BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

MAKERFLUIDICS: LOW COST MICROFLUIDICS FOR SYNTHETIC BIOLOGY

by

RYAN SILVA

B.S., United States Air Force Academy, 2005
M.S., The Air Force Institute of Technology, 2007

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2017

© 2017 by
RYAN SILVA
All rights reserved Chapter 4 adapted from Silva et al., 2016 Reproduced from Lab Chip, 2016,16, 2730-2741 with permission from the Royal Society of Chemistry.

Approved by

First Reader

Douglas M. Densmore, Ph.D.
Associate Professor of Electrical and Computer Engineering
Associate Professor of Biomedical Engineering

Second Reader

Jason W. Holder, Ph.D.
Principal Member, Technical Staff
The Charles Stark Draper Laboratory, Inc.

Third Reader

Ajay Joshi, Ph.D.
Associate Professor of Electrical and Computer Engineering

Fourth Reader

Michel Kinsy, Ph.D.
Assistant Professor of Electrical and Computer Engineering

Fifth Reader

James Galagan, Ph.D.
Associate Professor of Biomedical Engineering
Boston University, College of Engineering

Associate Professor of Microbiology
Boston University, School of Medicine

Shout out to Jesus

Acknowledgments

I would like to first thank my amazing, beautiful wife who puts up with my nonsense. Shout out to Baby Jay, even though he was a late arrival to this party. When I come home from a long day at the lab and see the smile on his little face, I just know he's about to jab me with something.

There is no better PhD advisor than Douglas Densmore. He taught me the one little word that will get me through academic life and beyond: *tranquillo*. Like me, he understands that sometimes in life you only get to order one *primi* [sic]. In all seriousness, I have yet to describe my PhD experience to anyone without inducing some serious PI envy. Thank you for everything and I will cherish the days you allowed me to spend in your lab. Thanks to the Fluigi crew (Josh, Krishna, Ali) and Prashant for always getting me home safe and on-time.

Special thanks to Jason Holder, who pushed me to develop a thesis of which I can be proud. Thank you for letting me be a part of your lab and your team. What can I say about the Acoustic Boyz? Thanks to Parker Dow for teaching me everything I know; thanks to Charlie Lissandrello for the dusty lemonade; apologies to Ryan Dubay, I blame it on the economy; thanks to Ken "Doctor" Kotz for your curiosity in my project by asking me every day if it's working; thanks to Peter, David, Chris and Helen for granting me four square feet in the productivity hub known simply as "The Pitt"; thanks to Jac, Sarah, and Sammy G. for staying up until 4am on a work night; thanks to Jason Fiering for allowing me to distract his best workers.

Shout out to my funding agency and my employer, the United States Air Force, where, maybe, just once, someone will call me "Sir" without adding, "You're making a scene."

MAKERFLUIDICS: LOW COST MICROFLUIDICS FOR SYNTHETIC BIOLOGY

RYAN SILVA

Boston University, College of Engineering, 2017

Major Professors: Douglas M. Densmore, Ph.D.
Associate Professor of Electrical and Computer Engineering
Associate Professor of Biomedical Engineering

Jason W. Holder, Ph.D.
Principal Member, Technical Staff
The Charles Stark Draper Laboratory, Inc.

ABSTRACT

Recent advancements in multilayer, multicellular, genetic logic circuits often rely on manual intervention throughout the computation cycle and orthogonal signals for each chemical "wire". These constraints can prevent genetic circuits from scaling. Microfluidic devices can be used to mitigate these constraints. However, continuous-flow microfluidics are largely designed through artisanal processes involving hand-drawing features and accomplishing design rule checks visually: processes that are also inextensible. Additionally, continuous-flow microfluidic routing is only a consideration during chip design and, once built, the routing structure becomes "frozen in silicon," or for many microfluidic chips "frozen in polydimethylsiloxane (PDMS)"; any changes to fluid routing often require an entirely new device and control infrastructure. The cost of fabricating and controlling a new device is high in terms of time and money; attempts to reduce one cost measure are, generally, paid through increases in the

other.

This work has three main thrusts: to create a microfluidic fabrication framework, called MakerFluidics, that lowers the barrier to entry for designing and fabricating microfluidics in a manner amenable to automation (Chapter 3); to prove this methodology can design, fabricate, and control complex and novel microfluidic devices (Chapter 4); and to demonstrate the methodology can be used to solve biologically-relevant problems (Chapter 5).

Utilizing accessible technologies, rapid prototyping, and scalable design practices, the MakerFluidics framework has demonstrated its ability to design, fabricate and control novel, complex and scalable microfludic devices. This was proven through the development of a reconfigurable, continuous-flow routing fabric driven by a modular, scalable primitive called a transposer. In addition to creating complex microfluidic networks, MakerFluidics was deployed in support of cutting-edge, application-focused research at the Charles Stark Draper Laboratory. Informed by a design of experiments approach using the parametric rapid prototyping capabilities made possible by MakerFluidics, a plastic blood–bacteria separation device was optimized, demonstrating that the new device geometry can separate bacteria from blood while operating at 275% greater flow rate as well as reduce the power requirement by 82% for equivalent separation performance when compared to the state of the art.

Ultimately, MakerFluidics demonstrated the ability to design, fabricate, and control complex and practical microfluidic devices while lowering the barrier to entry to continuous-flow microfluidics, thus democratizing cutting edge technology beyond a handful of well-resourced and specialized labs.

Contents

List of Tables

List of Figures

List of Abbreviations

ASIC	.....	Application Specific Integrated Circuit
CAD	.....	Computer-Aided Design
CAM	.....	Computer-Aided Manufacturing
EDA	.....	Electronic Design Automation
FPGA	.....	Field Programmable Gate Array
GUI	.....	Graphical User Interface
IDAST	.....	Identification and Antibiotic Susceptibility Test
ISA	.....	Instruction Set Architecture
JSON	.....	JavaScript Object Notation
mLSI	.....	Microfluidic Large Scale Integration
PCB	.....	Printed Circuit Board
PDMS	.....	Polydimethylsiloxane
RBC	.....	Red Blood Cell
UV	.....	Ultra-Violet
VLSI	.....	Very Large Scale Integration
WBC	.....	White Blood Cell

Chapter 1

Introduction

Cellular computation is necessary for the orchestration of large biological systems. Efforts to design, build and test genetic computers have existed since the introduction of the repressilator over a decade ago (Elowitz and Leibler, 2000). Yet, literature reveals that even the most recent efforts in biological computation are limited to two and three-layer logical operations: a paltry number when compared to the relative state of electrical or mechanical computation (Nielsen et al., 2016). Efforts to scale these systems can be stymied by cellular complexity and the sheer amount of time and human effort involved in conducting even the simplest of computational experiments. It is estimated that the development of artemisinic acid, a precursor to the malaria drug artemisinin, required over 150 person-years of work (Kwok, 2010). This lengthy design-build-test cycle can be attributed to time spent tuning the complex metabolic pathways of *Saccharomyces cerevisiae* to accept the new demands of producing the precursor (Ro et al., 2006). Rather than adding layers of complexity to an already complex system, such as the metabolic pathways of *Saccharomyces cerevisiae*, this work began with the desire to scale genetic computers by creating different, simple genetic circuits and then distributing them among many different cells. This effectively creates a distributed network of elementary genetic computers that can then use traditional biological processes, such as quorum sensing, to communicate (Tamsir et al., 2011). The question then became how best to facilitate communications between these cells.

Microfluidics, by definition, facilitate the movement of small amounts of fluid across a device. This movement of fluids can be in the form of, among other methods, a continuous flow, encapsulated droplets (Teh et al., 2008), or electrowetted microdroplets (Kim, 2001). Microfluidic devices can be used to dramatically scale the number of experiments in an automated fashion while reducing reagent costs. Additionally, microfluidic devices can be fabricated to meet certain specifications that allow for greater experimental reproducibility. Yet, microfluidics are not widely adopted by labs that regularly conduct experiments for which the benefits of microfluidic technology seem best suited (e.g., biology, chemistry, physics, etc.) (Whitesides, 2006).

One possible explanation for the lack of widespread adoption of microfluidic technologies is that microfluidics are difficult to both design and fabricate (Whitesides, 2006). Literature reveals that microfluidic devices at chip densities belonging to the classes of large scale integration (LSI), i.e., chips with hundreds to thousands of components, are drawn by hand in a graphics program, such as Adobe Illustrator, or using 3D modeling software, such as AutoCAD (Araci and Brisk, 2014). Any tweaks to common parameters (e.g., channel width, feature spacing, etc.) can necessitate a complete redrawing of large sections, or the entirety, of the chip. This can be mitigated using a parametric design interface, such as the one presented in Chapter 3.4.

Once a microfluidic design is complete, microfluidic devices must then be fabricated, the traditional process for which, namely soft lithography, resembles that of fabricating silicon microelectronics (Anderson et al., 2000). Soft lithography is a process that requires access to a clean room, in addition to hundreds of thousands of dollars in specialized equipment (Xia and Whitesides, 1998). Factoring in the costs of maintenance, training, and personnel, it is easy to see that the barrier to entry into

the field of microfluidics is high, which can explain the lack of widespread adoption of microfluidic technology. Reducing the barrier to entry into microfludic fabrication is a major thrust of this research, the methods for which are outlined in Chapter 3.5.

Utilizing accessible technologies, rapid prototyping, and scalable design practices, the microfluidic fabrication framework developed during the course of this research, titled MakerFluidics, demonstrated its ability to do the following:

1. Design, fabricate, and control complex and novel microfluidic devices (Chapter 4).

2. Solve biologically relevant problems in industry (Chapter 5).

A framework for manufacturing and controlling microfluidics should have the ability to scale in complexity. MakerFluidics proved its ability to do so through the development of a reconfigurable, continuous-flow routing fabric driven by a modular, scalable primitive called a transposer. The full specification of the transposer-based routing fabric, including the design, control, and applications thereof are provided in Chapter 4.

Democratizing microfluidic technologies is useful only insofar as the technology is experimentally (e.g., biologically) relevant, as this was the initial catalyst for pursuing microfluidic technology during the course of this research. Therefore, in addition to creating complex microfluidic networks, MakerFluidics was deployed in support of cutting-edge, application-focused research at Draper Laboratory. Informed by a design of experiments approach using the parametric rapid prototyping capabilities made possible by MakerFluidics, a plastic blood–bacteria separation device was optimized, demonstrating that the new device geometry can separate bacteria from blood while operating at 275% greater flow rate as well as reduce the power requirement by 82% for equivalent separation performance when compared to the state of the art.

These results are fully presented in Chapter 5.

Chapter 2

Background

This work was motivated by the desire to scale microfluidic devices using automation techniques derived from the evolution of microelectronics. The emergence of microfluidic large scale integration (mLSI) resulted in the formulation of methods to manage complexity in design and fabrication. These methods often draw upon analogies with design and computation using microelectronics (Minhass et al., 2013). Unfortunately, these efforts can often be disjointed — automated design methods rely on manually-intensive fabrication efforts, or vice versa. What results from a unification of these efforts is a design-to-device workflow made possible by a new microfluidic fabrication framework, MakerFluidics, presented in Chapter 3. This framework better utilizes automation via computer-aided manufacturing (CAM) at a lower cost in both time and money when compared to the traditional fabrication framework, namely soft lithography, the background for which is provided in Section 2.1.

MakerFluidics was proven viable by designing, fabricating, and controlling a system of novel microfluidic primitives aimed at providing an element of programmability in continuous-flow microfluidic devices, a problem better motivated in Section 2.2.

Finally, the MakerFluidics framework sought immediate experimental relevance by partnering with industry to push the state of the art in bacterial identification and antimicrobial susceptibility testing (IDAST). A solution to this problem would delay the post-antibiotic era (Alanis, 2005) and save tens of thousands of lives per year (Beaglehole et al., 2004). MakerFluidics was responsible for optimizing the sample

purification process, which involves separating bacteria from blood in a microfluidic chip using acoustic manipulation. The purified bacterial sample could then be used in a downstream assay where pathogen IDAST is performed. Background information for the IDAST system is provided in Section 2.3.

2.1 Microfluidic Fabrication Using Multi-Layer Soft Lithography

Soft lithography was adopted as the traditional method of fabricating large-scale microfluidics based on the demonstrated success of photolithography in fabricating microelectronics (Whitesides, 2006). A typical device design utilizes two layers: flow and control. The flow layer is the layer through which fluids (i.e., samples of interest) flow. The control layer is responsible for controlling fluids on the flow layer via pneumatic valving (Unger et al., 2000). Each layer requires its own replica mold. A detailed description of microfluidic fabrication using multi-layer soft lithography is provided in a number of reviews (Duffy et al., 1998)(Sia and Whitesides, 2003)(Weibel et al., 2007). A brief overview of the three main steps to this process are outlined below.

2.1.1 Replica Mold Fabrication

A light-sensitive material called a photoresist is applied to the surface of a silicon wafer using a spin coater. The wafer is then baked in an oven to remove solvents and bubbles from the photoresist that may have been introduced during application. The device design is printed onto an overhead transparency, which is then laid upon the coated silicon wafer. The stack containing the silicon wafer, photoresist, and transparency are exposed to UV light using a piece of equipment called a mask aligner. This process must take place within the confines of a cleanroom as stray UV light

or physical particles (e.g., lint, dust, etc.) can introduce erroneous features into the design. Upon completion of UV exposure the mask is placed in a developing agent, after which the silicon is washed to produce the replica mold.

