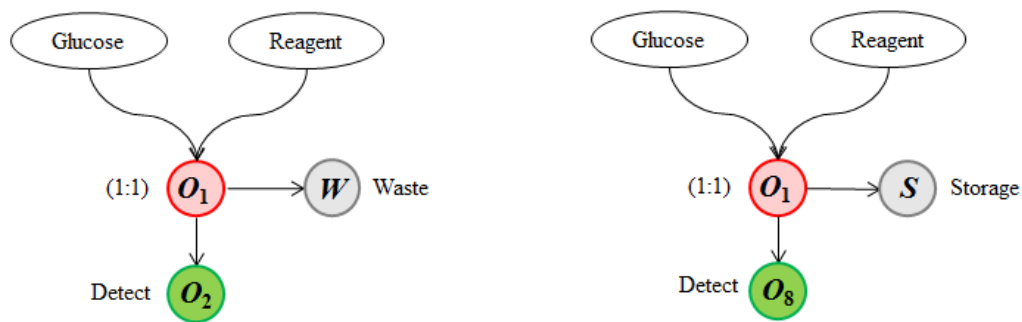
Assuming 1:1 mixing hardware is used instead of arbitrary mixers, some extra steps are necessary for the fluid consumption to be optimized. We determine mixing trees as explained in Sect. 5.3 to obtain our target ratios. NFB provides a solution that requires the smallest volume of input fluids. However, multiple solutions consuming the same amount of fluid might be available. These solutions differ in the ratio of the LOF that they create during the production of the target ratio. Fig 5.14 shows an example of this where the overall fluid consumption and result are the same, but different ratios of the input fluids are left over. We therefore have NFB calculate all solutions that require the same amount of input fluid allowing our volume management algorithm to choose the mixing tree that produces the most useful LOFs.



(a) Mixing of ratio 3:13 creating excess fluid of ratios 1:3 and 3:5

(b) Mixing of ratio 3:13 creating excess fluid of ratios 1:1 and 1:7

Figure 5.14: Two mixing trees using the same input fluids to produce the same output fluids, but creating different LOFs due to different mixing order.



(a) Mixing tree created by NFB for ratio 1:1

(b) Optimized mixing tree for ratio 1:1 storing the LOF of O_1 for use in another mixing tree

Figure 5.15: Mixing trees for target ratio 1:1 which can be precisely mixed and no approximation is necessary

We apply this technique to our previous example from Fig.5.12, but only focus on the four required mixing ratios of G and R. As mentioned previously, 1:1 mixing hardware has the inherent problem that not all mixing ratios are obtainable. Some ratios therefore have to

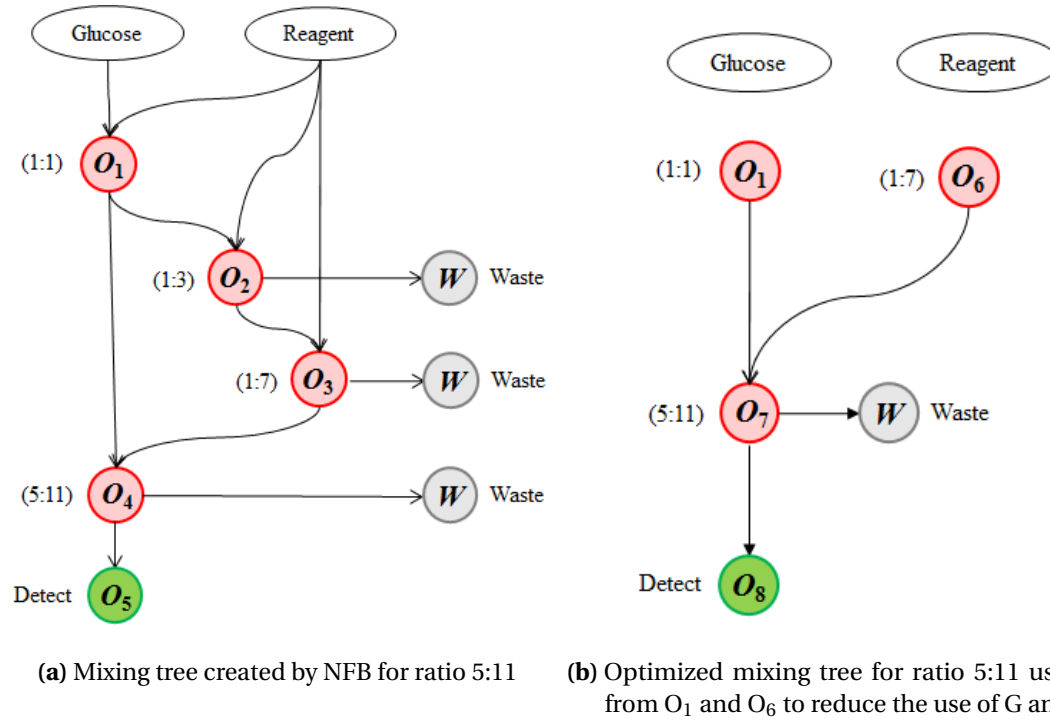
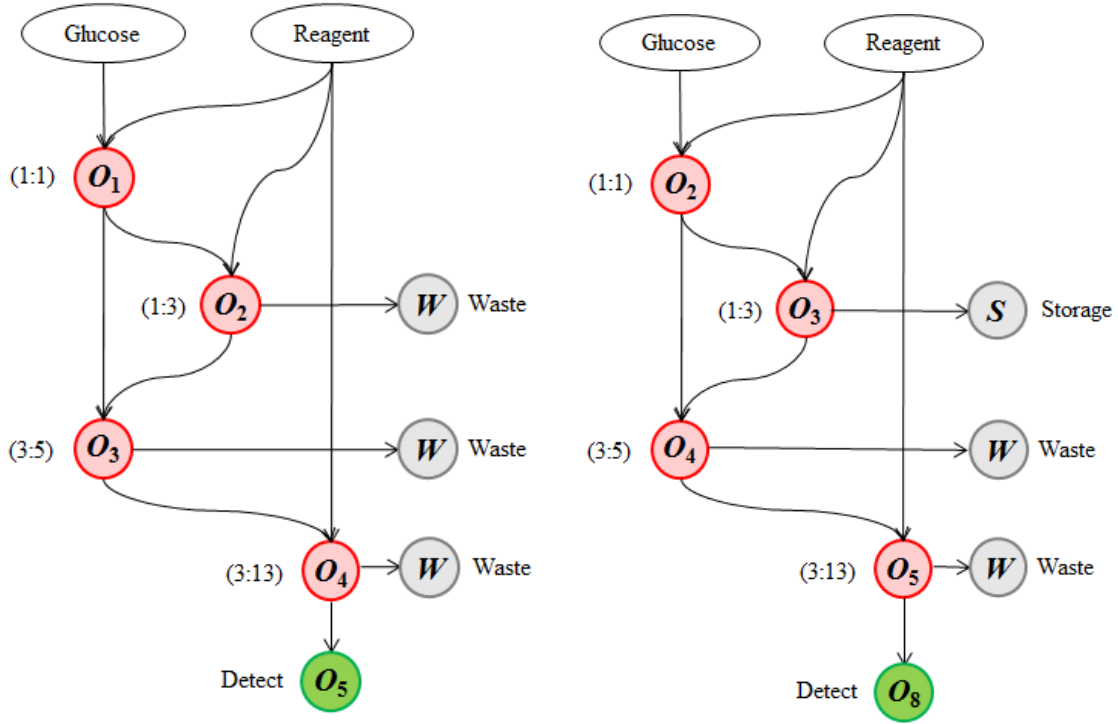


Figure 5.16: Mixing trees for target ratio 1:2 which is approximated using a maximum of 4 operations to ratio 5:11

be approximated. The resulting error is dependent on the number of consecutive mixing operations, where more mixing operations allow for a smaller error in the approximated ratio. For our example, we choose a maximum number of mixing operations, or precision level, of 4 for each approximation. The resulting mixing trees are shown in Figs. 5.15a, 5.16a, 5.17a and 5.18a. The target ratio 1:1 can be produced correctly, while the ratio 1:2 is approximated to 5:11, ratio 1:4 is approximated to 3:13 and ratio 1:8 is approximated to 1:7. These mixing trees are independent from each other and produce the target ratio by mixing reagents from the G and R input reservoirs. In most cases, the mixers output will not exactly match the detectors requirement and some fluid is discarded, e.g. a mixer with a capacity of 4 nl will pass on 2 nl to the detector and discard 2 nl. Our algorithm now minimizes the fluid consumption by reassigning the discarded fluids. The resulting mixing trees are shown in Figs. 5.15b, 5.16b, 5.17b and 5.18b. The mixing trees are no longer independent and the operation sequence is now shared across all mixing trees. However, using this setup and reassigning LOFs, the combined consumption of all four mixing trees is only 2 units of G and 6 units of R. In comparison, supplying the results of NFB with the required inputs without reassigning LOFs requires 4 units of G and 9 units of R.



(a) Mixing tree created by NFB for ratio 3:13

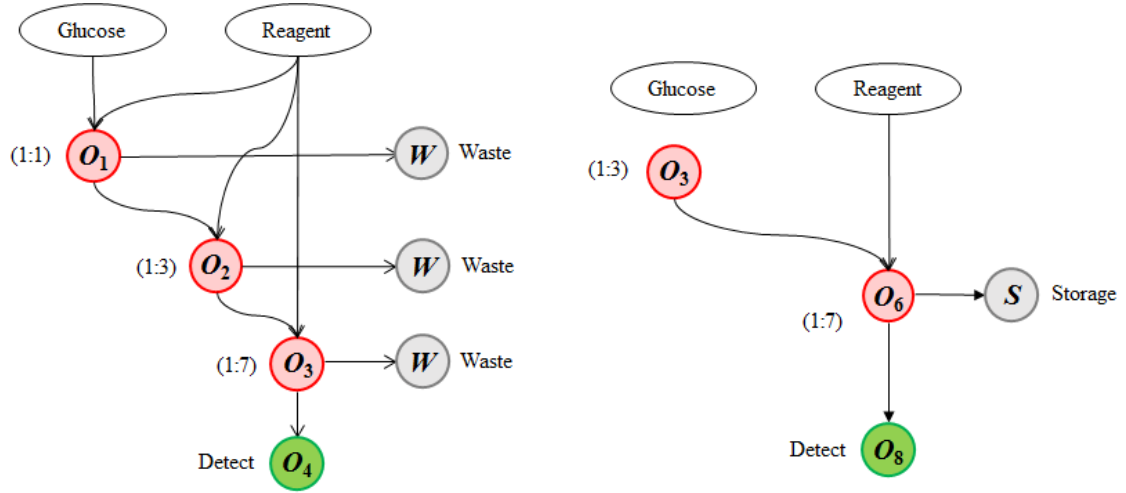
(b) Optimized mixing tree for ratio 3:13 storing the LOF of O_3 for use in another mixing tree

Figure 5.17: Mixing trees for target ratio 1:4 which is approximated using a maximum of 4 operations to ratio 3:13

5.5 Experimental Evaluation and Discussion

We have shown in our examples in Sect. 5.4 that we can reduce the volume of waste created by both arbitrary and 1:1 mixing hardware. For 1:1 mixing hardware, the result is reliant on the mixing tree optimization used as explained in Sect. 5.3. We have implemented and compared Min-Min (MM), REMIA, and the Network Flow Based (NFB) algorithms. The results are shown in Table 5.1, which incorporates the data of multiple test cases at three different precision levels for all three optimization techniques. We determined that NFB, even if pruned to four vertices, and therefore no longer optimal but significantly faster, noticeably outperforms both MM and REMIA in both number of operations and fluid consumption.

In the next set of experiments, we have evaluated our Fluid Volume Assignment (FVA) solution on real-life assays to validate the results and determine the efficiency of our technique.



(a) Mixing tree created by NFB for ratio 1:7

(b) Optimized mixing tree for ratio 1:7 using LOF from O_3 to reduce the use of G and R and storing the LOF of O_6 for use in another mixing tree**Figure 5.18:** Mixing trees for target ratio 1:8 which is approximated using a maximum of 4 operations to ratio 1:7

	Precision Level 8			Precision Level 9			Precision Level 10		
Algorithm	MM	REMIA	NFB4	MM	REMIA	NFB4	MM	REMIA	NFB4
Fluid Cost	219.3	135.2	49.3	278.7	169.7	56.1	334.6	178.9	56.2
Operations	59.4	76.2	41.4	73.0	100.6	51.3	85.3	107.9	59.3
Time (s)	0.01	0.02	1.19	0.01	0.03	1.36	0.01	0.04	6.74

Table 5.1: Comparison of fluid consumption, number of operations and execution time for MM, REMIA and NFB4 (NFB pruned to 4 vertices), at precision levels 8, 9 and 10. All results represent the average outcome of 10 test cases, 8 synthetic and 2 real world applications

We present the results of two additional assays, namely a Colorimetric Cholesterol Assay (CAA) as described in [93] and a Proteasome Activity Assay (PAA) as described in [95]. The experimental results for these assays can be seen in Table 5.2. For simplicity, we refer to the fluids used in both assays as Reagent (R) and Buffer (B), the actual components used in these assays can be found in the respective assay protocols. Both assays require 5 samples of R and B mixed at different ratios. For CAA the desired output volume for each sample is $50 \mu\text{l}$ and for PAA $100 \mu\text{l}$. The table data shows: 1) The assay name. 2) Minimal requirements of Reagent (R) and Buffer (B) to satisfy the target ratios and output volumes. 3) Consumption of R and B after approximation using NFB4 and assuming mixing hardware size that fits the required output volumes (i.e. $50 \mu\text{l}$ for CCA and $100 \mu\text{l}$ for PAA). 4) Same as 3, but assuming mixing hardware that fits double the volume. 5) Consumption of R and B after applying our

FVA, assuming mixing hardware size that fits the required output volumes. 6) Same as 5, but assuming mixing hardware that fits double the volume.