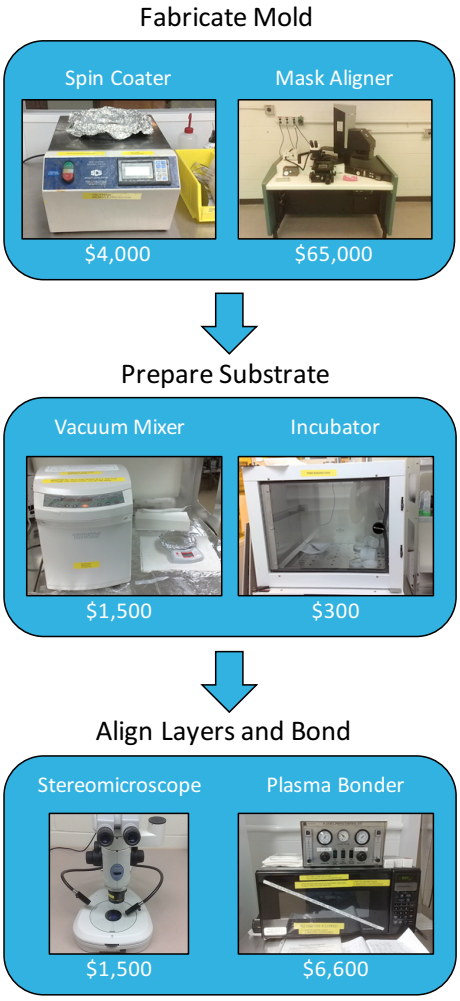


Figure 2·1: Microfluidic fabrication via multi-layer soft lithography requires about $80,000 in specialized equipment, not accounting for infrastructure, personnel, maintenance, or training. Cost estimates based on company quotes.

2.1.2 Preparation of Substrate

The substrate of choice for multi-layer soft lithography is a silicon elastomer known as polydimethylsiloxane (PDMS). This material is the result of combining a cross-linker and gel in a vacuum mixer, ensuring that no bubbles are present in the mixture. The flow layer is produced by carefully spin coating PDMS upon the silicon wafer ensuring that the features are fully immersed by the uncured elastomer. Excess elastomer is spun on top of the immersed features, which will result in the formation of a membrane used by the control layer to valve the flow of fluids. Unlike the flow layer, the control layer does not require the formation of a membrane of specified thickness, thus less precision is required for PDMS-coating the control layer's silicon mold. After coating, the PDMS for each layer is cured in an incubator.

2.1.3 Alignment of Layers and Bonding

After removing the PDMS from the silicon mold, the flow and control layers must be bonded together. This is often accomplished by modifying the surface chemistry of the PDMS using a plasma treatment. Each layer is placed, bonding-side-up in a plasma bonder. After removing the PDMS layers from the plasma bonder, the flow layer must be perfectly aligned with the control layer as the pneumatic valves must overlay their corresponding fluidic channels. This is accomplished by hand under a microscope and must be completed within a finite amount of time (often minutes) before the plasma treatment wears off. This process is then repeated to seal the fluidic layer to a piece of glass.

2.2 Programmability and the Microfluidic–Microelectronic Analogy

As illustrated in Section 2.1, typical microfluidic fabrication uses photolithography, a complex and expensive process used in fabricating silicon microelectronics. It is difficult to imagine a computer engineer having to create a very large scale integration (VLSI) device layout and fabricate an application specific integrated circuit (ASIC) every time they wanted to run a C program. While there will always be a place for custom circuit design in the world of digital electronics one basic tenant of digital design, namely abstraction, requires that it remain very much the exception rather than the rule. Unfortunately for the experimentalist, this is not the case in the field of microfluidics. The creation of custom, "one-off", designs for individual microfluidic experiments, no matter how user-friendly the corresponding CAD software is, could be is what is keeping the productivity of mLSI chips from achieving that of its silicon counterparts. Since Thorsen *et al.* successfully integrated thousands of micromechanical valves in 2002 (Thorsen et al., 2002), academic researchers have attempted to manage exponentially greater complexity in microfluidic design via the introduction of new design methodologies that attempt to introduce "top-down" specificity and move away from a "bottom-up" design philosophy (Minhass et al., 2013)(Melin and Quake, 2007)(Minhass et al., 2012) yet microfluidic experimentalists still find themselves in front of an oven baking a photoresist until it ceases to be sticky.

Managing complexity is a necessary craft in that it allows the engineer to design complicated systems without becoming overwhelmed by details. The art of managing complexity in digital electronics design is a mature process relative to that found in microfluidics. This is evident by the existence of larger scales of integration *in silico* and by microfluidic efforts to create tools that mirror the design-to-execution workflow found in electronic computing. Examples of such tools are Micado, for automation

of control layer routing (Amin et al., 2009), Fluigi Place and Route (Huang and Densmore, 2014a), and BioStream (Thies et al., 2008).

One goal of microfluidic technology is experimental automation. It could be argued that in order to achieve acceptable levels of automation then the expensive and time-consuming fabrication step should be removed from the work flow for the majority case as it is for electronic computation. Often, academic papers delving into the relm of mLSI begin by presenting an analogy between microfluidic LSI and LSI found in digital electronics. This research analogy seems strange as users of digital electronics can be productive using programmable tools (e.g., personal computers, Field Programmable Gate Arrays (FPGA), etc.) without having to know how to wash chemical from printed circuit boards (PCBs), use electronic design automation (EDA) tools to layout ASICs, or enlist the aid of experts in fabrication who can. Thus, the *in silico* analogy does not hold when applied to the common-use case: that of the individual scientist or engineer, as the microfluidic experimentalist cannot adequately abstract away the details of device fabrication.

Abstraction works by placing the user at only the highest level relevant to the computation being performed and masking all underlying details. It can, therefore, be contended that functionally complete automation of microfluidic experiments carries the implication that experimentalists should be placed at only the highest levels of abstraction, thereby masking all underlying details. Currently, even the best efforts in microfluidic tools only remove intermediate levels of abstraction in device design (e.g., place and route), while exposing the scientist to the highest (e.g., device topology and function) **and lowest** (e.g., spin rate for optimal photoresist coverage) levels. Imagine if the only output of a C program were a circuit schematic that must first be built and tested in order to obtain the result of the program.

"Working levels" of complexity imply that the person operating within that ab-

straction layer need not concern themselves with the details of a lower layer, as such requirements would ultimately defeat the purpose of abstraction. Lower levels of detail are said to be "abstracted" away when their use is considered automatic. However, designers of systems residing in one particular abstraction layer should have an understanding of how their design decisions affect the layers immediately above and below the working layer, such as a C programmer understanding the nature of an address space (Harris and Harris, 2013). Theis *et al* advocate for the creation of abstraction layers in microfluidics similar to that found in electronic computing (Thies et al., 2008). These layers achieve success by focusing on three basic fluidic operations: mixing, transport and storage. Their BioStream protocol is a good first step in decoupling microfluidic architecture from biological computation by providing a common language for describing an experimental protocol. BioStream served initially as a standard language for reporting biological protocols but expanded to an end-to-end system that effectively describes biological protocols within the BioStream Fluidic Instruction Set Architecture (ISA) and executes them at the hardware level independent of microfluidic chip microarchitecture (Thies et al., 2008). BioStream, however, does not fully address a functional purpose of abstraction, which is to provide automation, but it does accomplish a very important step the authors describe as a "division of labor" between the biology and microfluidic experts.

There is, and probably always will be, a place in digital electronics for PCB design and ASIC fabrication. However, before an engineer decides to begin the process of building a custom PCB or layout a new ASIC they should first consider how their deisgn decision addresses the productivity gap. In order to proceed, a working definition of the term "productivity" must be presented. Process and requirements engineers (Damian and Chisan, 2006) have defined productivity strictly in terms of hours saved (Lauesen and Vinter, 2001), as a function of on-time delivery (Wohlwend

and Rosenbaum, 1993) or as some measure of quality (Herbsleb and Goldenson, 1996). This section will define the productivity of a particular method as the number of hours saved through the implementation of a particular process.

Device fabrication is not a task oft performed by a computer scientist. Rather, a computer scientist spends many hours debugging a program such that it runs reliably and correctly within the confines of a particular ISA. This exemplifies the nature of design discipline. It is well-within the realm of possibility for a computer engineer to give up debugging a program and reach for a CAD tool, with which to build a custom chip designed for their particular purposes. That scenario would only make sense if the final custom-fabricated solution could overcome the extremely large gap in productivity inherent to designing and fabricating it. The amount of lead-time required to design and build an ASIC or PCB could significantly outweigh the benefits of having a single custom-chip to use only in very specific circumstances and only within that one engineer's lab. Why then is this practice deemed acceptable in microfluidics?

Even attempts to create some framework for flow-based microfluidic design, such as a common microfluidic ISA (Amin et al., 2009) or predefined software modules (Soe et al., 2013) are still, fundamentally, design methods requiring chip fabrication. Microfluidic chip fabrication is a highly unproductive task in that it requires many hours to design and build a device incapable of performing diverse and, often, repeatable experiments. Fortunately for the computer engineer there exists other prototyping options besides PCBs and ASICs, such as the use of an FPGA or a microcontroller. The microfluidic experimentalist is left with only one prototyping option that almost always requires some level of device fabrication.

Chapter 4 outlines a novel primitive for continuous-flow microfluidics aimed at unlocking elements of programmability in continuous-flow devices. It analyzes and

solves microfluidic design challenges using tools for automated design and control, ultimately providing a solution that would allow the microfluidic experimentalist to be one step closer to realizing the productivity potential of true large-scale integration.

2.3 Pathogen Identification and Antibiotic Susceptibility Testing

The World Health Organization listed infectious disease as the second leading cause of death worldwide (Beaglehole et al., 2004). One explanation as to why infections are particularly deadly is that multiple days of lab processing are currently required to determine a particular bacterial identity and its antibiotic susceptibility profile. In the interim, infections are often treated using broad-spectrum antibiotics, which are recurrently ineffective and contribute to the rise of antibiotic resistance and diminished clinical efficacy (Laxminarayan et al., 2013). The ultimate goal of the IDAST program at Draper Laboratory is to determine information regarding a pathogen's phenotype and antibiotic susceptibility at the point of care within the time-frame of a single visit to the doctor. This is performed using a multi-staged approach shown in Figure 2·2, whereby a microfluidic system purifies a bacterial sample from whole blood in preparation for a downstream biological assay resulting in an optical readout.

The microfluidic system is comprised of the three stages shown in Figure 2·3. The separation step is responsible for removing the large blood components such as red blood cells (RBCs) and white blood cells (WBCs) from a diluted whole-blood sample, as these large mammalian blood components can confound the assay performed by the synthetic biological system. It then uses a trapping stage to concentrate the bacterial sample and wash away smaller blood components (e.g., plasma, platelets, etc.). Finally, clean bacterial samples are enumerated and distributed into multiple wells for downstream processing using synthetic biology. The microfluidic leverages

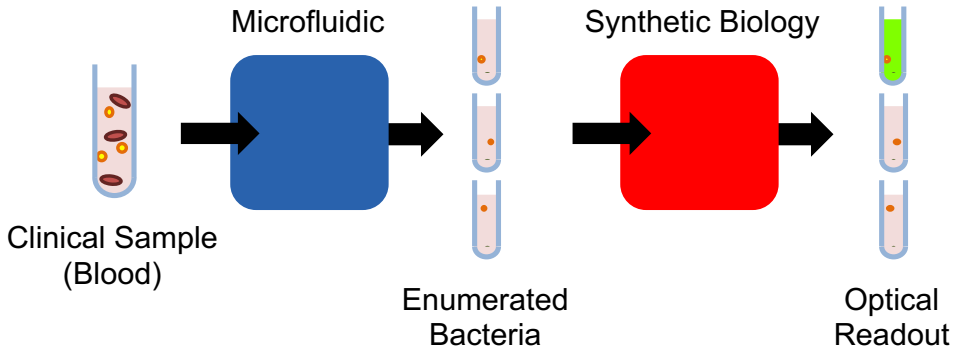


Figure 2·2: IDAST is performed using a microfluidic device for sample processing followed by a proprietary synthetic-biological approach resulting in an optical readout.

acoustic manipulation to accomplish the first two stages, the first of which is the subsystem MakerFluidics sought to optimize, the results of which are presented in Chapter 5. Acoustic manipulation is described in further detail in Section 5.1.

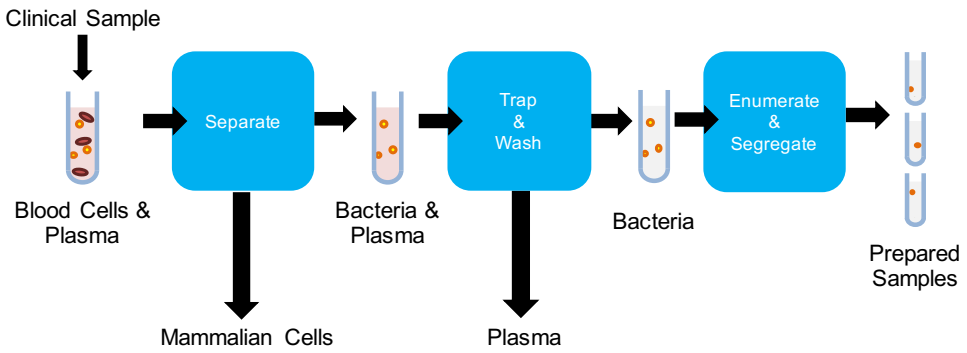


Figure 2·3: Sample processing for IDAST can occur in a microfluidic composed of three stages: separation, trap and wash, and enumerate and segregate. MakerFluidics was responsible for optimizing the first subsystem, namely separation.

Chapter 3

The MakerFluidics Framework

3.1 Introduction

MakerFluidics is a microfluidic fabrication and control framework that operates within a set of constraints guided by the ideals of automation and the modern "maker movement". This framework integrates into the larger microfluidic design flow shown in Figure 3·1, but it can also be employed on its own. MakerFluidics accepts physical and temporal valve control requirements and a microfluidic device design as inputs and provides the necessary control software, manufacturing files, and assembly information required to build a self-contained microfluidic device and control infrastructure.