The CCA target ratios have been approximated to a precision level of 4, resulting in the approximated ratios: 1:3, 1:15, 3:13, 5:11 and 3:5 of R:B respectively. The PAA target ratios have been approximated to a precision level of 6, resulting in the approximated ratios: 1:7, 3:29, 5:59, 1:15 and 1:31 of R:B respectively. As Table 5.2 shows, our FVA technique can reduce the required volumes significantly, for CAA the use of R and B are reduced to 60% and 75% and for PAA to 40% and 62%, respectively, compared to the results obtained using the mixing trees obtained from NFB without optimization. These experiments assume mixing hardware which has a capacity that exactly matches the desired output volume for each assay, marked M_1 in the table (i.e. 50 μl for CCA and 100 μl for PAA).

We also determined the results for the same assays for the case when larger capacity mixing hardware is available, with a volume double to the desired output volume of the assays (i.e. 100 μl for CCA and 200 μl for PAA). While the absolute fluid consumption rises as is to be expected when using larger mixing hardware, the optimization factor increases as well compared to the previous results. Using these larger mixers (the results marked M_2 in Table 5.2) we obtain a reduction of R and B in CAA to 40% and 50% and for PAA to 20% and 43% respectively, compared to unoptimized NFB results, using such larger mixers as well. This increased optimization is an advantage if architectures are used which are not specifically designed for one assay, but are general purpose and might therefore use larger mixers.

While the required fluid volumes compared to the naive NFB solutions are significantly smaller, there is still a noticeable gap to the minimal requirements. The M_1 solution of CCA requires 25% more R than the minimum and the M_1 solution of PAA requires more than 300%. This is essentially the same Hardware Transport Resolution (HTR) issue as discussed before. Due to the mixers fixed chamber size, only multiples of a certain volumes can be used. For example, CAA requires an output volume of 50 μl . The used mixers therefore have a capacity of 50 μl total, or 25 μl per mixing chamber, which can be considered the HTR for this case. With 75 μl being the smallest multiple in the HTR that is larger than the minimal requirement of 60 μl , the optimization is only hindered by the hardware restriction. The PAA optimization suffers from the same issue, however as it occurs multiple times throughout its mixing tree, the fluid consumption is significantly higher than the minimal fluid requirements. To improve

Assay	Minimal	NFB M ₁	NFB M ₂	Opt M ₁	Opt M ₂
CCA	60 R; 190 B	125 R; 300 B	250 R; 600 B	75 R; 225 B	100 R; 300 B
PAA	30 R; 470 B	250 R; 1050 B	500 R; 2100 B	100 R; 650 B	100 R; 900 B

Table 5.2: Experimental results for optimization of a Colorimetric Cholesterol Assay (CCA) and a Proteasome Activity Assay (PAA).

these results, additional mixing hardware would be required with different capacities (e.g. one mixer of 50 μl and one of 25 μl capacity) and/or different ratios (e.g. one mixer with 1:1 ratio and one with 1:2 ratio). This would however negatively impact the experiment in several ways, such as number of operations, architecture complexity and execution time.

Arbitrary ratio mixers can provide superior results in certain circumstances, for example CAA can be solved using the minimal fluid requirements if sufficiently large arbitrary ratio mixers and a HTR of 4 μl are available, as all required volumes can be approximated to multiples of 4 μl . While this would seem like the preferable solution for volume management, the significant disadvantages of arbitrary mixers mentioned previously make it unlikely that they can completely replace fixed ratio mixers.

5.6 Conclusion and Future work

The proposed fluid volume assignment technique allows for a more autonomous use of FBMBs by reducing the need for user intervention by determining required fluid volumes beforehand, preventing over- and underflow. Additionally, the application's fluid consumption is minimized by reusing waste generated during its operation. Our application graph optimizations are carried out early in the biochip's design process, allowing the required complex computations, such as determining mixing trees, to be done at a none time critical stage. On the other hand, the necessary fluid assignments and LOF reassignments can also be determined during runtime if necessary, for example as part of the recovery from an error.

We have shown that the fluid consumption of applications as well as the volume of generated waste can be reduced for architectures using either 1:1 or arbitrary mixing ratio technologies. Determining an optimal architecture, potentially utilizing both technologies in parallel, to further optimize the fluid consumption is possible using our presented algorithm, but is left

for future work. While this paper focuses on determining the required volume, the same technique can also be used to determine possible actions, given a volume of fluid. In cases where the volume of fluid available is unknown beforehand, such as after an error or when reagents are gathered on-site, the algorithm can be used to determine which operations can be executed, given the available volume.

Chapter 6

A Novel Metering Component for Volume Management

This chapter is based on the contribution P5 as described in Sect. 1.4.

6.1 Introduction

The continuous flow microfluidic biochips market is so far still dominated by simple, mostly passive biochips as described in Sect. 2.1. Due to their simplicity, they are cheap to fabricate and highly reliable, but also limited in their functionality. A proper transition to active, multi-purpose biochips capable of complex functionality requires at the very least a similar level of reliability, to allow their use in a broad range of applications. In this chapter we specifically look at the possibility to align and meter fluids inside the chip, as well as online error detection for insufficient volumes. The possibility to align fluids to a precise location is either assumed or the issue ignored in most publications about routing or scheduling [70, 77], but is clearly necessary to transport fluids to the desired location specified by the routing and scheduling algorithms. The ability to precisely meter volumes has several advantages regarding volume management as explained in detail in Chapter 5. Metering is furthermore necessary for reliable execution of most multi-operational assays, as volumes of fluids can change through operations such as mixing, separation and incubation (expanding fluids)

or through faults such as evaporation and leaks. Finally, as discussed in detail in Sect. 3.3, fault detection plays an important role in reliable biochips. The ability to detect occurring faults during the execution of an application is very limited at the time. A reliable detection of insufficient fluid volumes during runtime can prevent further operations from executing with an erroneous fluid volume, potentially saving time, reagents and data.

All three functions are made possible by venting air from the biochip through the gas permeable PDMS. We use venting technology which was previously introduced in [4] [31]. However, in contrast to the chip presented there, which are fully made from PDMS, we adapt this technology to our biochip design presented in Sect. 3.4.

In our design process shown in Fig. 6.1 we can see that our proposed component extends the currently existing component library. Assuming that no components for alignment, metering and error detection were available before, this extension allows to start the physical synthesis process for application graphs which require one of these operations, of which so far no component was capable. If individual components with such functionality have already been included in the component library, this addition allows to reduce the number of required components in the architecture, as all three operations can be executed in a single component.

6.2 Venting

The gas permeability of PDMS has previously been discussed in different disciplines, including microfluidics [4] [31] [55]. While this effect is often the subject of fault analysis as liquids can evaporate through the PDMS, we are interested in actively using this effect to expel unwanted gases. The gas permeation effect through PDMS is described in different forms, such as the gas permeation equation [55] [38], or Darcy's Law [4, 85], which however also point out that it is difficult to develop a generalized expression for the permeation of gas through PDMS in microfluidic biochips. This is partially due to the diverse features of PDMS based biochips, but other factors such as the non linear behaviour of gas permeation through PDMS [123] or permeability thickness dependence [36]. We use Darcy's Law (Equation 6.1) to describe the permeation process for this paper. While Darcy's law can be use to estimate the venting time to some degree, it is important to note that Darcy's Law describes the flow of a

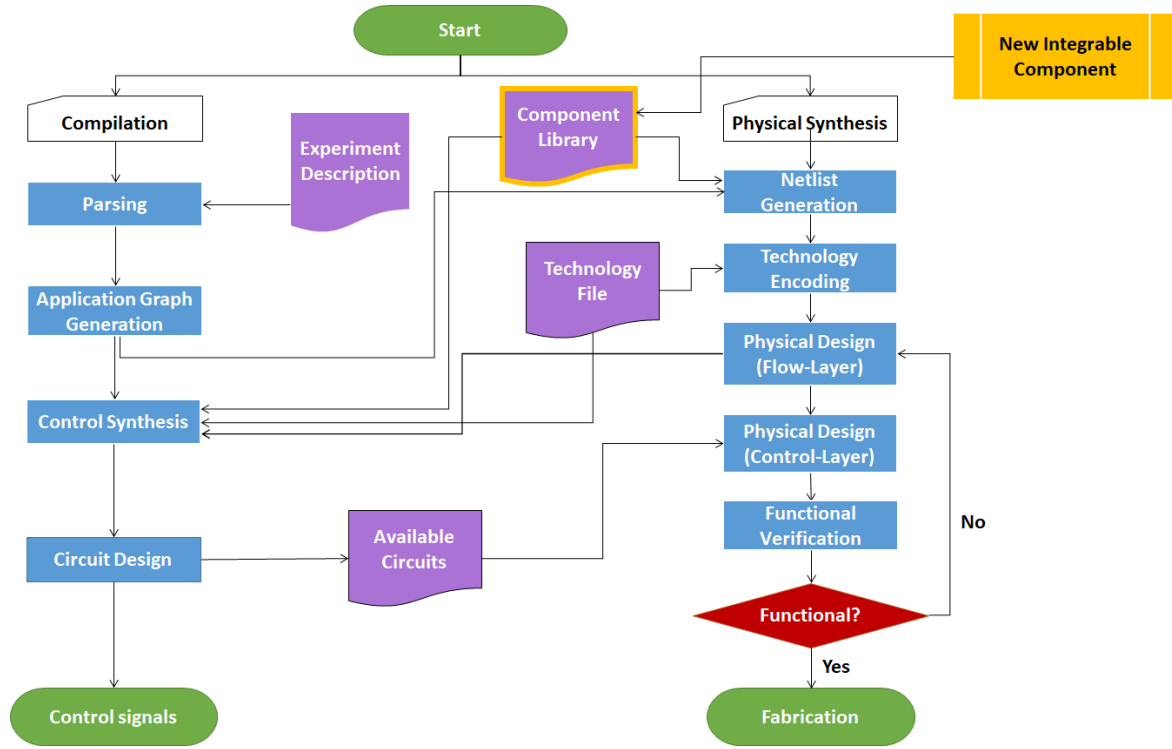


Figure 6.1: The proposed component is added to the component library, allowing other design steps to make use of it

fluid through a porous medium. Therefore it does not incorporate all parameters required in our situation. For example, while air is vented through the membrane, liquid of higher viscosity is pushed through the channel, in addition to the previously pointed out issues regarding permeation prediction. However, Darcy's Law clearly lays out all possible adjustable parameters to alter the venting duration. Fig. 6.2 shows where these parameters are applied on a PDMS membrane as used in FBMBs. The intrinsic permeability of the PDMS k measured in m^2 and the viscosity μ in Pascal-seconds of the vented air is considered constant in this paper. This leaves the membrane thickness L in meters, the pressure difference across the membrane P_A and P_B in Pascal and the area of exposure A in m^2 as adjustable parameters leading to the discharge Q in m^3/s . In this paper we propose three uses for gas permeability in FBMBs (1) Alignment, (2) Metering and (3) Erroneous Volume Detection and show in experimental results how the before mentioned parameters effect the duration required to perform these actions.

$$Q = \frac{(k * A * (P_A - P_B))}{(\mu * L)} \quad (6.1)$$

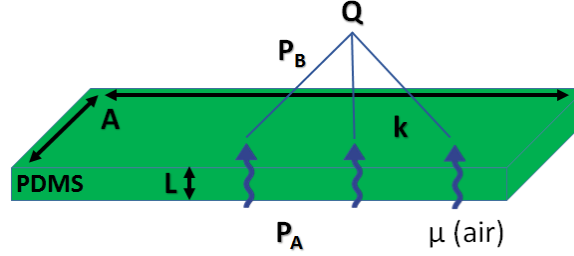


Figure 6.2: Visualization of Darcy's Law parameters applied to a PDMS membrane.

6.3 Alignment and Metering

Accurate metering of volumes in FBMBs remains a difficult task. Previous work has introduced small scale pumps, which are capable of displacing very small volumes of about 100 pl [41]. However, displaced volumes would vary depending of the fluids viscosity. Furthermore, correct alignment would, in addition to accurate displacement, require precise knowledge of the current location of the fluid on-chip (e.g. through previous fluid alignment). Finally does this restrict the type of pressure sources to such on-chip pumps.

To obtain a less restrictive method of on-chip metering, we make use of the venting technology introduced in [4] [31], which has shown that trapped air, which would otherwise hinder operations such as mixing of fluids, can be expelled from a channel using vents. This is possible through the gas permeability of the PDMS that the chip is build from. We use the same principle to vent air from our chip, however in our design only a thin membrane is made from PDMS, which separates the two layers of the chip which are made out of non gas permeable PMMA. We propose a venting structure in Fig. 6.3. The trapped air is vented from the Flow-Layer, through the PDMS membrane into the Control-Layer. There, control channels with access holes to the atmospheric pressure outside the chip are milled, allowing the vented air to dissipate outside of the chip. Venting occurs by introducing a pressure difference in the flow- and control channel at the vent. This pressure difference can be achieved by pushing fluid onto the closed valve, compressing the air trapped between the fluid and the closed valve. Alternatively, vacuum pressure can be applied to the control channel side of the vent, leading to a pressure difference between the layers as well.

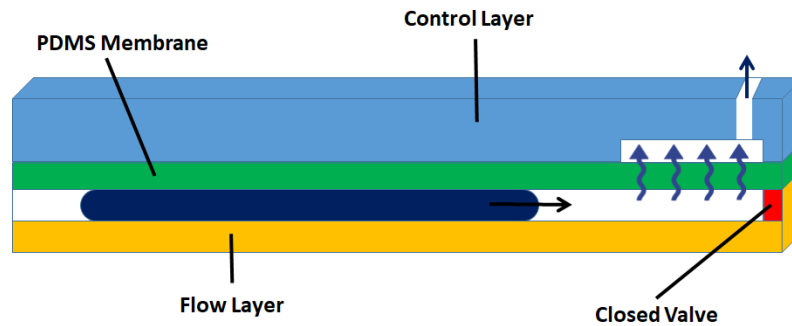


Figure 6.3: Metering channel containing an alignment vent. Fluid is pushed towards the closed valve (e.g. by an off-chip pressure source) and the trapped air is vented out of the chip through the PDMS membrane.