3.2 Problem Statement

A significant aim of the modern maker movement is to make technology and technological "know-how" accessible to the masses. One way to accomplish this goal is to devise solutions using resources that are flexible and ubiquitous (Schrock, 2014). A significant criticism often levied against the field of microfluidics is its high barrier to entry often as a result of the need for highly specialized fabrication and control equipment as well as expertise that is typically found only in labs dedicated to microfabrication (Whitesides, 2006). This work seeks to create a design-to-device, automated microfluidic work-flow constrained by the ideals espoused in maker culture.

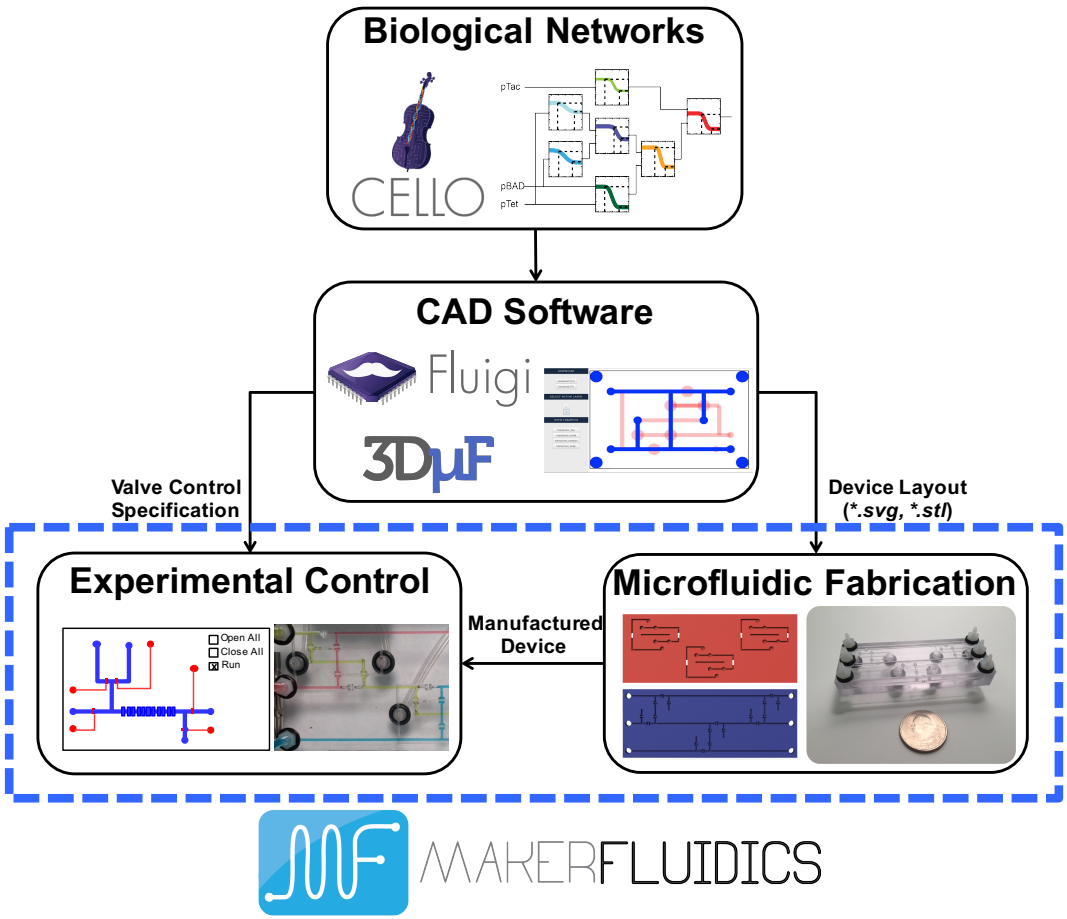


Figure 3·1: MakerFluidics is part of a larger microfluidic design flow that spans from biological specification to microfluidic fabrication and control

3.3 Contstraints

An important goal of the maker movement is to increase technology's accessibility. This ideal is levied on MakerFluidics in the form of a series of constraints, the first of which is that all fabrication equipment must be sourced through ubiquitous consumer and retail product outlets such as Amazon.com in the Unites States, or Amazon.ca, .co.uk, .jp, etc. internationally, and each individual piece of equipment required for fabrication and control must cost less than $100. A desktop computer numerical

control (CNC) mill and 3D printer are excepted from this constraint on the basis that they are common maker-space tools with a wide variety of uses extending well beyond the field of microfluidics; the cost of the CNC mills (Othermill V2 by Othermachine Co.; Othermill Pro by Othermachine Co.; and Nomad 883 by Carbide 3D) and 3D printer (Ultimaker 2, Ultimaker B.V.) used during the course of this research are each less than $3,100. Additionally, all elements of the complete software tool chain must be free and/or open-source.

To further facilitate microfluidic accessibility, all fabrication and control protocols must be accomplished without specialized infrastructure beyond a wall electrical outlet. This excludes fume hoods, clean room facilities, tank storage, vacuum lines and corona/plasma bonders. The process for designing, fabricating and controlling programmable (i.e., valved) microfluidics within the specified constraints is explored. Each stage of the process includes a comparison of current methods to methods developed or adopted by the MakerFluidics framework.

Table 3.1: Necessary infrastructure to perform soft lithography versus that required for MakerFluidics.

Infrastructure	MakerFluidics	Soft Lithography
Clean Room	Not Required	Required
Fume Hood	Not Required	Required
Vacuum Line	Not Required	Required
Tank Storage	Not Required	Required
Electrical Outlet	Required	Required
Equipment Overhead	$4,000	$80,000

3.4 Microfluidic Design

Three levels of software are required to design and fabricate a novel chip geometry using a CNC micromill: a computer aided design (CAD) tool is used to create a solid model of the device; computer aided manufacturing (CAM) software generates the commands (also called toolpaths) that are sent directly to the micromill; and control

software manages the connection between a computer and the micromill and sends individual toolpaths to the micromill.

Device designs were created using OpenSCAD (Wikibooks, 2017), a free and open source CAD tool that reads script files to generate solid models. The ability to describe a solid model using only text is an important distinction between OpenSCAD and typical 3D modeling software packages (e.g., AutoCAD, SolidWorks, etc.), which often require geometries to be manually drawn. Text-based modeling allows for automatic generation of device designs using a CAD workflow, such as that shown in Figure 3·1. MakerFluidics leverages OpenSCAD by creating device primitives that are stored in custom libraries. As shown in Listing 3.1, these primitives are parametric, allowing them to be "called" in a fashion similar to that of an object-based programming language.

Listing 3.1: This single line of OpenSCAD creates the 3D model shown in Figure 3·2(A). Once a primitive is defined in OpenSCAD, the parameters associated with each device geometry can be modified and the corresponding solid model will adjust to reflect the changes.

```
transposerNet(N=3, w_channel=1, r_valve=1, scale_x = 1);
```

Once a solid model is created using a CAD tool, the solid model must be imported by CAM software in order to generate the toolpaths that will be sent to the mill. MakerFluidics uses Autodesk Fusion 360 to create toolpaths for 3D models, such as mesh models (e.g., an STL file). All three CNC mills tested in the lab have machine-specific control software with the capability to directly process 2D models, such as a vector-graphics file (e.g., an SVG file). All of the CAM software listed above is free, but not necessarily open source. A standard operating procedure for interacting with Fusion 360 in order to generate toolpaths and operate the Othermill V2 or the Othermill Pro can be found in Appendix A.

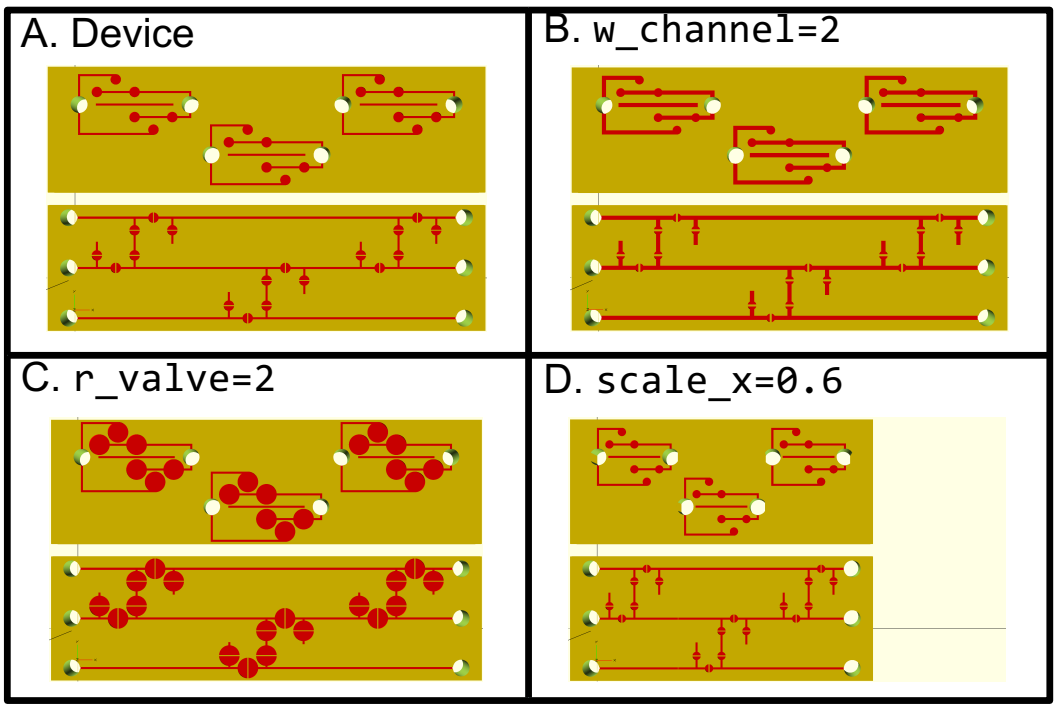


Figure 3·2: Example parameters associated with a network of transposer primitives. The primitive and routing topology are presented in detail in Chapter 4. (A) shows the baseline device. The control layer is shown above the flow layer. (B) illustrates modifying the channel parameter to make the channels wider. (C) shows how the valve dimensions can be modified using a parameter associated with valve radius. (D) shows how the entire device can scale in the x-dimension using yet another parameter.

3.5 Microfluidic Fabrication

The fabrication of a microfluidic device typically has two main steps: pattern geometries and seal layers (McDonald and Whitesides, 2002).

3.5.1 Pattern Geometries

Channel and valve geometries are etched in thermoplastic polymers using a desktop CNC mill. This stands in sharp contrast from conventional methods of microfluidic fabrication, namely photolithography, which requires the use of clean room facilities

and highly specialized equipment. The CNC approach is well-suited for integrating valving technologies such as monolithic membrane valves (Grover et al., 2003) (Figure 3·3) and centrifugal capillary valves (Madou and Kellogg, 1998). A significant trade-off for the relative ease of CNC milling thermoplastics using a desktop (i.e., not industrial-grade) CNC mill is that the maximum resolution is 25μm with an exponential increase in reliability seen at feature sizes greater than 250μm (Guckenberger et al., 2015), whereas microfluidic geometries using conventional methods such as photolithography can reliably achieve features smaller than 1μm (McDonald and Whitesides, 2002).

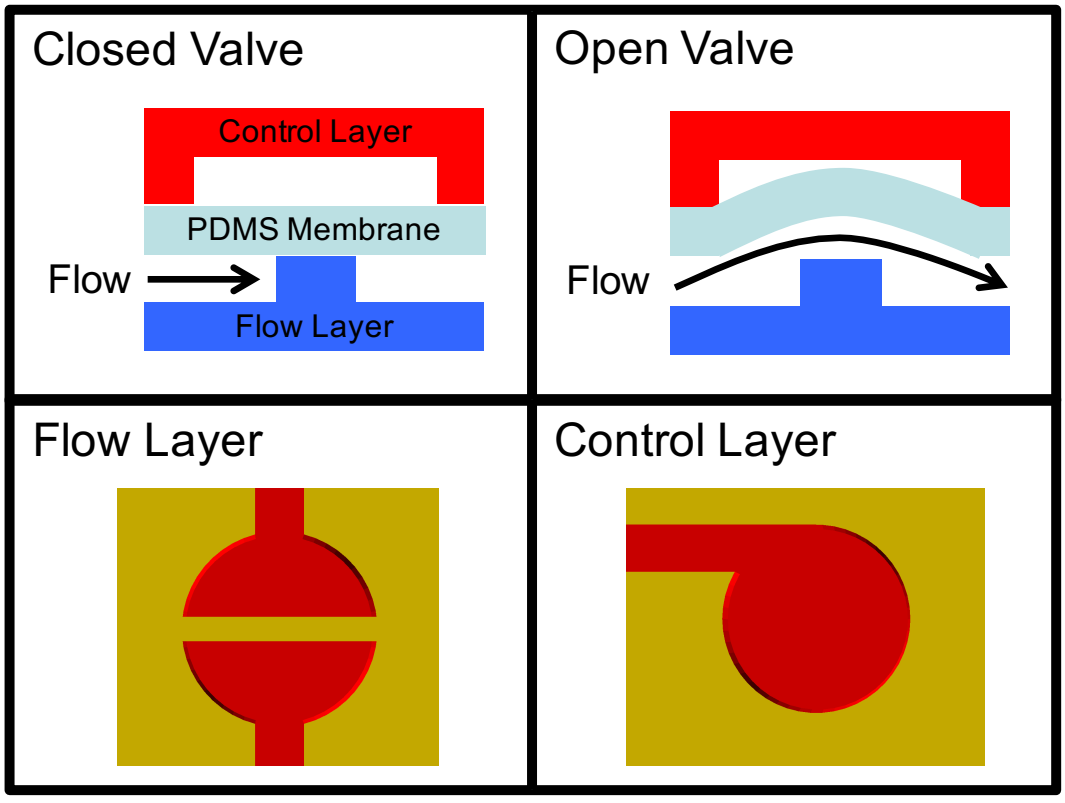


Figure 3·3: Normally-closed monolithic membrane valves (Grover et al., 2003) are realized by introducing discontinuities in the flow layer (blue) and a corresponding pneumatic cavity in the control layer (red). These two layers are separated by a PDMS membrane. To open the valve a vacuum is introduced into the cavity in the control layer.

Table 3.2: Published dimensional constraints regarding devices fabricated using the MakerFluidics protocol versus that of soft lithography. Aspect Ratio is defined as feature height to feature width.

Dimension	MakerFluidics	Soft Lithography	Reference
Aspect Ratio	Tool Dependant (often 3:1)	1:20	(Schaller et al., 1999) (Qin et al., 2010)
Minimum Feature Size	Tool Dependant (25μm Demonstrated) (5μm Theoretical)	<1μm	(Sweatt et al., 2008) (Qin et al., 2010)
Minimum Feature Spacing	25μm	<1μm	(Yen et al., 2016) (Qin et al., 2010)
Bonding Capacity	5 psi	50 psi	(McDonald and Whitesides, 2002)
Materials	Thermoplastics Soft Metals Cured Thermosets	PDMS Silicon	(Schaller et al., 1999) (Qin et al., 2010)

3.5.2 Seal Layers

Once geometries are etched into the desired substrate, sealing these channels becomes the next challenge. Polydimethylsiloxane (PDMS) is a common material for fabricating microfluidics (McDonald and Whitesides, 2002) and is also commonly used to encapsulate solar panels and outdoor lighting. It is because of the latter property that PDMS (Sylgard 184, Dow Corning) is available through retail outlets, such as Amazon, and, thus, falls within the constraints for adoption by the MakerFluidics fabrication paradigm. PDMS can be sealed irreversibly through modifications to its surface chemistry via plasma or corona exposure or sealed reversibly simply using the material's inherent van der Waals attraction to various materials including itself, glass, and thermoplastics (McDonald and Whitesides, 2002). Since irreversible sealing through surface treatments involves specialized machinery, MakerFluidics employs the latter method via van der Waals force. The trade-off being that the reversible seal cannot withstand pressures greater than 5 psi (McDonald and Whitesides, 2002).

The entire protocol is summarized in Figure 3·4.

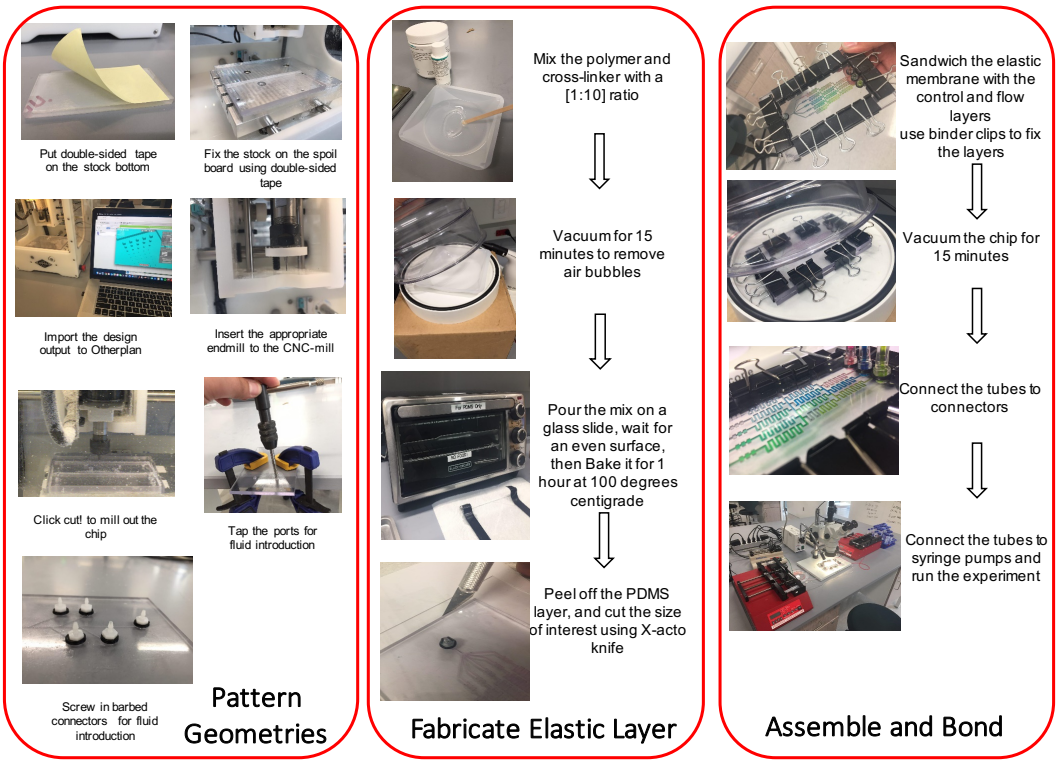


Figure 3·4: Workflow for fabricating microfluidics using the MakerFluidics framework.

3.6 Experimental Control

MakerFluidics views experimental conditions as a sequence of temporally-specified valve conditions. This necessitates a data structure consisting of an enumerated array of valve objects and a sequence of temporal specifications regarding their state. This data structure can be automatically generated by microfluidic CAD software, but it can also be created manually as a series of wait statements and valve conditions shown in Figure 3·5. These valve objects are linked to the microfluidic layout provided to the fabrication software through the use of a graphical user interface (GUI).

Pneumatics are provided through an array of 3D printed syringe pumps controlled

by custom firmware on an Arduino microcontroller. This microcontroller receives serial commands derived from the GUI running on a host computer. The MakerFluidics control GUI and firmware is extensible and interoperable with a conventional, solenoid-driven control infrastructure.

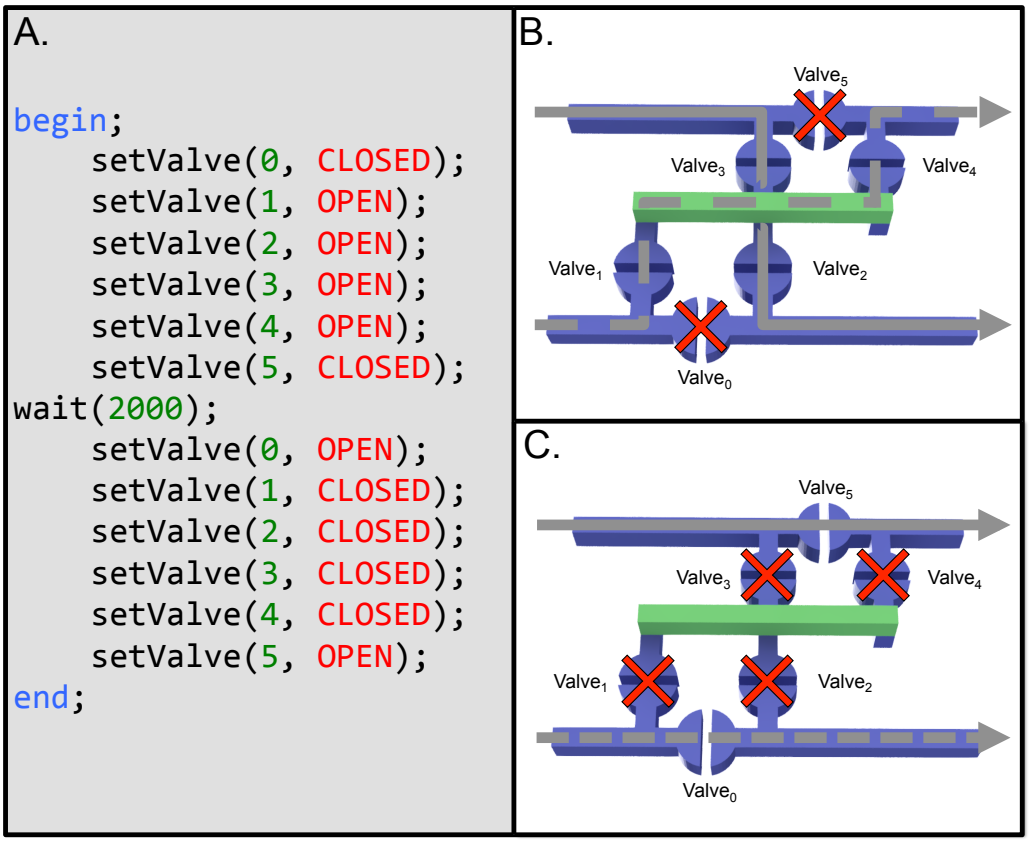


Figure 3·5: The temporal specification (A) (Thies et al., 2008) for a transposer device (presented in Chapter 4) containing six valves (B, C). The specification in (A) dictates the set of conditions in (B) to begin the assay. After 2000ms the valves change state to that shown in (C), thus affecting the movement of fluid through the device.

3.7 Applicability of MakerFluidics

While the collection of protocols and technologies that define MakerFluidics may seem simplistic, the relevance of the framework was demonstrated by its ability to

design, fabricate, and control a complex network of novel microfluidic primitives, as described in Chapter 4. Additionally the framework was used to fabricate experimentally relevant devices in industry; this work is presented in Chapter 5.

Chapter 4

A reconfigurable continuous-flow fluidic routing fabric using a modular, scalable primitive

4.1 Introduction

Microfluidic devices, by definition, are required to move liquids from one physical location to another. Given a finite and frequently fixed set of physical channels to route fluids, a primitive design element that allows reconfigurable routing of that fluid from any of n input ports to any n output ports will dramatically change the paradigms by which these chips are designed and applied. Furthermore, if these elements are "regular" regarding their design, the programming and fabrication of these elements becomes scalable. This paper presents such a design element called a *transposer*. We illustrate the design, fabrication and operation of a single transposer. We then scale this design to create a programmable fabric towards a general-purpose, reconfigurable microfluidic platform analogous to the Field Programmable Gate Array (FPGA) found in digital electronics.

4.2 Problem Statement

Engineering frequently involves the exploration of design tradeoffs. Such tradeoffs include time versus quality or performance versus cost (Otto and Antonsson, 1991). A tradeoff frequently encountered in the design of microfluidic systems is specificity

versus flexibility. Devices that perform specific tasks often can do these tasks with great precision and quality but users require new devices and/or device architectures for each subsequent task (Fidalgo and Maerkl, 2011). This lack of re-use increases the cost of experimentation over time, requires re-training for each device, and prevents common platforms for legitimate comparison of results across experiments. Flexible devices on the other hand may not meet stringent performance requirements or the cost of reconfiguring devices may be prohibitively high in terms of time or training (Thies et al., 2008). A microfludic device that offers both the performance of a specific solution with the programmability of a flexible solution coupled with low financial as well as time costs would enable entirely new classes of experimentation and design.

Continuous-flow microfluidic chip architectures have yet to fully benefit from the reconfigurable flexibility offered by digital microfluidics (Fair, 2007). Digital microfluidics benefit from extensive research on routing algorithms (Curtis and Brisk, 2015) that avoid cross-contamination stemming from undesired droplet collisions (Cho and Pan, 2008). These algorithms are predicated on the classification of fluids as discrete droplets, rather than as a continuous-flow. Additionally, digital microfluidic routing algorithms presume that droplets are able to bypass others using a grid of prefabricated paths, combined with the ability to stop the movement of some droplets while continuing to position others (Zhao and Chakrabarty, 2012). In contrast, continuous-flow microfluidics must maintain continuous-flow for correct operation of devices such as gradient generators (Hung et al., 2005), cell traps (El-Ali et al., 2006), and batch separators (Pamme, 2007), thus interrupting flow for the purposes of fluid routing is undesirable. This distinguishes the fluid routing problem from that of a digital microfluidic chip, but it also makes the problem analogous to signal routing in electronic digital design where wires cannot both carry a signal and intersect, which could result in a metastable condition (Kleeman and Cantoni, 1987).

One approach to the challenge of continuous-flow routing is to examine how reconfigurable computing hardware (Todman et al., 2005) provides both highly specific, yet programmable, computing elements linked together using a flexible routing fabric (Compton and Hauck, 2002). Such an approach retains the specificity of the dedicated resources while allowing those resource relationships (e.g. input and output) to be defined dynamically. Lessons gleaned from the development of modern reconfigurable platforms in the digital electronics domain, namely island-style FPGAs (Schmit, 2005), tell of the need for two major architectural components (Kuon et al., 2008):

1. Functional primitives that scale regularly, lending to automated design placement

2. A routing architecture that actuates regularly, lending to automated, dynamic signal routing

These architectural components must then be linked to a control environment using software that scales with the architecture. This requirement prevents chip designers from having to generate artisanal control software for every new chip architecture. As this work demonstrates, the regularity of the primitive and routing structure is what allows for algorithmic scaling in design and control.

Examples of functional, continuous-flow microfluidic primitives that scale regularly are multiplexors (Thorsen et al., 2002), storage elements (Thies et al., 2008), culture chambers (Fidalgo and Maerkl, 2011) and mixers (Jensen et al., 2013). Chip designs using scalable microfluidic primitives in the absence of a dynamic routing architecture carry similar tradeoffs to those found while designing electronic Application Specific Integrated Circuits (ASICs), whereby the major constraint is that, once implemented, these functional primitives are effectually "frozen in silicon," or in the

case of many microfluidic chips: “frozen in polydimethylsiloxane (PDMS).”