Using such Alignment Vents (AV) allows to align the fluid on-chip to a precise location. We furthermore propose to use this novel component to meter fluid volumes on-chip. A metering channel has to be fabricated to hold a specific volume of fluid which is adjustable through the channel length, width and depth. Such a channel requires a valve on one end and an adjacent vent. Fig. 6.4 shows how such a metering channel can be used: The metering channel is filled with fluid from a source in Fig. 6.4a. Once the trapped air between the fluid and the closed valve has been vented, the connection to the source is interrupted as shown in Fig. 6.4b. This assures that only the metered volume is moved and no excess fluid is added to the metered volume.

6.4 Detecting missing fluid

As a step towards more reliable FBMBs we propose an extension to the metering channel to enable detection of missing fluids as well. Biochemical assays have precise fluid requirements, and any errors that reduce the fluid volumes will lead to failure. The design of the previously proposed metering channel introduced in Fig. 6.3 is extended by another vent located on the opposite end of the AV in the metering channel as shown in Fig. 6.5. This Volume Error Detection Vent (VEDV) is fabricated the same way as the AV.

When this metering channel is filled with fluid and excess fluid is available, i.e. more than the metering channel's capacity, the addition of the VEDV has no effect on the metering channel as it is covered by fluid throughout the metering process. However, if too little fluid

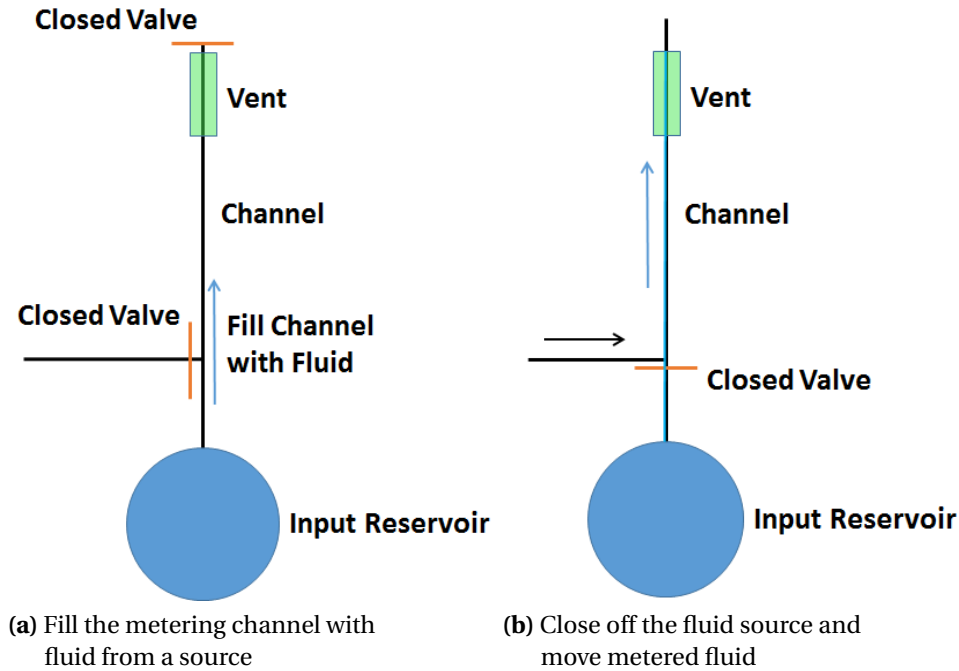


Figure 6.4: Metering using an Alignment Vent

is available, i.e. less than the metering channel's capacity, the VEDV will not be covered by fluid once the available fluid is aligned at the closed valve. The pressure that moved the liquid into the metering channel will now compress the air behind the fluid, causing some of the air to be vented through the VEDV. By detecting this venting of air through the VEDV (e.g. with a micro pressure sensor [106]) we can conclude that too little fluid is available to provide the desired volume. Depending on the architecture, dedicated VEDVs might be omitted. The normally-closed Grover valves as described in Sect. 2.2.2 are opened by applying vacuum pressure to their control channel. However, sufficient pressure at the flow channel will cause the PDMS membrane to deflect as well, opening the valve without any applied vacuum pressure. Therefore, the valve at the chamber's input can be repurposed as VEDV, as shown in Fig. 6.6. The pressure applied to the flow channel forces the valve open and in case the valve is not covered with fluid (i.e. insufficient fluid is available), some of the compressed air vents through the valve. As no vacuum pressure had to be applied to the valve control, the increasing pressure in the control channel can then be detected by a pressure sensor.

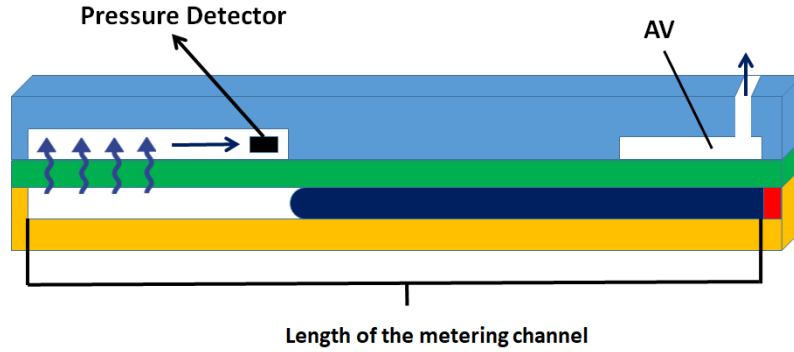


Figure 6.5: Volume error detection enabled metering component. In addition to the alignment vent, a volume error detection vent is added at the opposite end of the metering channel. If too little fluid is available to fill the metering channel completely, air escapes through the VEDV which is detected by a pressure sensor.

6.5 Experimental results

To test our proposed component that combines alignment, metering and fluid volume error detection features, we have fabricated prototypes, such as the chip shown in Fig. 6.7. To determine the functionality and advantages of architecture and material variations we have performed multiple tests and report the results in Table 6.1. We report the outcome of the experiments as fluid displacement in nanolitres per minute, which is calculated from the channel width and depth, and the duration it takes to move fluid 10 mm within the channel (unless specifically noted differently). Videos of the experiments and detailed information about the shown architectures are available online [105].

6.5.1 Setup

Our chips are fabricated from 70x70x3 mm cast plexiglass [40] pieces. Channels on both layers of the chips are milled using a Minitech Machinery Mini-Mill/3 [63]. We use 800 μm flat-end mills for the chip's channels, a 2.5 mm flat-end mill for control-pin holes and a 200 μm flat-end mill for smaller structures such as the grooves introduced later in this chapter. Even though much more shallow channels are possible to fabricate, we use a channel depth of at least 120 μm to assure that minor variations, for example caused by the thickness tolerance of the plexiglass, gave no significant impact on our measurements. Valves are formed by interrupting the channel on the Flow-Layer for a length of 1 mm. The

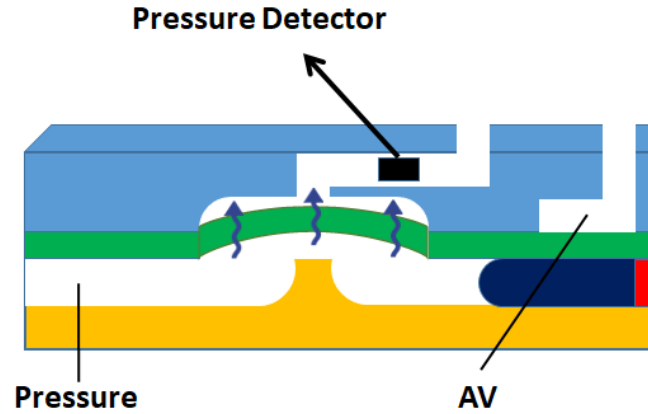


Figure 6.6: Pressure in the flow channel forces a grover valve open, with out any applied vacuum pressure. Air vented through the membrane at the valve can be detected by a pressure sensor.

corresponding displacement chamber on the Control-Layer measures 1.5 mm width and 2.5 mm length. The two layers are separated by a 100 μm or 250 μm thick PDMS membrane [61]. The plexiglass pieces are bonded to the PDMS membrane using a hydraulic press with heated plates at 7 kN of pressure and 70 degrees Celsius for 10 minutes, followed by a cool down phase to 50 degrees Celsius at 3 kN of pressure for approximately 15 minutes. The vacuum and pressure required to operate the chips is provided by a compressor and a vacuum pump. Each can provide an adjustable amount of pressure, which is split into multiple, individually controllable tubes using solenoid valves. These tubes can then be connected to the chip. More detailed information about the fabrication process can be found in Sect. 3.4.

For our tests we used architectures similar to the one shown in Fig. 6.7, containing a metering chamber and a 3-way switch, allowing to place fluids within the chip before using the metering chamber. Once the fluid is placed in the chip, vacuum only venting can be performed by applying vacuum pressure to the AV and V_2 . If pressure is applied to In1 for faster venting, pressure also has to be applied to V_1 and V_3 to assure the pressure in the flow channel does not unintentionally open these valves. The vacuum pressure at the V_2 control can be omitted in this case, as the pressure from the flow channel forces the valve to open as explained in Sect. 6.4, resulting in a total requirement of 1 vacuum and 3 pressure sources.

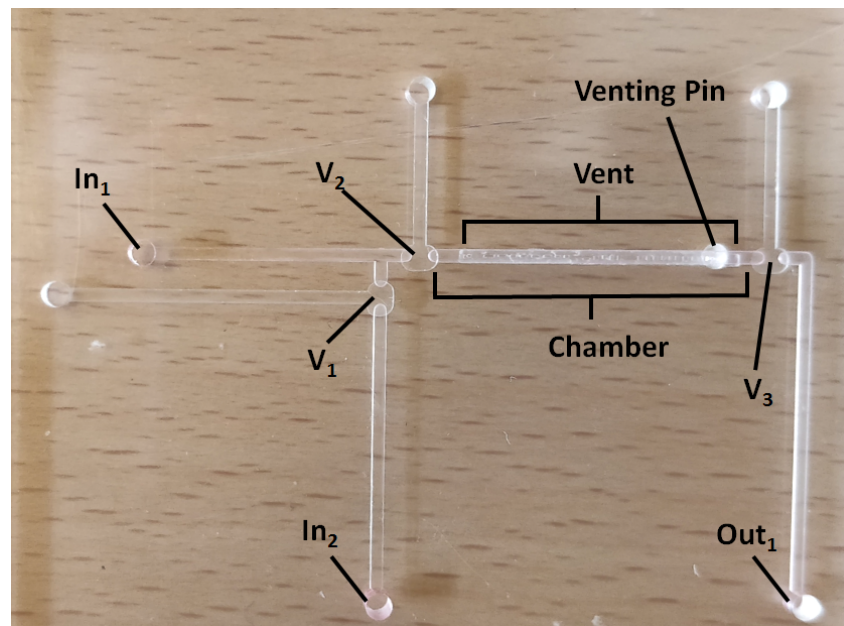


Figure 6.7: Prototype of a metering chamber with a large vent (19 mm vent length, 22 mm chamber length) for fast venting. Fluid is introduced through In_1 or In_2 by opening V_1 . Alignment of the fluid in the chamber can be started by opening V_2 and applying vacuum pressure to the venting pin.

6.5.2 Vent Architecture

Two vent architectures have been fabricated. (1) Channels on the Control-Layer used for venting are placed directly over the channel in the Flow-Layer and are connected to individual outlets. This solution is easy to fabricate and provides maximum surface area for gas permeation as the vent covers the whole width of the flow channel. However, as tests have shown, the vacuum pressure can cause the PDMS membrane to bend into the control channel as shown in Fig. 6.8. As part of the membrane is pressed against the top of the control channel, part of the surface area is lost. The data in rows 2 to 5 in Table 6.1 show that as a result, an increase in the vent's surface area does not have the desired outcome using this straight forward vent design. Doubling the surface area only results in a <50% increase in fluid displacement. It is possible to counteract this, by milling significantly deeper control channels, providing additional space for the membrane to deflect into. Tests on the same architecture with control channel depths of 1 mm show a significant increase in fluid displacement as shown in rows 6 and 7 in Table 6.1 with vent areas marked "(D)". This alteration does however require the chip to be sufficiently thick, and increases the chance of bonding failure and even tears in the membrane at high pressures due to the significant

deformation.

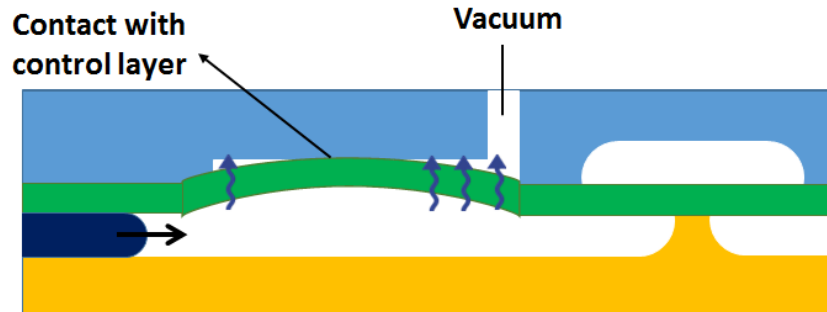


Figure 6.8: Vacuum pressure at the vent causing the PDMS membrane to deform into the control channel, onto the control layer, effectively reducing the surface area.



Figure 6.9: Control-layer part of a groove vent. Small parts of PMMA are left in the vent, keeping the PDMS membrane from significantly deforming into the control channel when vacuum pressure is applied

Such deeper control channels furthermore put more emphasis on the Chamber Volume Extension (CVE) caused by the deflected membrane. As the membrane is deflected into the control channel as shown in Fig. 6.8, the chambers volume increases. While this does not necessarily affect the components basic alignment and metering functionality, it does cause other issues. The increased chamber volume requires more fluid to be available to fill the chamber completely. Once the alignment is finished and the chamber is filled, cutting the vacuum pressure from the vent will cause the membrane to quickly return to its relaxed state, causing significant backflow of the excess fluid from the chamber into the chip. While the amount of backflow depends on the vent size and control channel depth, we have observed that even regular depth control channels allow for enough volume increase, that the resulting backflow is able to force open valves, if not actively kept shut.