4.3 Related Work

Microfludics at chip densities belonging to the classes of large scale integration (LSI), i.e., chips with hundreds to thousands of components (Thorsen et al., 2002), and very large scale integration (vLSI), i.e., millions of components, have been demonstrated (Araci and Brisk, 2014). Electronic Design Automation (EDA) tools for automated design (McDaniel et al., 2013), layout (Huang and Densmore, 2014b), and control (Thies et al., 2008) of these types of chips have also been developed. However, each tool and design paradigm cited above views signal routing as a static problem akin to signal routing in electronic ASIC design, the primary constraint being that continuous-flow channels cannot both carry a signal and intersect on the same fluidic layer without being separated by a valve. EDA tools and chip architectures can overcome this constraint by either incorporating design rules that necessitate the avoidance of channel collisions (Huang and Densmore, 2014b) or dealing with channel collisions individually through the creation of new primitives such as underpasses and vias (Huft et al., 2013). Additionally, some attempts at channel routing incorporate an element of programmability that necessitates interruption in continuous flow to achieve arbitrary routing (Thorsen et al., 2002). These programmable architectures will be be described in further detail in Section 4.5.4. What these efforts lack is an ability, integrated into both the chip architecture and control environment, to dynamically route signals to any functional primitive post-fabrication while maintaining continuous-flow and, thus, create a general-purpose fluidic architecture. Only by unlocking this capability can the field of continuous-flow microfluidics expand its design spectrum to include general-purpose platforms among its vast repertoire of application-specific chips thus mirroring the evolution of digital electronics towards

experimental pervasiveness.

This work addresses that need, namely for a scalable, continuous-flow fluidic routing architecture using a network of pre-fabricated channels and integrated software for automated device actuation. A key contribution is the introduction of a "transposer" primitive. Transposers allow for a reconfigurable routing fabric. This architecture allows microfluidic chips incorporating high chip densities to better realize the benefits of scaling and come closer to achieving the functional ubiquity of digital electronics.

4.4 Primitive Architecture

The theory of operation for a single transposer primitive is shown in Figure 4·1. Valves were designed using a monolithic membrane technique previously described (Grover et al., 2003). Briefly, normally closed valving mechanisms are created by introducing discontinuities in a flow channel. These discontinuities are covered by the elastomeric membrane. A corresponding pneumatic cavity is patterned on the control layer (Grover et al., 2006). When a vacuum is introduced into the pneumatic cavity, the elastomer is distended into the cavity, thus allowing flow to proceed in the channel (Nguyen et al., 2012).

A transposer primitive has the ability to selectively swap the contents of two channels, allowing the user to route fluids through the chip dynamically while maintaining continuous-flow. This design requires that one channel "jump" over another by traversing between the flow and control layers. One transposer primitive will have two input ports and two output ports. As shown in Figure 4·2, the primitive can take on one of two modes: crossed or straight. In the straight mode, fluids will pass through the primitive unbroken. When the primitive is crossed the contents of both inputs are swapped when viewed from the output ports.

Our primitive contains six valves, each of which are not individually addressable.

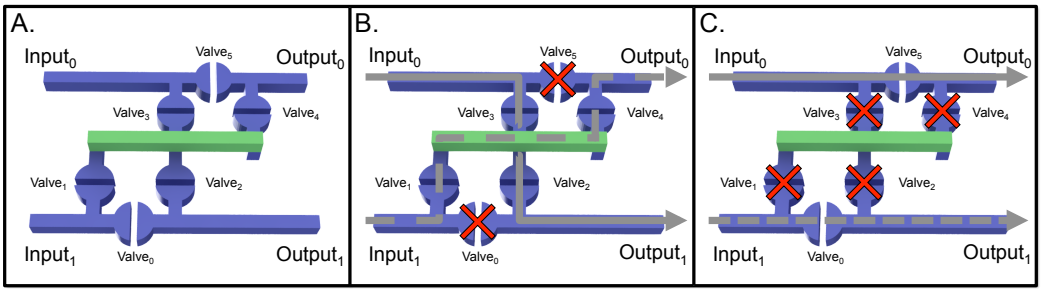


Figure 4·1: Theory of operation for a single transposer primitive. A. Normally closed elastomeric membrane valves are realized through discontinuities in flow channels and a corresponding pneumatic cavity on the control layer. The via (green channel) is located on the control layer and is accessed through holes punched in the PDMS membrane. Valves 0 and 5 and valves 1, 2, 3, and 4 are each driven by a single control line, thus one transposer primitive only requires two control lines. B. When valves 0 and 5 are closed and valves 1, 2, 3, and 4 are open the primitive will be in crossed mode in which the fluids will swap channels when viewed from the outputs. C. When valves 1, 2, 3, and 4 are closed and valves 0 and 5 are open the primitive will be in straight mode and fluids will flow straight through the primitive.

This is to reduce the number of pneumatic inputs required to operate the primitive allowing for greater scalability. Valves 0 and 5 in Figure 4·1 are controlled by a single control line as are valves 1, 2, 3, and 4; thus the six valves in each primitive only require two control lines for operation.

4.4.1 Microfluidic Materials and Assembly

The microfluidic assembly stack for a single transposer primitive is shown in Figure 4·3. Channel and valve features were ablated from polycarbonate (PC) (McMaster-Carr, Robbinsville, NJ, USA) using a desktop CNC-mill (Othermill, Othermachine Co., San Francisco, CA, USA). The PDMS membrane was fabricated by mixing a prepolyer (Sylgard 184 Silicone Elastomer Kit, Dow Corning, Midland, MI, USA) with the curing agent at a ratio of 10:1. This mixture was then degassed in a vacuum desiccator. A 250μm thick PDMS membrane was produced using a method previously

described (Martinez-Duarte, 2012). The via on the control layer, shown in Figure 4·3, is accessed through holes punched in the PDMS membrane. Holes were punched by hand using a 1mm biopsy punch (Miltex, York, PA, USA). Flow and control layers were bonded to the PDMS membrane using van der Waals force as described previously (McDonald and Whitesides, 2002)(Duncan et al., 2013). Additional bonding strength was achieved through the use of office binder clips (Duncan et al., 2015). The assembled PC-PDMS-PC device stack was then placed in a vacuum desiccator to remove air bubbles created at the material interfaces during assembly.

4.4.2 Routing Fabric Definitions

In order to generate an algorithm that automatically routes fluids through the routing fabric to their desired destinations we begin by representing the transposer-based routing fabric as a **directed acyclic graph** (DAG). A **graph** $G = (V, E)$ is a data structure consisting of a set of **vertices** V and a set of **edges** E connecting these vertices (Vaidyanathan et al., 2015). A **directed graph** is a subset of graphs in which all edges are ordered pairs of elements $(v_i, v_t) \in V$. An edge e_{it} begins with an **initial vertex** v_i and ends in a **terminal vertex** v_t. A **path** from v_0 to v_n is an ordered set of edges $(v_0, v_1), (v_1, v_2), (v_2, v_3), \cdots, (v_{n-1}, v_n)$, such that the terminal vertex of each edge is the same as the initial vertex of the next edge in the path. In an **acyclic graph**, there exists no path that starts with a vertex v_i and ends with the same vertex v_i.

An **unrouted** graph will contain all possible edges in the routing fabric, however since each individual primitive has only two modes (crossed and straight) not all combinations of edges are possible. Figure 4·5C shows an unrouted graph for a three-input fabric of transposer primitives. A **routed** graph contains only the edges used to route source nodes to the terminal nodes set by a user. Figure 4·2 shows examples

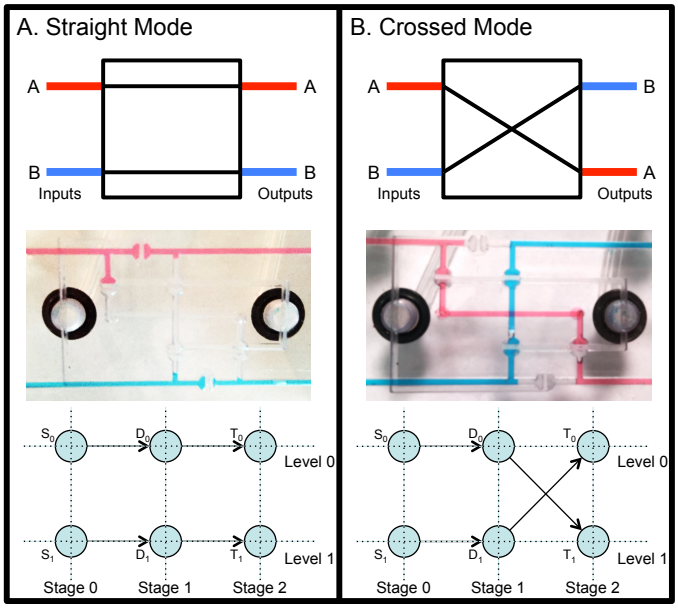


Figure 4·2: The functional, physical and graphical representations above demonstrate how the transposer can route fluids straight through the primitive (A.) or swap the inputs when viewed from the output ports (B.). Vertices labeled v_i where $v \in \{S, D, T\}$ correspond to source, decision and terminal nodes, respectively, the formal definitions for which are found in Section 4.4.2. The i value for source and terminal nodes represent the vertex's level, as there will only be one of each per level, whereas the i value for decision nodes are numbered incrementally by stage, left to right, starting at level 0, stage 1 as described by Algorithm 1. The addressing node v_o for the single transposer primitive represented above is D_0 since it is a decision node that exhibits heteroparity of coordinates (i.e., $p_l + p_s$ is an odd number).

of routed graphs for a single transposer in each primitive mode, straight and crossed. A vertex is a **decision node** when it is a terminal vertex for at least one edge and an initial vertex for exactly two other edges in the unrouted graph and exactly one other edge in a routed graph. A vertex is a **source node** when it is not a terminal vertex for any edge in the graph. A vertex is a **terminal node** when it is not an initial vertex for any edge in the graph.

Each source node occupies a horizontal **level** in the graph. At a decision node,

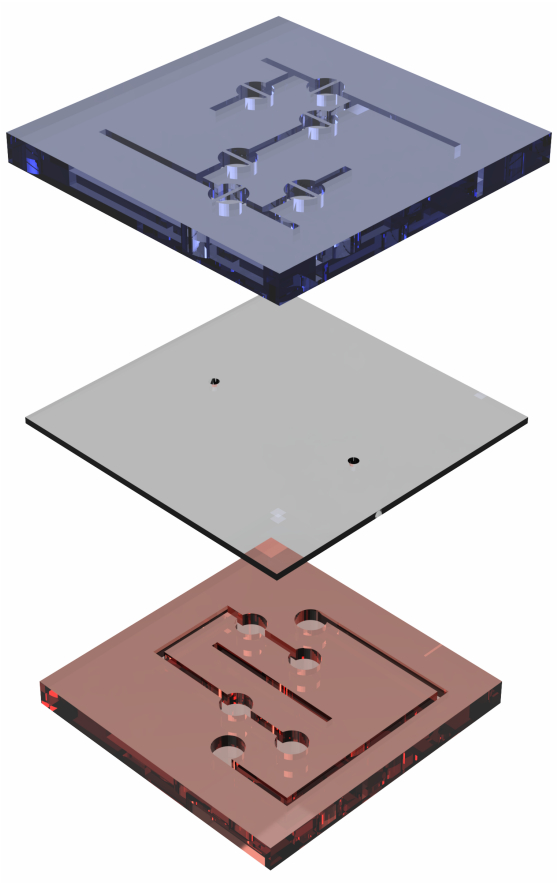


Figure 4·3: Microfluidic assembly stack. Channel and valve geometries in flow (blue) and control (red) layers were ablated from polycarbonate. The via on the control layer is accessed from the flow layer by through-holes punched in the PDMS membrane that separates the two polycarbonate layers.

the next edge in the graph may either stay on the same level or traverse a level. The direction of the traversal (either increment a level or decrement a level) depends on the position of the vertex, as described in Section 4.6.1. For example, the decision node D_0 in Figure 4·2 may either traverse from level 0 to level 1, thus incrementing a level, or stay on level 0, whereas the decision node D_1 may only traverse from level 1 to level 0, thus decrementing a level, or stay on level 1. Each vertex occupies a vertical **stage** in the graph. All edges in the graph traverse stages. For each vertex

$v \in V$, $\exists$ a set of points $p = p_l, p_s$ that represent the vertex's coordinates expressed in terms of level, l, and stage, s.

Microfluidic elements are mapped onto a graph framework in order to translate the results of a graph traversal into proper fluid flow on a physical device. **Flow ports** are placed at all source (input) and terminal (output) nodes. One **control port** must be placed on each channel of equipotential in the control layer shown in Figure 4·3, with the exception of the via. Based on this requirement, each primitive requires two control ports. A **transposer primitive** x_o is addressed by a single decision node $v_o \in V$. v_o is said to have heteroparity of coordinates (i.e., $p_l + p_s$ is an odd number). A list of transposer primitives X maintains the locations of each individual primitive x_o addressed by p_{v_o}.

4.5 Theory of operation

4.5.1 Scalable Routing Fabric

A single transposer primitive can route fluids from two different input channels to two different output channels. A routing fabric, by definition, is composed of multiple transposer primitives and can be built to allow for fluid routing of n-input channels. The architecture on which this fabric is built is a mason layout, which can be defined as an array of primitives with either homoparity or heteroparity of coordinates; our fabric arranges primitives based on heteroparity of coordinates to account for source nodes at stage 0. The scaling of such an architecture is demonstrated in Figure 4·4. The number of primitives scales according to the number of inputs, n, as defined by the recursively defined function in Equation 4.1. Since a single transposer primitive can route two inputs, the number of primitives required to route an n less than two would be zero.

$$f(n) = \begin{cases} 0, & \text{if } n \text{ is } < 2 \\ f(n-1) + n - 1, & \text{if } n \text{ is } \geq 2 \end{cases} \tag{4.1}$$

Thus, for $n \geq 2, f(n) = \sum_{i=1}^{n-1} i = n \cdot (n-1)/2 \in \Theta(n^2)$.

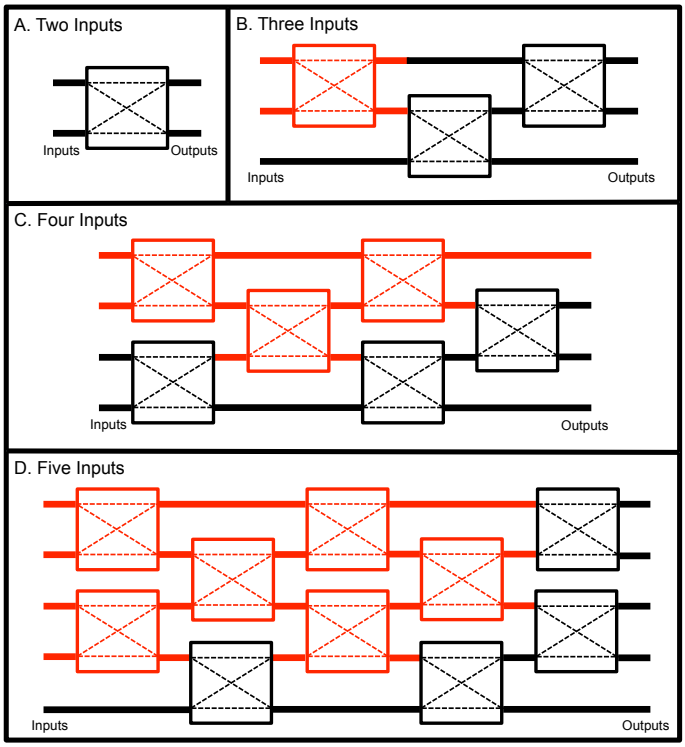


Figure 4·4: Examples of a populated routing fabric. A. shows a single primitive, represented as a box, which is able to route two fluidic inputs to two fluidic outputs. This primitive (highlighted in red in B.) can be tiled with two other primitives, thus enabling the ability to route three fluidic inputs. The addition of three primitives (C.) to the three-input architecture (red) results in a four-input fabric. D. shows the natural continuation of the tiling process by adding four additional primitives to the four-input fabric (red) in order to create a five-input fabric. The number of transposer primitives required $f(n)$ for A–D is 1, 3, 6 and 10, respectively, as dictated by Equation 4.1.

4.5.2 Optimality of the number of transposers

Below, we show that a quadratic number of transposers is necessary and sufficient for routing any permutation of n inputs to n outputs.

Necessity

Routing an arbitrary permutation of n inputs requires reordering the input fluids using transposers, in such a way that they arrive at their intended output destinations. We observe that each transposer is capable of reordering exactly two channel positions. Consider the scenario where inputs i must be routed to output $n-i+1$. Here, input i must require $|n-i+1-i| \in \Theta(n)$ transpositions, and therefore $\Theta(n)$ transposers. Thus, to route all n input channels in this case, the problem scales on the order of $\Theta(n^2)$ transposers. It is important to note that this is a claim regarding the routing problem's complexity; the actual number of transposers required is described in Equation 4.1.

Sufficiency

Routing is accomplished by correctly routing an initial input to its destination, and then recursing. In this fashion, all inputs can be routed based on the necessary condition outlined above. The definition of correct routing is described in Section 4.6.2.

4.5.3 Planar and non planar fabrics

Consider a graph with a vertex for each transposer and an arc from vertex u to v if fluid can flow directly out of the transposer corresponding to vertex u into that corresponding to vertex v, without flowing through any transposers along the way. It is easy to observe that this graph is planar for any n-input fabric. The "intersection"

of flows occurs within a transposer. For planar fabrics, the above argument shows that the fabric design described is optimal in the number of transposers.

4.5.4 Comparison with alternative primitives

Similar programmable routing could be achieved using existing primitives. For example, a one-of-n input could be routed to any of n outputs using a multiplexer-demultiplexer pair with $\Theta(\lg n)$ control lines (Thorsen et al., 2002). Additionally, this design would not allow for continuous-flow across all n input channels, as multiplexors and demultiplexers are one-of-n and n-of-one devices, respectively. Flow in channels not selected would, therefore, be stopped. A naive way to obtain an n-input n-output routing fabric from this would be to use n mux-demux pairs resulting in $\Theta(n \lg n)$ control lines. Such a design, however, would be complex and non planar. Conversely, sub quadratic non planar fabric designs may exist, but we do not explore them here.

4.5.5 Analysis of control requirements

Since each transposer primitive requires two control lines, the number of control lines also scales on the order of $\Theta(n^2)$ with the actual number of control lines required equaling $n(n-1)$. Ballooning control requirements is a fundamental weakness of microfluidic LSI. This weakness has been largely mitigated through the use of microfluidic multiplexors (Thorsen et al., 2002). The microfluidic multiplexor creates an infrastructure in which m valves are controlled using $2log_2(m)$ control lines. Thus, if a microfluidic multiplexor is integrated into to the routing fabric's control framework using latched or unlatched multiplexed addressing, as previously described (Melin and Quake, 2007), the number of control channels would scale at $2log_2[n(n-1)]$ and as such 32 fluids could be routed with 20 control lines and the number of fluids able to be routed would increase exponentially with each additional control line (ex. 50

control lines will arbitrarily route 5,798 fluids).

4.6 Routing Algorithm

There are two main steps in our algorithm for routing within a transposer fabric:

1. Generate an unrouted graph based on the number of inputs to be routed n
2. Delete unnecessary edges based on desired fluid destinations to form a routed graph

Locations on a graph are described in terms of stages and levels. A graph for $n > 2$ will have n levels and $n + 2$ stages, the additional two stages are a result of source and terminal nodes. Coordinates are zero-indexed.

4.6.1 Generate Unrouted Graph

Generating an unrouted graph for $n > 2$ requires three primary steps:

1. Populate vertices according to rules set forth in Algorithm 1.
2. Assign vertices to individual transposers according to Algorithm 2.
3. Create edges to form an unrouted graph.

Populate Vertices

The first step in generating an unrouted graph for $n > 2$ is to populate the vertices based on the number of inputs, n. Vertices are placed based on a set of rules. Source nodes are placed on stage 0 of each level, while terminal nodes are placed on stage $n+1$, remembering that coordinates are zero-indexed. There are two main edge cases when placing decision nodes, Level 0 and Level $(n - 1)$. Placing nodes on all other levels is regular as demonstrated by Algorithm 1.

Algorithm 1: Populate Vertices. Creates vertices and assigns coordinates p and type characteristics $v \in \{S, D, T\}$ to each created vertex.

```
Data: Number of inputs to be routed n
Result: A list of vertices V
begin
    levels ← n − 1
    stages ← n + 1
    count ← 0
    V ⟵ ∅
    for Each level do
        S_level ⟵ p = level, 0
        T_level ⟵ p = level, n
        AppendToV(S_level)
        AppendToV(T_level)
    end
    /* Populate decision nodes                                    */
    for Each level do
        for stages 1 through n do
            if level = 0 and stage is odd then
                D_count ⟵ p = level, stage
                AppendToV(D_count)
            end
            else if level = n − 1 then
                if n is odd and stage is even then
                    D_count ⟵ p = level, stage
                    AppendToV(D_count)
                end
                else if n is even and stage is odd then
                    D_count ⟵ p = level, stage
                    AppendToV(D_count)
                end
            end
            else if level ≠ 0 or level ≠ n − 1 then
                D_count ⟵ p = level, stage
                AppendToV(D_count)
            end
            Increment count
        end
    end
end
```

Algorithm 2: Place Transposers. Finds all decision nodes with heteroparity of coordinates, creates a new transposer object x_o in that location and adds the new transposer to the set of transposers X

```
Data: A list of vertices V
Result: A list of transposer objects X
begin
    X ⟵ ∅
    for v ∈ V where p_{s_v} ≠ 0 or p_{s_v} ≠ n + 1 do
        if p_{l_v} + p_{s_v} is odd then
            x_o ⟵ v
            AppendToX(x_o)
        end
    end
end
```

Place Transposers

Individual transposers x_o are addressed by a single decision node v_o that exhibits heteroparity of coordinates (i.e., $p_{l_{v_o}} + p_{s_{v_o}}$ is odd). Since v_o can only be of type D, for decision node, this excludes all vertices in stage 0 (source nodes) and stage $n + 1$ (terminal nodes). Once v_o is identified for all transposers in a graph, it is then possible to correctly create edges to form an unrouted graph. Algorithm 2 takes the list of vertices V generated by Algorithm 1 and searches for decision nodes with heteroparity. The algorithm then creates a new transposer primitive x_o, assigns it an address p_{v_o} and adds the individual transposer to the set of transposers X.

Generate Unrouted Graph

This step links the vertices in list V with edges and, upon completion, results in an unrouted graph that represents a fabric of transposers for n inputs. Algorithm 2 ensures that for each transposer $x \in X$, $\exists$ a decision node $v_o \in V$, and v_o has the characteristic of heteroparity of coordinates; these characteristics make it possible to correctly link all vertices with edges that will form an unrouted graph representing a

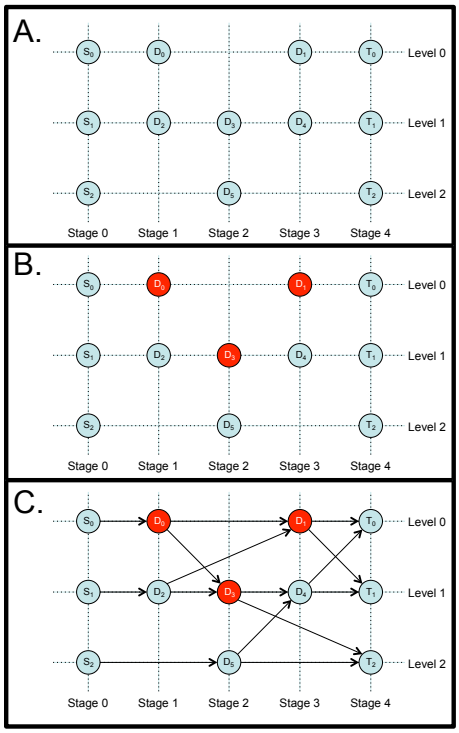


Figure 4·5: Algorithmic progression for generating an unrouted graph for a three-input ($n = 3$) routing fabric. A. shows the outcome of Algorithm 1, where vertices v_i have coordinates p and types $v \in \{S, D, T\}$ corresponding to source, decision and terminal nodes, respectively. B. shows in red the locations of the three transposer primitives as dictated by Algorithm 2. The number of primitives is governed by Equation 4.1 and the primitives are defined by their addressable node v_o. The graph in C. is the complete unrouted graph as dictated by Algorithm 3.

fabric of transposer primitives.

The first edge will connect v_o to the next vertex on the same level as v_o, i.e., $p_{l_{v_o}}$; therefore, the initial vertex for this edge will be v_o and the terminal vertex will be the infimum of the set of vertices on $p_{l_{v_o}}$ with a stage number greater than $p_{s_{v_o}}$. The second edge will be the same as the first except that the terminal vertex will be the infimum of the set of vertices on the level incremented from that of v_o, i.e., $p_{l_{v_o}} + 1$, that has a stage number greater than $p_{s_{v_o}}$. For each transposer primitive, two edges will also be added to each vertex that exists on the same stage as v_o, but incremented

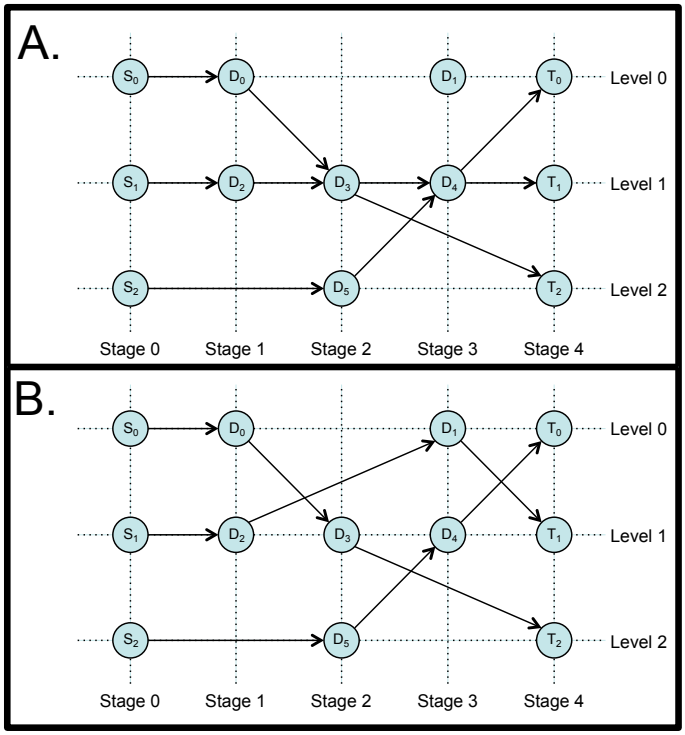


Figure 4·6: Graphical representations of illegally routed (A.) and correctly routed (B.) graphs. Each graph routes inputs from S_0 to T_2, S_1 to T_1 and S_2 to T_0. The graph in A. is illegal because vertices D_3 and D_4 appear in more than one path.

one level, i.e., $p_l + 1$; we will refer to this vertex as v_e. The first edge will connect v_e to the next vertex on the same level as v_e, while having a stage number greater than $p_{s_{v_e}}$. The second edge will terminate at the next vertex on the level decremented from v_e, which is also the same level as v_o, having a stage number greater than $p_{s_{v_e}}$. These steps are illustrated in Figure 4·5C and formalized in Algorithm 3.