(2) Alternatively, we milled rectangular grooves of 200 μm thickness in the control layer, leaving small parts of PMMA between them as shown in Fig. 6.9 and schematically in Fig. 6.10. This keeps the PDMS membrane from significantly deforming into the control channel, while

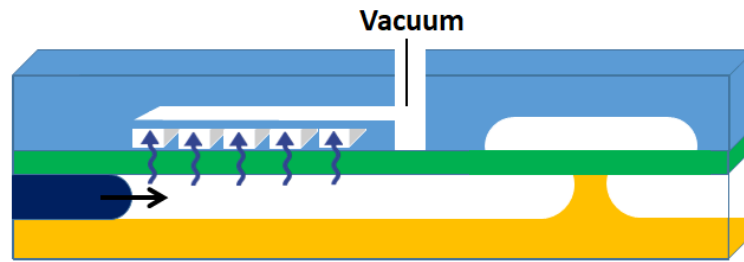


Figure 6.10: Grooves are left in the vent, keeping the PDMS membrane from deforming with minimal impact of the surface area.

minimizing the decrease in surface area relative to the occupied area. Results in rows 1, 8 and 11 in Table 6.1 (with vent areas marked “(G)”) show that grooved vents perform significantly better, even at smaller vent areas or lower pressure compared to our previous solution. This design does not only allow for faster venting, but also alleviates the problem of CVE, by only allowing minor deflection of the membrane into the small grooves.

6.5.3 Fluid displacement

The necessary pressure difference at the vent can be achieved by two means. (1) Applying pressure to the flow channel, i.e. pushing the fluid towards the vent, which creates high pressure in the flow channel between the fluid and the vent. (2) Applying vacuum pressure at the vent outlet (on the control-layer) causes the pressure to drop in the flow channel as well, creating suction to move the fluid towards the valve. It is possible to apply both techniques at the same time, resulting in a faster venting as shown in rows 9 and 12 in Table 6.1. However, using only vacuum pressure to align the liquid has proven beneficial, as faults are unlikely to occur due to the applied pressure. Even at high pressure (-60 kPa), no component failure was detected. Applying high pressure at the flow layer is far more likely to cause faults such as leaks as it puts a direct strain on the chip’s bonding. Furthermore, Grover valves are susceptible to unintentional opening by high pressure on the flow-layer, without any activation pressure at the control-layer. Even low pressures (< 5 kPa) in the flow channel can be enough to force a valve open. To avoid this, the same pressure has to be applied to the valve control as well, forcing the valve to stay shut.

As the fluid fills the chamber and starts to partially cover the vent, the effective vent area decreases, causing the fluid displacement to slow down. Rows 12, 13 and 14 in Table 6.1 show

Row No.	PDMS Thickness	Vent Area (mm^2)	Vaccum	Pressure	Fluid displacement
1	250 μm	1.5 (G)	-40 kPa	-	115 nl/min
2	250 μm	5 (S)	-40 kPa	-	90 nl/min
3	250 μm	10 (S)	-40 kPa	-	130 nl/min
4	100 μm	5 (S)	-40 kPa	-	200 nl/min
5	100 μm	10 (S)	-40 kPa	-	290 nl/min
6	100 μm	10 (D)	-20 kPa	-	160 nl/min
7	100 μm	10 (D)	-40 kPa	-	400 nl/min
8	100 μm	6 (G)	-40 kPa	-	500 nl/min
9	100 μm	6 (G)	-40 kPa	10 kPa	670 nl/min
10	100 μm	6 (G)	-60 kPa	-	740 nl/min
11	100 μm	6 (G)	-20 kPa	-	220 nl/min
12	100 μm	6 (G)	-40 kPa	10 kPa	200 nl/min *
13	100 μm	6 (G)	-40 kPa	-	140 nl/min *
14	100 μm	6 (G)	-60 kPa	-	220 nl/min *
15	100 μm	11 (G)	-40 kPa	-	250 nl/min @
16	100 μm	11 (G)	-40 kPa (pre)	-	300 nl/min @

Table 6.1: Experimental results for vents using various vent designs and sizes. Vent areas marked (S) use basic, normal depth control channels at the vent, (D) indicated deeper (1 mm) control channels to allow increased membrane deflection and (G) marks vents using a groove design. * Marks the average flow while covering the vent from no cover to full cover. @ Marks tests filling a complete chamber.

the average fluid displacement from the vent being uncovered to the vent being fully covered (complete alignment). The used architecture is identical to the one used for rows 8, 9 and 10, showing the significant decrease of fluid displacement in relation to the decreasing effective vent area.

6.5.4 Increasing the throughput

Our final prototype contains a chamber that is mostly covered by the vent (19 out of 22 mm chamber length) to provide optimal venting speed. As shown in row 15, the large vent allows for significantly faster average venting than in row 13. To put in perspective, this architecture vented a channel of approximately 2.1 mm^3 in 8 minutes and 30 seconds.

The fluid displacement rate can further be increased if the chamber is “precharged”. Applying vacuum pressure to the vent of a closed metering chamber, containing only air causes the air pressure in the closed flow channel to drop. Opening the input valve of the metering chamber then causes the fluid to rapidly fill part of the metering chamber. Rows 15 and 16 in Table 6.1 show the average fluid displacement for an identical architecture. As a result the same 2.1 mm^3 chamber is vented in 7 minutes, after 2 minutes of precharging.

If faster venting is required, our data in Table 6.1 shows how adjustments of the membrane thickness, vent area and higher pressure differences can decrease the venting duration. In particular we want to point out that our test-setup in Section 6.5.4 uses a channel depth of $120\text{ }\mu\text{m}$ to minimize the impact of fabrication irregularities on the data. Channels can however be fabricated at $10\text{--}20\text{ }\mu\text{m}$ depth [4] [32], allowing to significantly increasing the vent area in relation to the venting volume, thereby speeding up the venting process. Furthermore, PDMS sheets significantly thinner than $100\text{ }\mu\text{m}$ are available [61] which can also drastically increase the throughput of our vents.

6.5.5 Detecting insufficient volumes

As mentioned in Section 6.4 and shown in Fig. 6.6, we use our architectures valve that is placed in front of the metering chamber (i.e. V_2 in Fig. 6.7) as the VEDV instead of a dedicated vent. After aligning an insufficient volume of fluid in the chamber, leaving the chamber

partially empty, we apply pressure to the flow channel. The video “VEDV” in [105] shows proof of concept, as air builds up from the vent and pushing through the water droplet. Accurate detection of such pressure and an adequate feedback loop are left for future work.

6.5.6 Arbitrary ratio mixing using vents

The venting technology and resulting metering and alignment capabilities can benefit several operations. We showcase this by extending a mixer with vents and thereby improving its functionality. Mixing architectures as introduced in Sect. 2.3 present major drawbacks. As shown in a video available here, missing alignment and metering options result in the need to discard fluids. Furthermore does the mixer require additional channels to dispose of this fluid. We therefore extend our use of vents to mixing architectures as shown in Fig. 6.12a. Placing vents at the end of each chamber allows to align fluid to their location instead of pushing fluid into the chamber. Valves 7 and 8 which are located at the same position in a regular mixer usually assure that while filling one chamber, no fluid is pushed into the other. Since our vents align the fluid to a specific point without pressure, these valves are no longer necessary. A video called “MixerTop” demonstrating the alignment process without separating valves is available in the video repository [105].

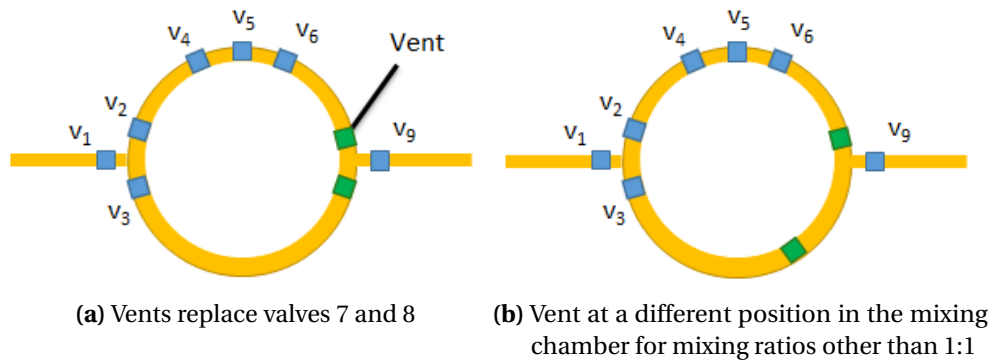


Figure 6.11: Metering using an Alignment Vent

Furthermore, common mixers only allow the mixing of two fluids at a fixed ratio, the result of which having to undergo additional mixing operations to reach other mixing ratios. While the size of one mixing chamber can be adjusted to allow a different ratio to be mixed, each mixer is limited to the ratio its architecture is build for. The introduction of vents allows mixing ratios to differ while keeping a uniform architecture size by placing the vent at a different

position in the chamber as shown in Fig. 6.12b and in video “MixerBottom” in [105] which aligns fluid to a vent, filling the bottom chamber to 3/4. However, each mixing chamber is not limited to a single vent. Placing multiple vents along a chamber as shown in Fig. 6.12a allows a single mixer to achieve several mixing ratios with a single mixing operation. In this example the top chamber can only be filled completely. The bottom chamber can be filled completely, or half for a 2:1 mixing ratio as shown in Fig. 6.12a. In this case, the second part of the chamber can also be filled with a third fluid achieving a 2:1:1 mixing ratio of three fluids. Introducing a small number of vents to a mixer therefore allows for high flexibility in mixing ratios using a single component.

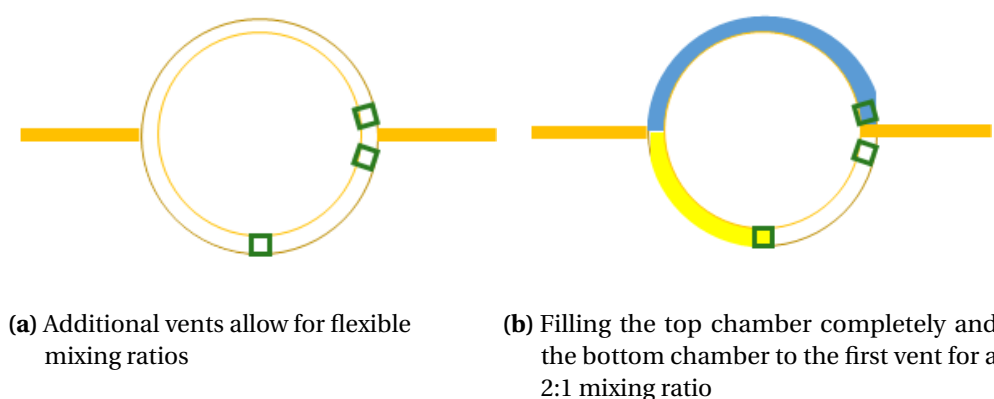


Figure 6.12: Mixer with additional vent

If a mixing operation commences with part of the mixer empty (i.e. filled with air), the mixing process will cause air bubbles to form in the liquid. This trapped air can however be removed using a vent in an identical fashion as a metering channel introduced before.

6.6 Conclusion

We have proposed a novel concept that combines fluid alignment, metering and fluid volume error detection in a single component. Aligning and metering fluids on-chip enables the execution of more complex, multi-operational applications without loss of precision. We have demonstrated through several experiments that venting of volumes commonly used in microfluidics is possible within minutes, using vacuum and / or pressure to drive the air through adjustable vents. It is difficult to put this duration into context as scheduling

papers [24, 77] have so far omitted metering or alignment operations all together (or are considered part of transport operations). Operations such as incubation and mixing are shown to only take seconds, making several minutes long alignment as demonstrated in this paper seem impractical. However, no relation of these durations to original assay protocols can be found where such operations take minutes to tens of minutes and waiting times of hours might be involved [27, 93, 95]. Even if the duration of assays can be significantly shortened, execution times of minutes are more realistic than seconds. For example the duration of DNA amplification has been cut down from several hours to 4 minutes, mostly due to very fast heating cycles enabled by microfluidics [39]. In this experiment 1.7 μl of fluid are used, which if compared to our data in Sect. 6.5.4 can be aligned into a chamber within a similar time frame. We therefore conclude that the required time for alignment and metering is reasonable.

Error detection in FBMBs is so far limited to specific testing phases which are run prior to an application [50]. To the best of our knowledge, the fluid volume error detection feature of our component is the first that could detect erroneous behaviour at runtime for FBMBs. While the current experiments only provide proof of concept for detectable pressure through VEDVs, we are confident that future work can provide suitable detection methods. The combination of these features into a single component minimizes the impact to the design complexity of the chip, allowing to continue the trend of increased integration density in FBMBs.

Chapter 7

Discussion and Conclusions

Microfluidic biochips have the potential to revolutionize both industrial and academic fields by replacing current technologies from bioanalysers in laboratories to advancing point of care treatment and diagnostics. With increasing traction in both research and commercial applications, biochips will quickly evolve to execute more complex tasks while also becoming more commonly available to the public. The contributions in this thesis are meant to help this development and provide basic tools that are necessary to overcome the issues that arise along this way.

Microfluidics is a very interdisciplinary field, and receives an increasing amount of attention from the computer science community, likely due to its similarities to electrical- and computer-engineering. Many common solutions in design, optimization and automation can be applied to biochips to determine their architecture, routing, scheduling and more. However, it is common for researchers and developers in computer science to be among the end-users, or at least share a broad spectrum of the end-users knowledge. The application for the product that is being developed is therefore clear to the developer, making it easier to understand the needs and possible issues. When it comes to microfluidic biochips however, the computer science developer has usually very little understanding of the desired applications. There have already been attempts to start bridging this gap [53], but many publications do not focus on the usability of their contributions. While the targeted issues are legitimate, the number and broadness of assumptions that need to be made bring into questions whether

certain contributions can ever be integrated into realizable systems.

Such assumptions are for example, that fluid movement can happen without pressure sources and unlimited outlets are available for individual pressure sources.