4.6.2 Correctly Traverse Unrouted Graph

A routing fabric for continuous-flow microfluidics must avoid cross-contamination of fluids by ensuring that two channels do not intersect and carry a signal unless they are separated by a valve. The transposer-based routing fabric accomplishes this based on the bimodal nature of the primitive. In other words, since the primitive only has two

Algorithm 3: Generate Unrouted Graph. Creates edges between all vertices in V based on their locations on the graph and their association with transposer elements

Data: A list of vertices V and transposers X
Result: $G = (V, E)$
begin
  $E \longleftarrow \emptyset$
  `/* Route all source nodes to first decision node in level */`
  for $v \in V \mid p_{s_v} = 0$ **do**
    $v_i = v$
    $v_t = inf\{V \mid p_l = p_{l_v}, p_s > p_{s_v}\}$
    $e_{it} \leftarrow (v_i, v_t)$
    `AppendToE`(e_{it})
  end
  for $x \in X$ **do**
    $v_{t_0} = inf\{V \mid p_l = p_{l_{v_o}}, p_s > p_{s_{v_o}}\}$
    $e_{it_0} \leftarrow (v_o, v_{t_0})$
    `AppendToE`(e_{it_0})
    $v_{t_1} = inf\{V \mid p_l = p_{l_{v_o}} + 1, p_s > p_{s_{v_o}}\}$
    $e_{it_1} \leftarrow (v_o, v_{t_0})$
    `AppendToE`(e_{it_1})
    $v_e = v \in \{V \mid p_l = p_{l_{v_o}} + 1, p_s = p_{s_{v_o}}\}$
    $v_{t_2} = inf\{V \mid p_l = p_{l_{v_e}}, p_s > p_{s_{v_e}}\}$
    $e_{it_2} \leftarrow (v_e, v_{t_2})$
    `AppendToE`(e_{it_2})
    $v_{t_3} = inf\{V \mid p_l = p_{l_{v_e}} - 1, p_s > p_{s_{v_o}}\}$
    $e_{it_3} \leftarrow (v_e, v_{t_3})$
    `AppendToE`(e_{it_3})
  end
end

Algorithm 4: Traverse Graph. Discards unnecessary edges by setting the mode (straight or crossed) for each transposer primitive in the unrouted graph generated by Algorithm 3

Data: $G = (V, E)$, A set D containing map elements d_i in the from of an ordered pair of source nodes and their desired terminal node (S_i, T_i)
`/* Ex: In Figure 4·6` $D = \{(S_0, T_2), (S_1, T_1), (S_2, T_0)\}$ `*/`
Result: $G' = (V, E')$, where $E' \subset E$ containing correctly routed paths $\forall d \in D$
begin
  for $d_i \in D$ **do**
    Find all paths from S_i to T_i
  end
  for *all paths* **do**
    Find paths $\forall d \in D$ containing no duplicate nodes
    Discard unused edges
  end
end

modes, crossed and straight, and since the architecture is based on an alternating, mason grid layout, this ensures that cross contamination will never occur.

Traversing the unrouted graph within the constraints of the primitive and architecture can be accomplished by creating paths that do not share vertices. Therefore, an **illegally routed graph** is one in which a vertex of any type appears in more than one path. An example of an illegally routed graph is shown in Figure 4·6. The graph is illegal because vertices D_3 and D_4 appear in more than one path. For example, vertex D_3 appears in two paths: $(S_0, D_0), (D_0, D_3), (D_3, T_2)$ and $(S_1, D_2), (D_2, D_3), (D_3, D_4), (D_4, T_1)$. Therefore, traversing an unrouted graph given an ordered set of non-repeating inputs, representing the desired fluid destinations, is a matter of generating paths for each individual inputs to their desired outputs while ensuring that no paths share vertices. This process is outlined in Algorithm 4.

4.7 Discussion

4.7.1 Primitive and Fabric

This work focuses primarily on the physical and algorithmic construction of a programmable fabric using multiple primitives towards flexible continuous-flow microfluidic routing. In order to demonstrate the sufficiency of our routing framework, we constructed a three-input fabric using the fabrication techniques described in Section 4.4.1. We then implemented our algorithms in software and used it to control the device. All possible permutations of three-input routes are demonstrated in Figure 4·7.

The primitive used in this work is provided simply as a means to demonstrate the functionality of the fabric. The actual architecture of the primitive used to construct the fabric is irrelevant, provided that it accomplishes the functional specification illustrated in Figure 4·2; thus, the fabric can be integrated with any continuous-flow microfluidic technology independent of fabrication technique and control environment. This effectively removes fluid routing considerations from chip design, thereby raising the level of abstraction, allowing experimentalists and chip designers to focus on developing functional continuous-flow microfluidic devices within the context of their particular application area. Furthermore, this work is applicable regardless of an experimentalist's existing microfluidic fabrication or control infrastructure, therefore what we present is immediately useful at no additional cost.

We acknowledge that our transposer primitive design can be optimized in many ways; for example, the primitive used in this work incorporates Manhattan geometries, which may increase transport waste and create turbulent flow patterns at right angle junctions. Manhattan geometries were chosen for clarity in demonstrating the functionality of the fabric as well as to utilize a microfluidic place-and-route tool developed in our lab (Huang and Densmore, 2014b). Our fabrication methods could

also be optimized with regards to materials, miniaturization, irreversible bonding, etc. Irrespective of optimizations to the primitive, the fabric in which the primitive integrates remains unchanged. In this regard, the routing fabric presented above can be integrated in any continuous-flow microfluidic technology, fabrication method or control environment regardless of the primitive's architecture or performance.

4.7.2 Comparison with Other Programmable Architectures

This work differs from other programmable architectures, such as the recently reviewed (Kim et al., 2016) class of programmable devices that consists of planar, two-dimensional valve arrays (Fidalgo and Maerkl, 2011)(Thorsen et al., 2002)(Jensen et al., 2013)(Linshiz et al., 2016), in its ability to achieve arbitrary continuous-flow routing while coupled with application-specific microfluidic primitives. While these two-dimensional valve array architectures present compelling cases for their ability to route, mix and store fluids, one significant shortcoming in this regard is their inability to arbitrarily route fluids while maintaining continuous flow through the device. This limitation precludes an experimentalist from using the architecture to serialize a computational work flow within a single chip architecture as described in Section 4.7.4.

The valve array architecture is proven to be an excellent platform for flexible, general-purpose microfluidic operations, to include operations other than fluid routing. The transposer-based routing architecture makes no claims as to its ability to perform any function other than algorithmically scalable, arbitrary fluid routing while maintaining continuous flow; as such, we believe our architecture consisting of a programmable routing fabric combined with application-specific (or programmable) functional blocks allows the experimentalist a maximum degree of flexibility to integrate their most-trusted microfluidic constructs into a programmable device.

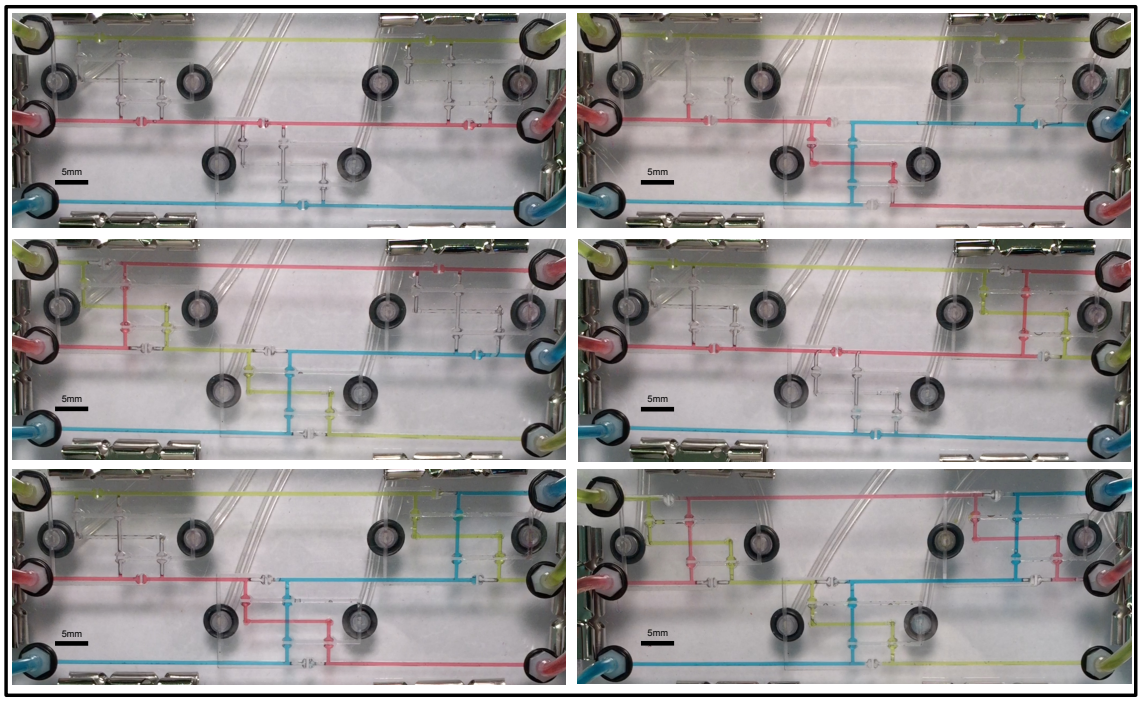


Figure 4·7: All possible permutations for a three-input routing fabric. Fluid moves from left to right in each photo. The inputs from top to bottom are always in the order: green, red, blue. The outputs of the photos correspond to all possible routing permutations for the three input fluids to the three output ports.

4.7.3 Integration into a larger functional architecture

Programmable routing is a key element in reconfigurable electronic computing architectures such as FPGAs (Compton and Hauck, 2002). An FPGA derives its basic computational abilities from configurable logic blocks (CLB), which are responsible for performing elementary logic and storage functions (Farooq et al., 2012). More complex computations are achieved by chaining many CLBs together. An FPGA infers flexibility from the programmable nature of both the CLB and the routing fabric. Island-style FPGAs (Schmit, 2005) are a particular FPGA architecture that organize CLBs in a two (Chang et al., 1996) or three dimensional (Alexander et al., 1995)

array connected by a grid of prefabricated wire segments and programmable routing structures (Chang et al., 1996). The CLBs can be viewed as computational "islands" among an ocean of routing elements (Farooq et al., 2012), as shown in Figure 4·9A.

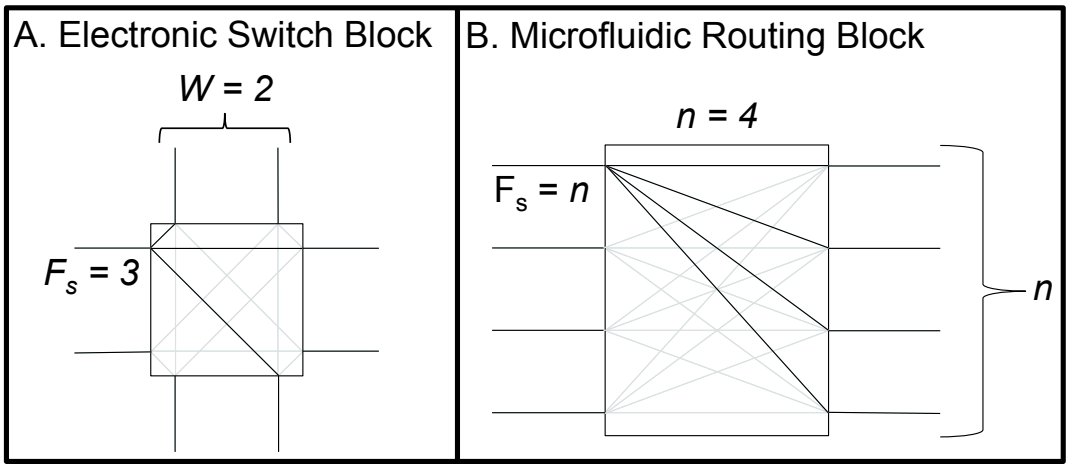


Figure 4·8: Electronic (A.) and Microfluidic (B.) programmable routing blocks. The channel width (W) and flexibility (F_s) of the electronic switching block (A.) is two and three, respectively. These switching block attributes are the same as the ones used in Figure 4·9A and 4·9B.

Programmable routing structures within an island-style FPGA can take on one of two general forms: **switching blocks** or **connection blocks**. Connection blocks are used to connect the prefabricated wire segments directly to CLBs. Switching blocks are used to connect prefabricated wire segments at dimensional intersections within the 2D or 3D array; for example, a 2D island-style FPGA architecture places a switching block at all intersections of vertical and horizontal prefabricated wire segments (Schmit and Chandra, 2002), as Figure 4·9A demonstrates. As shown in Figure 4·8, W denotes the number of prefabricated wire segments on each side of the switching block, while F_s denotes the switching block's flexibility, i.e., the number of possible routes for each prefabricated wire segment entering the switching block.

Figure 4·9B is an example of signal routing using a switching block with $W = 2$ and $F_s = 3$.

A transposer-based routing fabric can be viewed as having the functionality of a combination switching block and connection block, which we will call a **routing block**, in a 1D FPGA. A microfludic routing block can be described in the same terms as an FPGA switching block. As shown in Figure 4·9C, the routing block developed in this work has a flexibility $F_s = n$ and width $W = n$. Microfluidic **functional blocks** can take the form of any microfluidic primitive such as cell traps, gradient generators, or optical reporting areas. These functional blocks need not be uniform with regard to the number of input or output ports between blocks, nor must they be symmetric in the number of input and output ports in the same block, as Figure 4·9C shows. Functional blocks may take on any class of microfluidic function including, but not limited to, combining operations (mixing, droplet generation, etc.), biological operations (cell culture, polymerase chain reaction (PCR), etc.), or measurement/sensing operations (microscopy, pH, etc.).