About the application execution time for assays run on biochips: The execution time is often stated as one of the advantages of biochips over conventional methods. Examples such as DNA amplification PCR [39] have shown tremendous decrease in execution time made possible by the small volumes of fluids used and the resulting extremely short heating cycles. Many publications that deal with scheduling, routing or similar topics that closely relate to the duration of the application's operations are assuming similarly short or even shorter execution times for all their operations [14]. However, referenced assay such as Glucose Assays [94], Cholesterol Assays [93] or Proteasome Activity Assays [95] contain operations that require significantly longer durations (i.e. minutes to tens of minutes as stated in their assay protocols) than the mere seconds they are given in a biochip schedule. With no distinct source for these short durations they are likely to be assumptions that are made from experience regarding scheduling as presented in computer science. There, many operations of similar duration are given which makes the outcome of the schedule so important. While extremely fast assay exist that suit such schedules, especially when using very small volumes in biochips, this should not be considered the norm. This reiterates the importance of bringing this interdisciplinary field closer together, so that issues and their parameters are communicated by the actual end user and do not have to be assumed by the developer, and opens many other future work directions.

In this thesis, three mayor contributions, pin-count reduction, volume management and a novel metering component, have been described in detail. Additionally, a design tool has been developed which allowed for the fabrication of prototypes to verify the functionality of proposed techniques. In general, all these contributions aim to make FBMBs easier to use, i.e. the setup of the biochip, and make them more efficient in their use. This allows other contributions to overcome limitations and so far unrealisable assumptions, to take the step from virtual models to physical prototypes.

The presented pin-count reduction technique allows to significantly reduce the number of individual outside pressure sources that are required for a biochip. This in turn reduces the need for bulky and expensive external hardware which stands in direct contrast to key

features of biochips such as their portability. At the same time, this reduces the number of required control-pins and control channels on the chip, reducing the required chip size. In contrast to previous techniques, we do not base our pin-count reduction algorithm on data gathered from scheduling the application to the architecture. Instead only the architecture design is used as input and the application is scheduled onto the pin-count reduced design. This potentially increases the overall execution time of the schedule. However, if an extension to the execution time is feasible for the desired application, the pin-count can be reduced to a significantly higher degree.

Our waste-aware volume management algorithm offers more efficient use of the available mixing technology and reagents, while satisfying all constraints that prevent underflow and overflow. As examples have shown, certain biochemical assays will be significantly more wasteful when applied to a biochip than when using conventional (i.e. pipettes, tubes, etc.) tools. We have determined efficient mixing operations that minimize the volume of waste created. Furthermore, fluid that is leftover from mixing operations, which usually is considered waste, is reused in future mixing operations whenever possible, further reducing the amount of actual waste created. Besides the conventional mixing architecture which only allows for fixed ratio mixing operations, we have also investigated the effects of our volume management on arbitrary ratio mixers and have applied similar optimization techniques to their use. The architectural design and functionality of such arbitrary ratio mixers has then been investigated in detail in our contribution on a novel metering component.

The novel metering component for volume management has shown promising results for various applications. Exploiting the gas permeability of the PDMS used as a membrane in our biochips, allows to expel trapped air, enabling the fabrication of vents in our architectures. The use of vents allows to align and meter fluids, but also to potentially detect insufficient volumes during the metering process, which are crucial features for accurate fluid transport within a biochip. We have shown that the same principle can be extended to other applications such as mixers. To overcome the limitations of fixed ratio mixers, vents can be placed in the mixing chambers, enabling a single components to achieve various mixing ratios of two or more input fluids.

7.1 Outlook and Future Work

While the conclusions of each chapter include some possible future work, several other contributions become possible through the combined use of recent work. Pin-count reduction for example has seen another promising solution in on-chip control [97, Chapter 10]. Similar to our presented technique, on-chip control can significantly reduce the number of required control-pins to operate a biochip. However, instead of additional execution time, on-chip control requires complex systems of flow- and control-channels, valves and layer changes, which besides being challenging to design, can occupy multiple times the original chip size. Finding a suitable combination of both techniques is likely to offer substantial pin-count reduction while keeping the trade-off of each technique at an acceptable degree.

Much of the outlined future work in Chapter 5 is made possible through the contribution in Chapter 6. As shown, arbitrary ratio mixers require fewer mixing operations and can reduce the volume of waste created, compared to fixed ratio mixers. However, other adaptations to the architecture are possible as well using this venting technology. As proposed in Chapter 5, biochips could be built without waste outlets, if all fluid volumes that are introduced to the chip during runtime can be accounted for beforehand. Using venting technology, such waste-storage components could be filled efficiently, leaving the biochip with fewer external hardware.

The venting technology offers promising solutions to issues such as the described alignment and metering of fluids on the chip. The versatility of this technology however offers potential functionality not explored so far. For example, when applying pressure to a vent instead of vacuum, fluids can be moved from instead of to a precise location as shown in the video “ReverseVent” in our video repository [105]. This can offer an additional metering solution as fluid is be split at a precise location or simply provide additional pressure sources for fluid transport. In a similar manner, vents can be used to provide controllable amounts of oxygen or other gases, necessary for certain biochemical reactions.

Appendix A

Biochip Designer Tool and Fabrication

This appendix presents a tool we created to aid in the design of biochip prototypes fabricated for this thesis. Related work has proposed tools that simplify or even automate parts of the biochip design process [104, 119]. This work focuses on the biochip architecture in regard of the target biochemical application and aids in design and functional efficiency of the architecture. Related work for DBMBs also indicates crucial functional necessities and limitations of CAD tools applicable to FBMB design as well. While single steps in the design process can be handled individually and sequentially, it has been shown that prediction and incorporation or “Co-Design” of steps is beneficial [16]. This previous work however misses the link between the biochip design and the biochip fabrication for biochips without design limitations such as the grid design [119]. Furthermore is it not possible to adapt the design according to different technologies (e.g. valves) or fabrication methods which heavily impact the design for example because of precision and spacing limitations. It is therefore necessary to allow interchangeable technology files and component libraries to dictate a large number of design details, leaving the design tool with the tasks of visualization, functional verification and design automation.

The Biochip Designer Tool (BDT) is programmed in QT/C++ and the source files are available here [105]. The main intention of the BDT is to generate G-Code from a graphical biochip design to be manufactured in a micromilling machine or similar (e.g. laser cutter). However, it also provides aid in the design process. (1) In a graphical designer, many low level details such as component design or channel dimensions can be hidden from the user and instead

be determined from component libraries and technology files which have been created by experts and apply to the vast majority of users. (2) Tedious tasks can be automated such as the placement of valves and the connection of control channels to control-pins. (3) Furthermore, any kind of optimization can be integrated into the BDT, such as pin-count reduction. All features are accessible from the GUI as shown in Fig. A.1 and no programming or scripting is necessary for the design process.

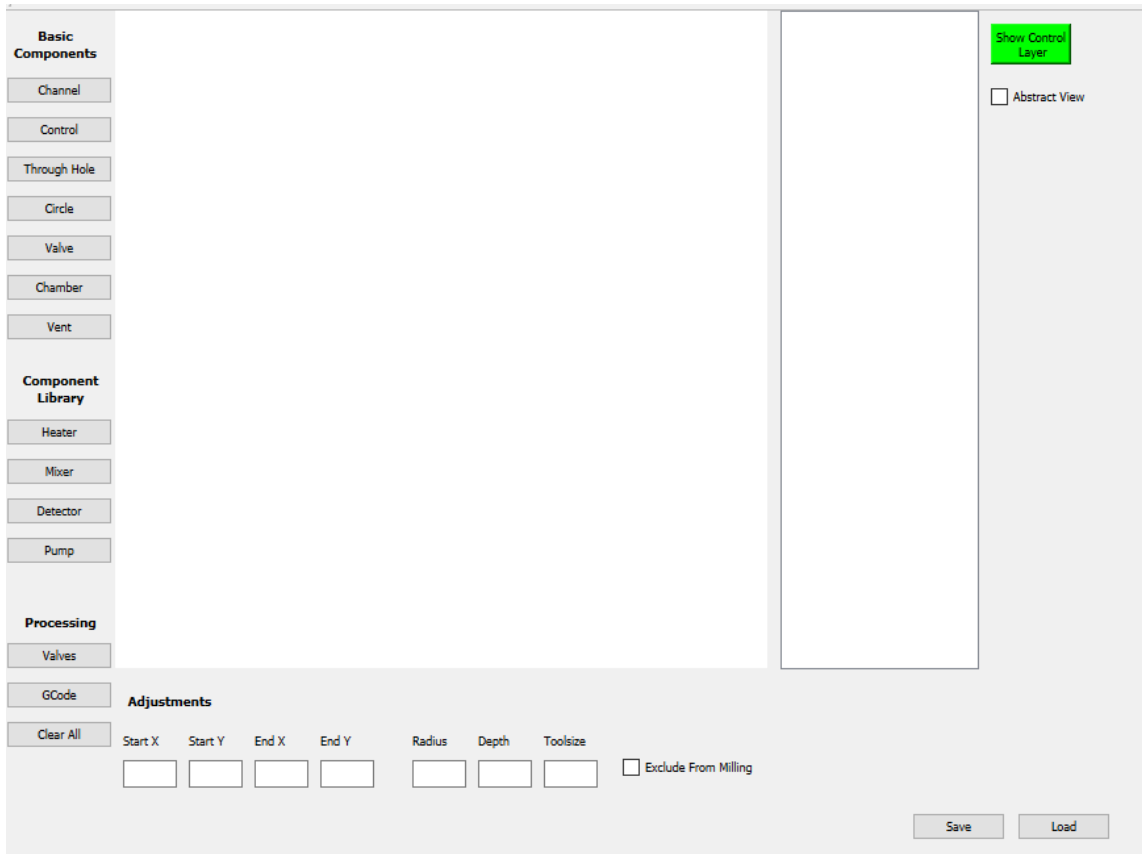


Figure A.1: GUI of the BDT

At the current stage, the BDT covers all design steps except netlist generation of our physical synthesis part of the design process as indicated in Fig. A.2. The user creates a netlist according to the needs of the desired application. The only other part of the Physical-Synthesis that is left to the user is the location of each component on the chip. Future work on the BDT can expand the extent of automation that is offered to the user to earlier steps than netlist generation. Allowing the user to define an application, for example according to the application model introduced in Sect. 2.5 can allow for optimizations regarding the application itself, and lead to automatic netlist generation. An additional step is the automatic generation of

the application graph from an assay. However, as pointed out before in Sect. 3.5, interpreting assay protocols is no simple feat, to a large extent due to ambiguities in the description.

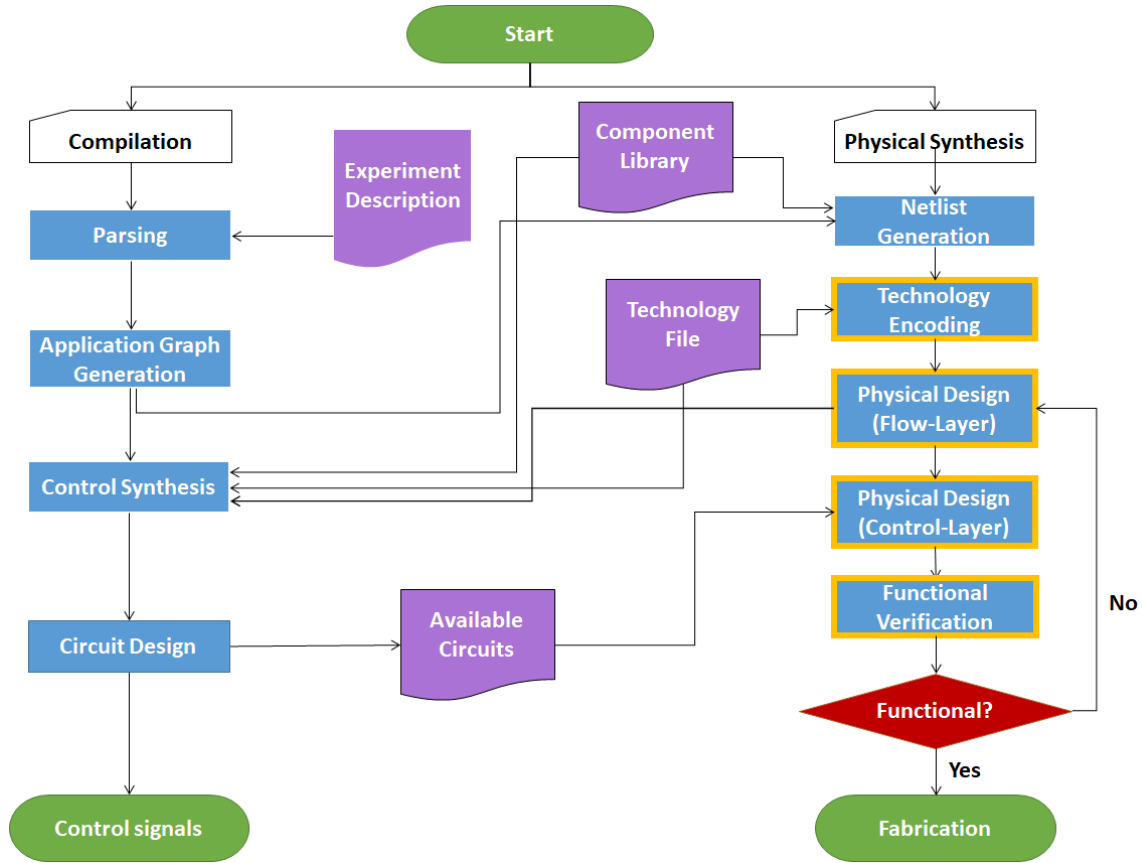


Figure A.2: Biochip design process overview with yellow borders indicating the steps covered by the BDT

The tool provides the option to design the chip at a very basic level, placing channels or other geometric forms manually for both the flow- and control-layer as shown in Fig. A.3. This allows for fast changes in architectures with non standard components or technology. Even in this mode, technical details about the fabrication process are handled automatically such as multi-pass milling for deep channels or the mapping of the design from the design area in the tool, to any desired size of biochip.

If this low level design is not required, component libraries and technology files are used to provide the information that is masked in a more abstract view as shown in Fig. A.5. Here, predefined components are placed into the graphical design area, which only tell the user which functionality they will provide to the architecture. The physical design of the components remains hidden. Fig. A.4 shows a partial example from our component library available

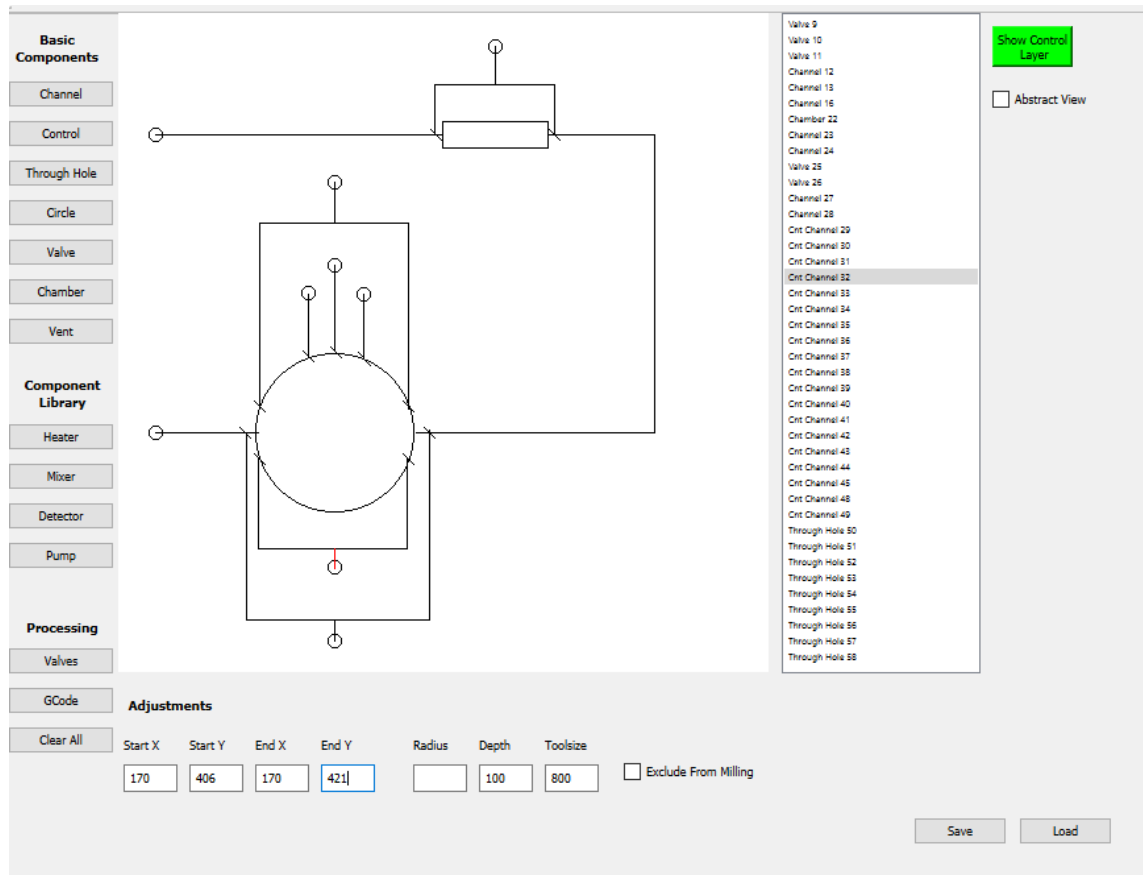


Figure A.3: Low level design view

to the BDT. Components can be designed at a scale that the designer determines sufficient to place the required subcomponents within. The actual dimensions of the component can then later be determined by the BDT, where the component can also be scaled as a whole. All subcomponents used in our component library are basic components used in the BDT such as valves, channels or chambers. This allows for components to also be changed or extended within the BDT, if persistent changes to the component library are unwanted. In either mode, many settings of individual components can be adjusted if necessary such as their depth or used milling tool.

With these features the BDT allows to significantly speed up the design of biochips, especially to untrained users. At the same time, no design details are lost as any higher abstraction can be turned off to gain precise control of every component.

```

{
  "Components": [
    {
      "CompType": 0,
      "CompName": "Mixer",
      "FLConnections": 2,
      "Valves": 9,
      "Parts": [
        {
          "PartType": "Circle",
          "Fill": 1,
          "Diameter": 120,
          "PositionX": 75,
          "PositionY": 75
        },
        {
          "PartType": "Channel",
          "StartPositionX": 0,
          "StartPositionY": 75,
          "EndPositionX": 17,
          "EndPositionY": 75
        },
        {
          "PartType": "Channel",
          "StartPositionX": 137,
          "StartPositionY": 75,
          "EndPositionX": 154,
          "EndPositionY": 75
        },
        {
          "PartType": "Valve",
          "PositionX": 7,
          "PositionY": 75
        }
      ]
    }
  ]
}

```

Figure A.4: Component library used by the BDT

The BDT generates basic, sequential G-Code without macros (see Sect. 3.4.2) as manual editing of the G-Code is not necessary. The generated G-Code is then transferred to the Minitech Machinery Mini-Mill/3 micromilling machine using a software tool called Mach 3 [1]. Using this tool, the micromilling machine can be operated manually, to automated according to a G-Code file.

A.1 Fabrication Example

Our fabrication example begin with an empty project in our BDT. We design a simple architecture that allows us to test the venting functionality from Chapter 6. Since this is a novel component, no corresponding entry is available in the component library, therefore we design the architecture using basic components. The finished design is shown in Fig. A.6, which uses channels on the flow- and control-layer, through holes, valves and a vent. As mentioned before, many technical details can be ignored and left to the provided technology file to fill in. This includes properties such as mill sizes for the used components (e.g. 800 μm

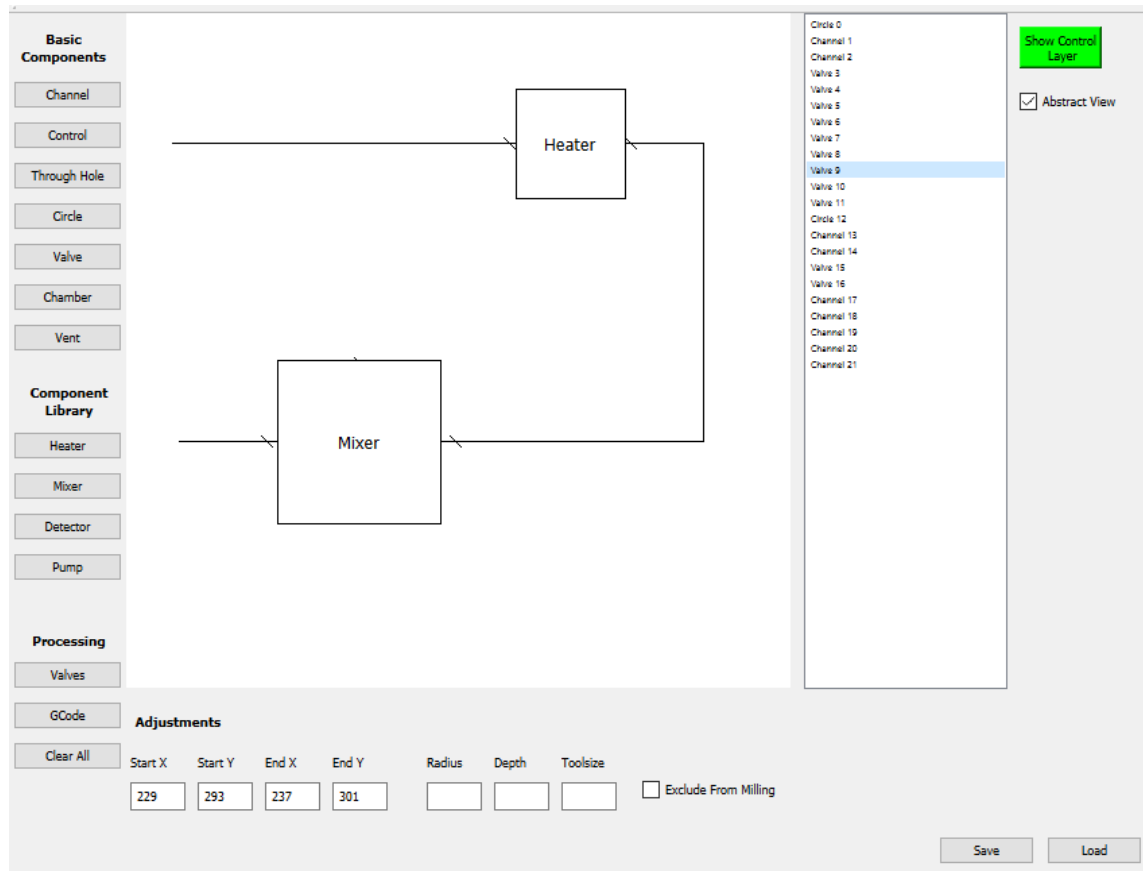


Figure A.5: Abstract view of the architecture design showing a mixer and heater but hiding component details and control connections

for channels, and $200\ \mu\text{m}$ or vents) and channel depth (i.e. $120\ \mu\text{m}$). Furthermore, the sizes of the components only matter relative to each other. The BDT will determine the physical size according to the available space on the PMMA piece.

With the design finished, the BDT generates the G-Code that represents the architecture design. Multiple G-Code files are generated, which splits the G-Code according to what can be milled without interruption. Interruptions in the milling process are necessary whenever the mill, or the PMMA piece has to be changed. For example, our design will generate a G-Code file, as partially shown in Fig. A.7, for the channels and valve displacement chambers on the control-layer, which are milled using the same mill. The G-Code for the vent however is located in another file, as a smaller mill has to be used.

Next, a piece of PMMA of the desired size is secured in the micromilling machine. The mill can be aligned to the PMMA using the Mach 3 micromilling software tool. As discussed before,

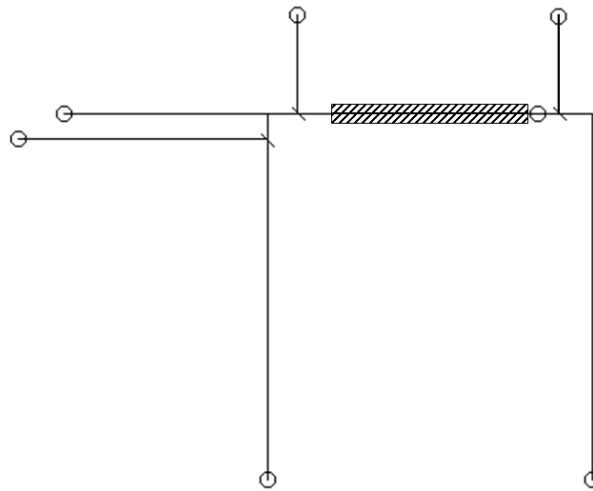


Figure A.6: BDT design for a vent testing architecture

the alignment can be done manually, using a probe (as available on our Minitex Machinery Mini-Mill/3 as shown in Fig. A.8) or using more precise alignment measures. The G-Code files are then loaded into the same software tool. The order in which the G-Code files are executed is irrelevant, as long as the correct mill and PMMA piece (for the flow- and control-layer) are in place. Once both layers are fully milled, a sheet of PDMS is placed on one and the other layer aligned on top. For prototyping with component sizes as stated here, alignment of the layers by hand is sufficiently accurate. The assembled biochip is then bonded in a hydraulic press as shown in Fig. A.9 and explained in Sect. 3.4. After bonding the biochip is ready to use, it is however recommended to let the chip cool to room temperature before use, to assure maximum bonding strength.

A.2 PMMA preparation

We use PMMA pieces which were cut from a sheet using a table saw. As this can damage the edges of the resulting pieces and does not guarantee 90° angles we prepare the PMMA pieces using the micromilling machine. By milling in a rectangular motion through the full depth of the PMMA, we can not only assure precise corners and clean edges but also scale the pieces to the desired chip size. The outer part of the original piece can later be used as a sacrificial piece if required. A sacrificial piece can be useful to optimize the distance of the mill-head to the surface of the PMMA. Optimally, the mill-head should be aligned directly to

```

276 G01 X59.7525 Y56.2409
277 G01 X59.8975 Y56.2409
278 G02 X59.7825 I-0.0575
279 G01 X59.7825 Y55.4359
280 G02 X59.8975 I--0.0575
281 G01 X59.8975 Y56.2409
282 Z1.0
283 G00 X28.3008 Y31.6208 Z1.0
284 G01 Z-0.2 F700
285 X28.3008 Y41.0544 F700
286 Z1.0
287 G00 X25.4848 Y31.9024 Z1.0
288 G01 Z-0.2 F700
289 X25.4848 Y44.2928 F700
290 Z1.0
291 G00 X35.6224 Y8.5296 Z1.0
292 G01 Z-0.2 F700
293 X25.4848 Y8.5296 F700
294 Z1.0
295 G00 X16.0512 Y24.1584 Z1.0
296 G01 Z-0.2 F700
297 X16.0512 Y8.5296 F700
298 Z1.0

```

Figure A.7: Code snippet of the control-layer G-Code used in the example

the surface of the PMMA, resulting in a depth precisely as described in the G-Code. However, the mill-head is often adjusted optically, using the naked eye or a microscope. This can result in the mill-head resting slightly above the PMMA, causing more shallow channels as desired. On the other hand, the mill-head might also be pressed onto the PMMA, causing deeper cuts. Many micromills come with integrated sensors or attached probes which are used to determine the mill-heads Z-axis location. However, even such probes are limited in their precision, and can cause an offset of several μm . To optimize the mill-head location after setting it using optical or automatic methods first, a sacrificial part of the same PMMA piece can be used. The mill is run over a long distance (as far as possible) while slowly descending a predetermined amount. Assuming the mill-head started slightly above the PMMA, the mill will intersect with the PMMA part way through this run as shown in Fig. A.10. The point at which the mill and the PMMA intersected will be clearly visible as the mill starts removing material from this point on. By measuring the distance the mill travelled before intersecting with the PMMA and the total drop in the Z-axis, the distance of the mill in its original position to the PMMA can be calculated with high accuracy. This offset can then be taken into account by the micromilling software program.

Other parts of the milling process need to be setup accurately as well to achieve accurate results. As pointed out in Sect. 3.4, cast PMMA has a thickness tolerance of $\pm 15\%$, which

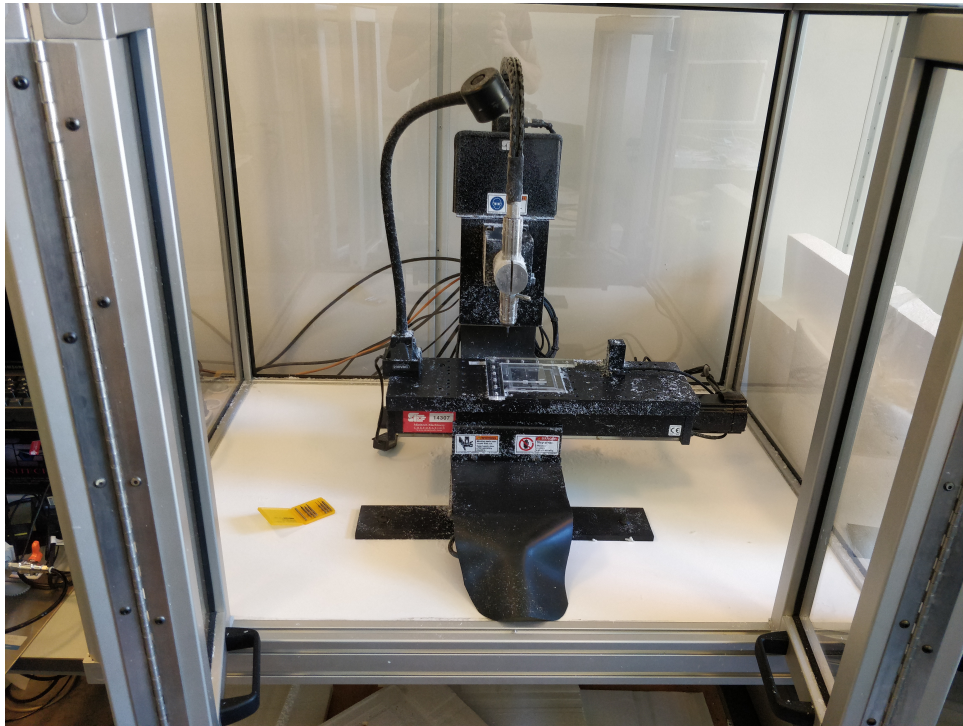


Figure A.8: Minitech Machinery Mini-Mill/3

needs to be corrected using for example a fly cutter. Lastly, the method that holds the PMMA piece in place during the milling process can interfere with depth precision. If the PMMA piece is secured using this squeeze thingy, it has to be assured that not too much pressure is applied, as this can cause the PMMA to bend. If this is not available, double-sided tape has become a popular choice as a quick and easy solution. While tape is usually of uniform thickness, it is important to note that the pressure applied by the mill to the PMMA causes enough force to bend the PMMA downwards, as the PMMA is basically “standing” on the tape, providing empty space between the PMMA and the milling platform. These optimizations do however require a significant amount of time and are likely unnecessary for many experiments. For most prototyping work done for this thesis, the components scale was chosen so that small inaccuracies would not cause a noticeable alteration of experiments. For example, using channel depths of $100\text{ }\mu\text{m}$ and more, a misalignment of the mill-head on the Z-axis by a few μm is inconsequential. Once the design is finalized, a highly accurate version can be fabricated to obtain more accurate data.



Figure A.9: Hydraulic bonding press with heated plates

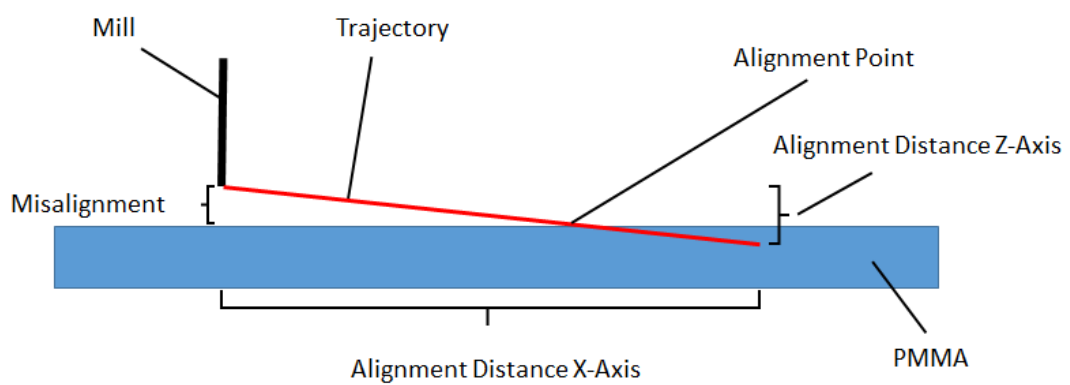


Figure A.10: Aligning the mill's Z-Axis to the PMMA using a sacrificial piece to determine the misalignment

Bibliography

- [1] Newfangled solutions llc. <http://www.machsupport.com/software/mach3/>. Accessed: 14.06.2018.
- [2] *Global Routing*, pages 247–290. Springer US, Boston, MA, 1999.
- [3] M. Alistar and P. Pop. Towards droplet size-aware biochemical application compilation for am-ewod biochips. In *Proc. of Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, 2015.
- [4] Ahmed M. Amin, Raviraj Thakur, Seth Madren, Han-Sheng Chuang, Mithuna Thottethodi, T. N. Vijaykumar, Steven T. Wereley, and Stephen C. Jacobson. Software-programmable continuous-flow multi-purpose lab-on-a-chip. *Microfluidics and Nanofluidics*, 15(5):647–659, 2013.
- [5] Ahmed M. Amin, Mithuna Thottethodi, T. N. Vijaykumar, Steven Wereley, and Stephen C. Jacobson. Automatic volume management for programmable microfluidics. In *Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation*, 2008.
- [6] Ahmed M. Amin Amin et al. Aquacore: A programmable architecture for microfluidics. In *ISCA*, pages 254–265, 2007.
- [7] Vaishnavi Ananthanarayan et al. Biocoder: A programming language for standardizing and automating biology protocols. *Journal of biological Engineering*, 4(13), November 2010.
- [8] I. E. Araci et al. Microfluidic very large scale integration mVLSI with integrated micro-mechanical valves. In *Lab on Chip*, volume 12, pages 1463–1475, 2012.

- [9] I.E. Araci et al. Microfluidic very large-scale integration for biochips: Technology, testing and fault-tolerant design. In *Test Symposium (ETS), 2015 20th IEEE European*, pages 1–8, May 2015.
- [10] Ismail Emre Araci et al. Recent developments in microfluidic large scale integration. *Current opinion in biotechnology*, 25:60–68, 2014.
- [11] Frederick K. Balagaddé, Lingchong You, Carl L. Hansen, Frances H. Arnold, and Stephen R. Quake. Long-term monitoring of bacteria undergoing programmed population control in a microchemostat. *Science*, 309(5731):137–140, 2005.
- [12] Holger Becker, Richard Klemm, Rene Sewart, and Claudia Gaertner. A multiport metering valve technology for on-chip valving. In *The 16th International Conference on Miniaturized Systems for Chemistry and Life Sciences*, page M.1.8, Nov 2012.
- [13] Paolo Bollella, Giovanni Fusco, Cristina Tortolini, Gabriella Sanzò, Gabriele Favero, Lo Gorton, and Riccarda Antiochia. Beyond graphene: Electrochemical sensors and biosensors for biomarkers detection. *Biosensors and Bioelectronics*, 89:152 – 166, 2017. 2D Materials in Biosensors and Bioelectronics.
- [14] K. Chakrabarty. Design, testing, and applications of digital microfluidics-based biochips. In *18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, pages 221–226, Jan 2005.
- [15] K. Chakrabarty and T. Xu. *Digital Microfluidic Biochips: Design Automation and Optimization*. CRC Press, Boca Raton, FL, 2010.
- [16] J. W. Chang, S. H. Yeh, T. W. Huang, and T. Y. Ho. Integrated fluidic-chip co-design methodology for digital microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(2):216–227, Feb 2013.
- [17] Microfluidic Chipshop. Microfluidic chipshop. <http://www.microfluidic-chipshop.com>. Accessed: 23.01.2018.
- [18] H. Chou, M. Unger, and S.R. Quake. A microfabricated rotary pump. *Biomedical Microdevices*, 3, 2001.
- [19] G Code Commands. G code commands. <http://reprap.org/wiki/G-code>. Accessed: 02.04.2018.

- [20] A J Conway, W M Saadi, F L Sinatra, G Kowalski, D Larson, and J Fiering. Dispersion of a nanoliter bolus in microfluidic co-flow. *Journal of Micromechanics and Microengineering*, 24(3):034006, 2014.
- [21] Tormach Fly Cutter. Tormach fly cutter. https://www.tormach.com/product_tts_flycutter.html. Accessed: 27.03.2018.
- [22] Naga Sai Gopi K. Devaraju and Marc A. Unger. Pressure driven digital logic in pdms based microfluidic devices fabricated by multilayer soft lithography. *Lab Chip*, 12:4809–4815, 2012.
- [23] T. A. Dinh, S. Yamashita, and T. Y. Ho. A network-flow-based optimal sample preparation algorithm for digital microfluidic biochips. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 225–230, Jan 2014.
- [24] Trung Anh Dinh et al. A clique-based approach to find binding and scheduling result in flow-based microfluidic biochips. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 199–204. IEEE, 2013.
- [25] Chandra K. Dixit. Biochips based in vitro diagnostics: Market trends and research. *Journal of Biochips & Tissue Chips*, 3(2), 2013.
- [26] Philip Nathanael Duncan. *Design and Miniaturization of Integrated Pneumatic Controls*. PhD thesis, University of California Irvine, 2013.
- [27] Shirit et al. Einav. Discovery of a hepatitis c target and its pharmacological inhibitors by microfluidic affinity analysis. *Nature biotechnology*, 26(9):1019–1027, 2008.
- [28] Elveflow. Elveflow plug and play microfluidics. <https://www.elveflow.com>. Accessed: 23.01.2018.
- [29] Morten Chabert Eskesen, Paul Pop, and Seetal Potluri. Architecture synthesis for cost-constrained fault tolerant flow-based biochips. *Design, Automation and Test in Europe Conference and Exhibition*, 2015.
- [30] Goesmann Fred et al. The mars organic molecule analyzer (moma) instrument: Characterization of organic material in martian sediments. *Astrobiology*, 17:655 – 685, 2017.

- [31] Han-Sheng Chuang et al. Characterizations of gas purge valves for liquid alignment and gas removal in a microfluidic chip. *Journal of Micromechanics and Microengineering*, 22(8), 2012.
- [32] Marc A. Unger et al. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science*, 288(7):113–116, 2000.
- [33] Min Ku Jeon et al. Design and characterization of a passive recycle micromixer. *Journal of Micromechanics and Microengineering*, 15:346, 2004.
- [34] Sudip Roy et al. Waste-aware single-target dilution of a biochemical fluid using digital microfluidic biochips. *Integration, the VLSI Journal*, pages 197–207, 2015.
- [35] Yuyan Shao et al. Graphene based electrochemical sensors and biosensors: A review. *Electroanalysis*, 22(10):1027–1036, 2010.
- [36] G. Firpo, E. Angeli, L. Repetto, and U. Valbusa. Permeability thickness dependence of polydimethylsiloxane (pdms) membranes. *Journal of Membrane Science*, 481:1 – 8, 2015.
- [37] Fluidigm. Fluidigm. <https://www.fluidigm.com/reagents/genomics/bmk-m-96.96-96-96-dynamic-array-ifc-for-gene-expression>. Accessed: 23.01.2018.
- [38] Kanchan Ghosal and Benny D. Freeman. Gas separation using polymer membranes: an overview. *Polymers for Advanced Technologies*, 5(11):673–697, 1994.
- [39] B.C. Giordano, J. Ferrance, S. Swedberg, A.F.R. Hühmer, and J.P. Landers. Polymerase chain reaction in polymeric microchips: Dna amplification in less than 240 seconds. *Analytical Biochemistry*, 291(1):124 – 132, 2001.
- [40] Evonik Performance Materials GmbH. Evonik performance materials gmbh. [https://www.plexiglas-shop.com/DK/en/cutter.htm?\\$product=42z9hcq12jv~p&comeFrom=detail&\\$category=4aql41br8z4](https://www.plexiglas-shop.com/DK/en/cutter.htm?$product=42z9hcq12jv~p&comeFrom=detail&$category=4aql41br8z4). Accessed: 29.11.2017.
- [41] William H Grover, Alison M Skelley, Chung N Liu, Eric T Lagally, and Richard A Mathies. Monolithic membrane valves and diaphragm pumps for practical large-scale integration into glass microfluidic devices. *Sensors and Actuators B: Chemical*, 89(3):315 – 323, 2003.

- [42] W. H. Grover et al. *"Monolithic Membrane Valves and Pumps" chapter in Lab-on-Chip Technology*. Caister Academic Press, 2009.
- [43] William H Grover et al. Development and multiplexed control of latching pneumatic valves using microfluidic logical structures. *Lab on a Chip*, 6(5):623–631, 2006.
- [44] David J. Guckenberger, Theodorus E. de Groot, Alwin M. D. Wan, David J. Beebe, and Edmond W. K. Young. Micromilling: a method for ultra-rapid prototyping of plastic microfluidic devices. *Lab Chip*, 15:2364–2378, 2015.
- [45] Hany Hassanin, A Mohammadkhani, and K.C. Jiang. Fabrication of hybrid nanostructured arrays using a pdms/pdms replication process. 12:4160–7, 08 2012.
- [46] M Hecke and W K Schomburg. Review on micro molding of thermoplastic polymers. *Journal of Micromechanics and Microengineering*, 14(3):R1, 2004.
- [47] Martin Hørslev-Petersen. Computer-aided design for the physical synthesis of biochip control logic. 2016. Master's Thesis, Technical University of Denmark.
- [48] Martin Simonsen Hørslev-Petersen. Computer-aided design for the physical synthesis of biochip control logic. Master's thesis, Technical University of Denmark, 2015.
- [49] K. Hu, B. N. Hsu, A. Madison, K. Chakrabarty, and R. Fair. Fault detection, real-time error recovery, and experimental demonstration for digital microfluidic biochips. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 559–564, March 2013.
- [50] Kai Hu et al. Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration. In *IEEE TCAD, Vol. 33, No. 10, October 2014*, pages 1463–1475, 2014.
- [51] W. L. Huang, A. Gupta, S. Roy, T. Y. Ho, and P. Pop. Fast architecture-level synthesis of fault-tolerant flow-based microfluidic biochips. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1667–1672, March 2017.
- [52] Yongguang Huang, Shibing Liu, Wei Yang, and Chengxin Yu. Surface roughness analysis and improvement of pmma-based microfluidic chip chambers by co2 laser cutting. *Applied Surface Science*, 256(6):1675 – 1678, 2010.

- [53] M. Ibrahim and K. Chakrabarty. Digital-microfluidic biochips for quantitative analysis: Bridging the gap between microfluidics and microbiology. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1787–1792, March 2017.
- [54] Erik C. Jensen et al. Micropneumatic digital logic structures for integrated microdevice computation and control. *IEEE Journal of Micromechanical Systems*, 16(6):1378–1385, 2007.
- [55] Yu Chang Kim Joo H. Kang and Je-Kyun Park. Analysis of pressure-driven air bubble elimination in a microfluidic device. *Lab on a Chip*, (1), 2008.
- [56] H. Öktem, T. Erzurumlu, and H. Kurtaran. Application of response surface methodology in the optimization of cutting conditions for surface roughness. *Journal of Materials Processing Technology*, 170(1):11 – 16, 2005.
- [57] Chia-Yen Lee, Chin-Lung Chang, Yao-Nan Wang, and Lung-Ming Fu. Microfluidic mixing: A review. *International Journal of Molecular Sciences*, 12(5):3263–3287, 2011.
- [58] Libretexts. Van der waals forces. https://chem.libretexts.org/Core/Physical_and_Theoretical_Chemistry/Physical_Properties_of_Matter/Atomic_and_Molecular_Properties/Intermolecular_Forces/Specific_Interactions/Van_Der_Waals_Interactions. Accessed: 19.03.2018.
- [59] C. H. Liu, T. W. Chiang, and J. D. Huang. Reactant minimization in sample preparation on digital microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(9):1429–1440, Sept 2015.
- [60] Carville Ltd. Carville ltd. http://www.carvilleplastics.com/markets-products/micro-fluidics/?gclid=CjwKCAiA4vbSBRBNEiwAMorER809KQ0XVUu95lqyPWesCMhW_4uBA2w9pxfhpuOeR6KbSljzQYeExxoCIBMQAvD_BwE. Accessed: 24.01.2018.
- [61] Silex Silicones LTD. Silex silicones ltd. <http://www.silex.co.uk/shop/ultra-thin-silicone-film/ultra-thin-silicone-film-30-shore-250mm-wide/c-24/c-165/p-13182>. Accessed: 29.11.2017.
- [62] J. Madsen M. Alistar, P. Pop. Redundancy optimization for error recovery in digital microfluidic biochips. *Design Automation for Embedded Systems*, pages 129–159, 2015.

- [63] Minitex Machinery. Minitex machinery. <http://www.minitex.com/milldescrip.php>. Accessed: 27.03.2018.
- [64] Jürgen Mairhofer, Kriemhilt Roppert, and Peter Ertl. Microfluidic systems for pathogen sensing: A review. *Sensors*, 9(6):4804–4823, 2009.
- [65] Abhik Majumder and Sambit Majumder. Effect of corner radius of a t-junction mini-square channel on fluid flow and heat transfer in the developing region: A three dimensional numerical simulation. *Procedia Engineering*, 105:89 – 95, 2015. The 6th BSME International Conference on Thermal Engineering.
- [66] A. Manz, N. Graber, and H.M. Widmer. Miniaturized total chemical analysis systems: A novel concept for chemical sensing. *Sensors and Actuators B: Chemical*, 1(1):244 – 248, 1990.
- [67] Daniel Mark et al. Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications. *CSR*, 39:1153–1182, 2010.
- [68] Markets and Markets Research Private Limited. Markets and markets research private limited. <https://www.marketsandmarkets.com/PressReleases/biochips.asp>. Accessed: 22.01.2018.
- [69] J. McDaniel, A. Baez, B. Crites, A. Tammewar, and P. Brisk. Design and verification tools for continuous fluid flow-based microfluidic devices. In *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 219–224, Jan 2013.
- [70] J. McDaniel, B. Crites, P. Brisk, and W. H. Grover. Flow-layer physical design for microchips based on monolithic membrane valves. *IEEE Design Test*, 32(6):51–59, Dec 2015.
- [71] Jessica Melin and Stephen Quake. Microfluidic large-scale integration: The evolution of design rules for biological automation. *Annual Reviews in Biophysics and Biomolecular Structure*, 36:213–231, 2007.
- [72] Quake SR Merlin J. Microfluidic large-scale integration: the evolution of design rules for biological automation. *Annual Review of Biophysics and Biomolecular Structure*, 36:213 – 231, 2007.
- [73] Giovanni De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1st edition, 1994.

- [74] Darwin Microfluidics. Darwin microfluidics. <https://darwin-microfluidics.com/>. Accessed: 22.03.2018.
- [75] Fluigent Smart Microfluidics. Fluigent smart microfluidics. <https://www.fluigent.com/>. Accessed: 22.03.2018.
- [76] ThinXXS Microtechnology. Thinxxs microtechnology. <http://www.thinxxs.com/>. Accessed: 31.05.2018.
- [77] W. H. Minhass, J. McDaniel, M. Raagaard, P. Brisk, P. Pop, and J. Madsen. Scheduling and fluid routing for flow-based microfluidic laboratories-on-a-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(3):615–628, March 2018.
- [78] W. H. Minhass, P. Pop, J. Madsen, and Tsung-Yi Ho. Control synthesis for the flow-based microfluidic large-scale integration biochips. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 205–212, Jan 2013.
- [79] Wajid Hassan Minhass. *System-Level Modeling and Synthesis Techniques for Flow-Based mVLSI Biochips*. PhD thesis, DTU Compute, 2012.
- [80] Wajid Hassan Minhass, Paul Pop, and Jan Madsen. System-level modeling and synthesis of flow-based microfluidic biochips. In *CASES*, 2011.
- [81] Wajid Hassan Minhass, Paul Pop, and Jan Madsen. *System-Level Modeling and Synthesis Techniques for Flow-Based Microfluidic Very Large Scale Integration Biochips*. PhD thesis, Technical University of Denmark, 2012.
- [82] M. H Minhass et al. System-level modeling and synthesis of flow-based microfluidic biochips. In *CASES*, 2011.
- [83] Yoshinori Mizuno and Mitsuaki Funakoshi. Chaotic mixing caused by an axially periodic steady flow in a partitioned-pipe mixer. *Fluid Dynamics Research*, 35:205, 2004.
- [84] Maria F. Mora, Frank Greer, Amanda M. Stockton, Sherrisse Bryant, and Peter A. Willis. Toward total automation of microfluidics for extraterrestrial in situ analysis. *Analytical Chemistry*, 83(22):8636–8641, 2011. PMID: 21972965.
- [85] Morris Muskat and Milan W. Meres. The flow of heterogeneous fluids through porous media. *Physics*, 7(9):346–363, 1936.

- [86] Transon V. Nguyen, Philip N. Duncan, Siavash Ahrar, and Elliot E. Hui. Semi-autonomous liquid handling via on-chip pneumatic digital logic. *Lab Chip*, 12:3991–3994, 2012.
- [87] TV Nguyen et al. Microfluidic finite state machine for autonomous control of integrated fluid networks. *Proc. of Micro Total Analysis Systems*, pages 741–743, 2011.
- [88] Nazila Norouzi, Heran C. Bhakta, and William H. Grover. Orientation-based control of microfluidics. *PLOS ONE*, 11(3):1–12, 03 2016.
- [89] University of Rhode Island. agilent 2100 bioanalyzer. <https://web.uri.edu/gsc/agilent-2100-bioanalyzer/>.
- [90] Web of Science. Web of science. <http://apps.webofknowledge.com>. Results related to the keyword "microfluidic". Accessed: 22.01.2018.
- [91] European Patent Office. European patent office. <https://www.epo.org/index.html>. Results related to the keyword "microfluidic" in 2017. Accessed: 22.01.2018.
- [92] US Patent and Trademark Office. Us patent and trademark office. <http://patft.uspto.gov/netahtml/PTO/index.html>. Results related to the keyword "microfluidic" in 2017. Accessed: 22.01.2018.
- [93] Abcam plc. Cholesterol assay kit. <http://www.abcam.com/cholesterol-assay-kit-hdl-and-ldlvldl-ab65390.html>. Accessed: 01.03.2018.
- [94] Abcam plc. Glucose assay kit. <http://www.abcam.com/glucose-assay-kit-ab65333-protocols.html>. Accessed: 01.03.2018.
- [95] Abcam plc. Proteasome activity assay kit. <http://www.abcam.com/proteasome-activity-assay-kit-ab107921.html>. Accessed: 01.03.2018.
- [96] P. Pop et al. *Allocation and Schematic Design*. Springer International Publishing, 2016.
- [97] P. Pop et al. *Microfluidic Very Large Scale Integration (VLSI): Modeling, Simulation, Testing, Compilation and Physical Synthesis*. Springer International Publishing, 2016.
- [98] Paul Pop et al. Continuous-flow biochips: Technology, physical design methods and testing. In *IEEE Design and Test of Computers*, 2015.

- [99] Ultimaker 3D Printer. Ultimaker 3d printer. <https://ultimaker.com/en/products/ultimaker-3/specifications>. Accessed: 21.03.2018.
- [100] Induflex Acrylic Processing. Induflex acrylic processing. http://www.pmma.dk/acryl_stobt_kontra_ekstruderet.aspx?Lang=en-GB. Accessed: 27.03.2018.
- [101] AutoCAD Products. Autocad products. <https://www.autodesk.com/products/autocad/overview>. Accessed: 23.01.2018.
- [102] M. L. Raagaard and P. Pop. Pin count-aware biochemical application compilation for mvlsi biochips. In *Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP), 2015 Symposium on*, pages 1–6, April 2015.
- [103] N Rott. Note on the history of the reynolds number. *Annual Review of Fluid Mechanics*, 22(1):1–12, 1990.
- [104] M. F. Schmidt, W. H. Minhass, P. Pop, and J. Madsen. Modeling and simulation framework for flow-based microfluidic biochips. In *2013 Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, pages 1–6, April 2013.
- [105] Alexander Schneider. Additional material. https://drive.google.com/open?id=1ghACB_RsQbeRTdeBVjWiR9nWVDZt3sPA. Accessed: 13.06.2018.
- [106] Mouser Pressure Sensors. Mouser pressure sensors. <https://eu.mouser.com/ProductDetail/Silicon-Microstructures-Inc/SM5652-001-D-3LHR?qs=SaTeGCoDkn17mM8FR1XIjQ==>. Accessed: 16.03.2018.
- [107] Steve C. C. Shih, Garima Goyal, Peter W. Kim, Nicolas Koutsoubelis, Jay D. Keasling, Paul D. Adams, Nathan J. Hillson, and Anup K. Singh. A versatile microfluidic device for automating synthetic biology. *ACS Synthetic Biology*, 4(10):1151–1164, 2015. PMID: 26075958.
- [108] Felix Sima, Jian Xu, Dong Wu, and Koji Sugioka. Ultrafast laser fabrication of functional biochips: New avenues for exploring 3d micro- and nano-environments. *Micromachines*, 8(2), 2017.
- [109] Peder Skafte-Pedersen, Mette Hemmingsen, David Sabourin, Felician Stefan Blaga, Henrik Bruus, and Martin Dufva. A self-contained, programmable microfluidic cell culture system with real-time microscopy access. *Biomedical Microdevices*, 14(2):385–399, Apr 2012.

- [110] F. Su, S. Ozev, and K. Chakrabarty. Concurrent testing of droplet-based microfluidic systems for multiplexed biomedical assays. In *2004 International Conference on Test*, pages 883–892, Oct 2004.
- [111] William Thies, John Paul Urbanski, Todd Thorsen, and Saman Amarasinghe. Abstraction layers for scalable microfluidic biocomputing. *Natural Computing*, 7(2):255–275, Jun 2008.
- [112] William Thies et al. Abstraction layers for scalable microfluidic biocomputing. *Natural Computing*, 7(2):255–275, June 2008.
- [113] Todd Thorsen et al. Microfluidic large-scale integration. *Science*, 298(5593):580–584, 2002.
- [114] Bantam Tools. Bantam tools. <https://www.bantamtools.com/pages/products>. Accessed: 27.03.2018.
- [115] Rokon Uddin, Robert Burger, Marco Donolato, Jeppe Fock, Michael Creagh, Mikkel Fougth Hansen, and Anja Boisen. Lab-on-a-disc agglutination assay for protein detection by optomagnetic readout and optical imaging using nano- and micro-sized magnetic beads. *Biosensors and Bioelectronics*, 85:351 – 357, 2016.
- [116] J.D. Ullman. Np-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3):384 – 393, 1975.
- [117] John Paul Urbanski, William Thies, Christopher Rhodes, Saman Amarasinghe, and Todd Thorsen. Digital microfluidics using soft lithography. *Lab on a Chip*, 6(1):96–104, 2006.
- [118] Rafał Walczak and Krzysztof Adamski. Inkjet 3d printing of microfluidic structures—on the selection of the printer towards printing your own microfluidic chips. *Journal of Micromechanics and Microengineering*, 25(8):085013, 2015.
- [119] Junchao Wang, Philip Brisk, and William H. Grover. Random design of microfluidics. *Lab Chip*, 16:4212–4219, 2016.
- [120] Q. Wang, H. Zou, H. Yao, T. Y. Ho, R. Wille, and Y. Cai. Physical co-design of flow and control layers for flow-based microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(6):1157–1170, June 2018.

- [121] George M. Whitesides. The origins and the future of microfluidics. *Nature*, 442:368 – 373, 2006.
- [122] G Code Wizard. G code wizard. <https://www.cnccookbook.com/g-code-simulator-viewer-generator-gwizard/>. Accessed: 02.04.2018.
- [123] Lung-Jieh Yang. On gas permeation in pdms. *J. Micromech. Microeng.*, 20(11):115, 2010.
- [124] Dae Jin Yun, Tae Il Seo, and Dong Sam Park. Fabrication of biochips with micro fluidic channels by micro end-milling and powder blasting. *Sensors*, 8(2):1308–1320, 2008.