In microfluidic and electronic FPGA architectures, elementary functions are performed in functional blocks and CLBs, respectively. More complex operations are accomplished by chaining multiple blocks together using our routing fabric. Since all elements are modular and programmable, both architectures provide flexibility that is impossible to achieve in traditional, application-specific chips. Chaining multiple primitives through microfluidic routing blocks has compounding effects on a chip's flexibility and functional capability in a manner similar to that of an FPGA. This concept is described in further detail in Section 4.7.4. These novel capabilities are impossible in a continuous-flow microfluidic device without the capability of dynamic fluid routing provided by the transposer routing fabric.

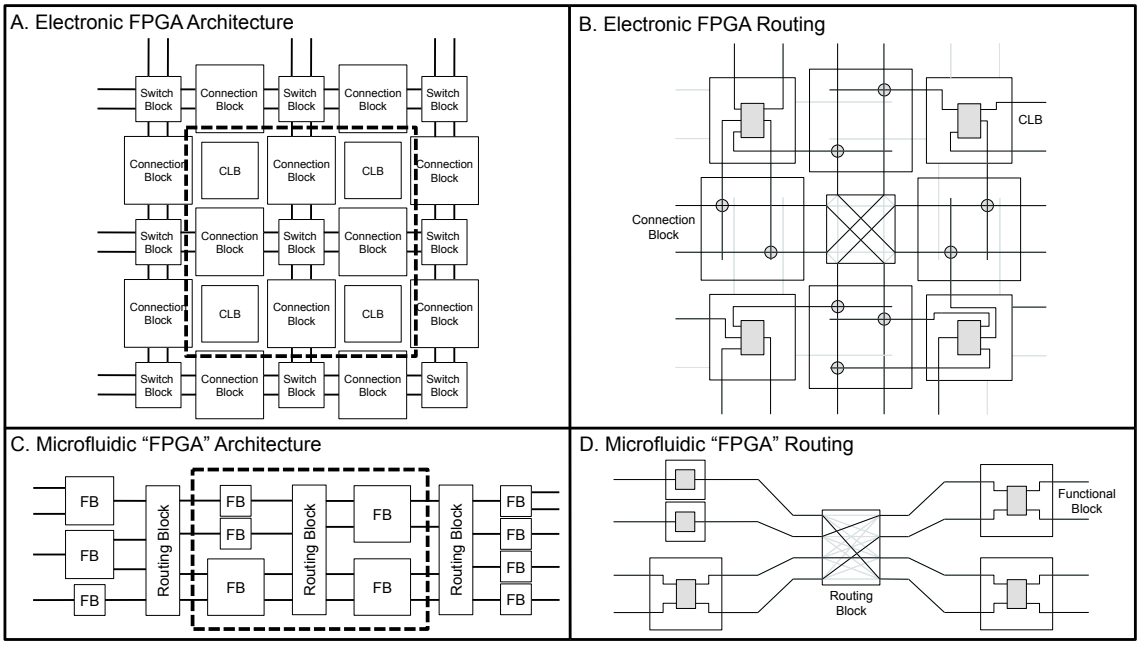


Figure 4·9: Electronic and Microfluidic programmable architectures. A 2D, island-style Electronic FPGA architecture (A.) with example routing (B.) of the area of interest indicated by a dashed line in A. A transposer-based microfluidic chip architecture (C.) with example routing of area of interest (D.) indicated by a dashed line in C. Microfluidic functional blocks (FB) can be any continuous-flow microfluidic function such as mix, measure, etc. The microfluidic architecture in C. demonstrates that functional blocks need not be uniform (i.e., all have the same number of input/output channels). Grey lines in B. and D. indicate unconnected paths.

4.7.4 Example Architecture

Consider a lab that utilizes two distinct microfluidic devices. The first device is a two-stage microfluidic designed to mix a particle stream with a buffer stream and then perform separation as part of a protocol for rare cell isolation (Pamme, 2007). The second device is also a two-stage microfluidic but instead of performing mixing and separation, this device utilizes cell trapping followed by a thermal treatment within a temperature-controlled reaction chamber as part of a polymerase chain re-

action (PCR) protocol (Zhang et al., 2006). If these two distinct chips are brought together, with a single transposer separating the two stages, a flexible architecture is created with novel functionality. For example, the two independent chips were originally only capable of mixing followed by separation and trapping followed by temperature cycling, respectively; the new chip, incorporating a single transposer, is now capable of mixing followed by temperature cycling, which could be useful in chemical synthesis (Jensen et al., 2014), and at the same time is capable of trapping followed by separation, which is part of a protocol for debulking platelets from blood samples (Shields IV et al., 2015).

Figure 4·10 shows a schematic representation of all possible functionalities derived from such an architecture. The addition of more functional blocks vertically (i.e., increasing n on a routing block) increases the number of possible simultaneous operations the chip can perform, while adding more functional blocks horizontally, with each stage separated by a routing block, increases the number of possible sequential operations. This specificity achieved through well-tested and characterized functional blocks (ex., mixers, separators, cell traps and reaction chambers) coupled with the flexibility of a programmable routing architecture unlocks a new category of programmable devices made possible only through arbitrary, continuous-flow microfluidic routing.

Functional blocks may have incompatible channel geometries relative to other functional blocks in a larger architecture. This issue can be mitigated by modifying the primitive's channel geometries for each differing channel of interest. This would only be necessary should a channel reduction or expansion impede function and if the differing channels require rerouting; otherwise, these geometries would be considered internal to a functional block and not part of the routing fabric. Should a chip architecture require rerouting of multiple channels of differing dimensions, this would

result in separate routing fabrics for each channel geometry that requires rerouting. The routing fabric is flexible such that a device containing separate routing fabrics, based on channel dimensions, could be integrated without any modifications to the algorithmic framework.

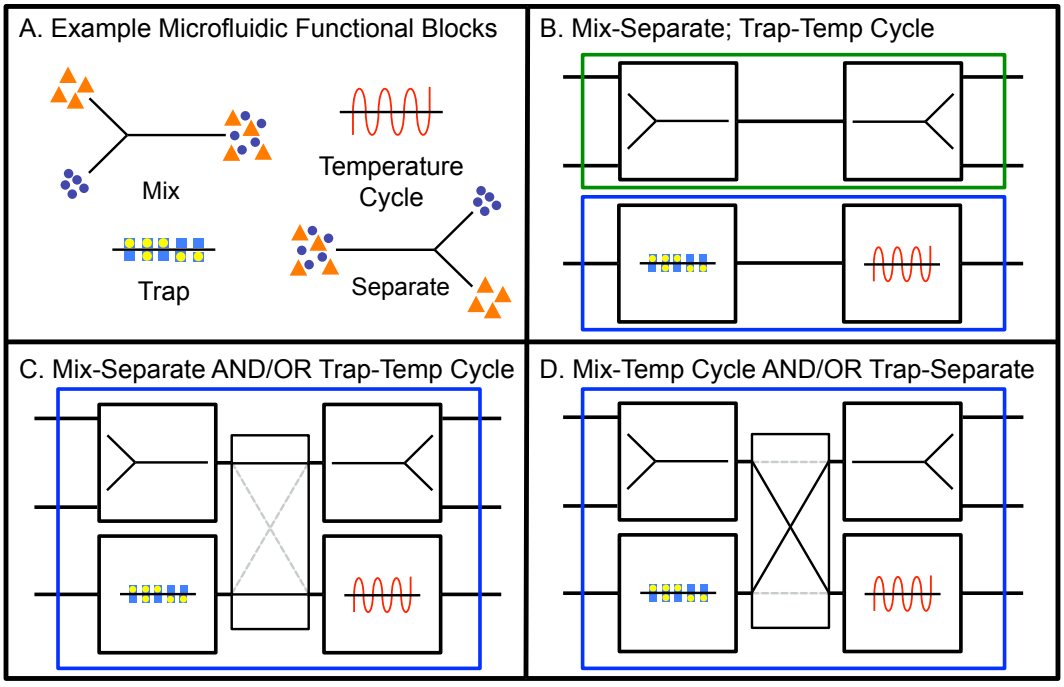


Figure 4·10: Example architecture utilizing a single transposer. Microfluidic functional blocks (A.) are connected via static channels to form two separate devices (B.) the functions of which are immutable. One device can perform mixing and then separation, while the other can perform cell trapping and then temperature cycling. A transposer is used to join the functional blocks in lieu of static channels (C. and D.). This results in two novel functions, mix then temperature cycle and trap then separate (D.). Additionally, The single microfluidic chip is able to perform both functions shown in C. simultaneously as well as both functions in D.

4.7.5 Dynamic Reconfiguration

Integrating intermediate sensing operations as functional blocks within a transposer-based microfluidic routing architecture unlocks a unique ability to dynamically reroute

fluids based on real-time measurements. This is particularly valuable in experiments involving fluidic operations that depend on dynamic fluidic variables such as directed evolution (Wang et al., 2014), refinement (Swain et al., 2013) and genetic logic (Tamsir et al., 2011). However, reconfiguration of the routing fabric during run time carries a distinct limitation whereby at the moment of reconfiguration, a contaminated plug of liquid will propagate through the device. The problem is analogous to the concept in digital electronics of a dynamic discipline (Harris and Harris, 2013). The dynamic discipline affirms that the output of a digital logic element (ex. a flip flop) is not deterministic if queried inside of its established propagation delay. Extending this analogy to microfluidic routing, it can be stated that if a transposition occurs within channels containing functionally orthogonal fluids (so as to prevent permanent contamination downstream) that the output of the system will be valid only after waiting for the contaminated plugs to propagate through the device.

4.7.6 Future Work

One particular strength of this work is the technology-agnostic manner in which the framework was built. This means that the transposer primitive can be optimized to suit any control or fabrication environment so long as the basic functionality shown in Figure 4·2 is retained. The optimal transposer primitive may look dramatically different between experimentalists performing separation using acoustophoresis (Petersson et al., 2007) versus filtration (Pamme, 2007), yet the routing framework would be identical.

Since each primitive only requires two control lines, which are of equipotential and toggled in a manner similar to the microfluidic multiplexor previously described (Thorsen et al., 2002), the optimization of control lines based on variable routing constraints is a solvable problem.

Finally, the formulation of a flexible microfluidic architecture that encompasses a maximum subset of microfluidic assays is under development. This chip would allow experimentalists to fabricate devices in bulk, yet maintain maximum flexibility to perform a large range of assays and experiments using the same chip. Organic software will accompany this architecture and serve to integrate device-level microfluidic control with assay-level specification. This effectively raises the level of abstraction for the microfluidic experimentalist and frees the community from the need to design, fabricate, and control singular microfluidic architectures for each experiment; rather, the experimentalist will reprogram the same chip architecture for many types of experiments through the same control environment.

4.8 Conclusions

This work demonstrates a novel microfluidic routing fabric for continuous-flow devices using a scalable primitive called a transposer. The fabric exists independent of any optimizations made to the primitive itself. We proved that fluidic routing through the fabric is extensible and developed algorithms to do so. We then integrated the fabric into a larger architecture towards the development of a programmable continuous-flow microfluidic device akin to the class of electronic devices known as FPGAs.

The barrier to entry for continuous-flow microfluidics can be prohibitively high, yet the benefits of microfluidic technology are too good to ignore (Whitesides, 2006). This work serves to introduce a new class of programmable microfluidic devices aimed at decreasing the design, fabrication, and operational costs of continuous-flow microfluidics thus increasing the accessibility of microfluidic technology. It is our hope that programmable continuous-flow microfluidics could serve as the catalyst towards microfluidic ubiquity as programmable electronics was for electronic computing.

Chapter 5

Rapid prototyping and parametric optimization of plastic acoustofluidic devices for blood–bacteria separation

5.1 Introduction

Acoustic manipulation has emerged as a versatile method for microfluidic separation and concentration of particles and cells. Most recent demonstrations of the technology use piezoelectric actuators to excite resonant modes in silicon or glass microchannels. Here, we focus on acoustic manipulation in disposable, plastic microchannels in order to enable a low-cost processing tool for point-of-care diagnostics. Unfortunately, the performance of resonant acoustofluidic devices in plastic is hampered by a lack of a predictive model . In this manuscript, we build and test a plastic blood-bacteria separation device informed by a design of experiments approach, parametric rapid prototyping, and screening by image-processing. We demonstrate that the new device geometry can separate bacteria from blood while operating at 275% greater flow rate as well as reduce the power requirement by 82% , while maintaining equivalent separation performance and resolution when compared to the previously published plastic acoustofluidic separation device.

5.2 Problem Statment

Separation and concentration of particles and cells via microfluidic acoustic manipulation has emerged as a versatile method for rapid and efficient fluid processing. It is an attractive alternative over other fluid manipulation techniques because it is label-free, requires no electrodes or specialized structures in the microchannel, and has the potential for scale-up for high throughput processing (Antfolk and Laurell, 2017)(Bhagat et al., 2010). In so-called "bulk" acoustic microfluidic devices, the acoustophoretic force is maximized as the fluid-filled microchannel resonates as a cavity and establishes a standing pressure wave transverse to flow. Hence, the magnitude of the acoustic force on a particle depends strongly on the physical dimensions of the channel and walls, which must be appropriately selected for the ultrasonic excitation frequency (Bruus, 2012).

The force exerted on a particle by the acoustically-driven standing pressure wave scales cubically with particle diameter (Settnes and Bruus, 2012). Thus, acoustic manipulation is particularly well suited for isolating bacterial samples from larger blood components such as red blood cells (RBCs) and white blood cells (WBCs) based purely on relative differences in size (Ohlsson et al., 2016)(Li et al., 2016). This is useful as downstream assays, such as antimicrobial susceptibility testing, benefit from a purified, well-defined input with reduced contamination of mammalian cell components.

Silicon, glass, or metal devices are commonly used for acoustophoresis because the rigid channel walls provide a near ideal acoustic boundary against the sample fluid, enhancing the required standing wave resonance (Barnkob and Bruus, 2009)(Hill et al., 2008). This ideal boundary simplifies design because one-dimensional analysis can be used to estimate the resonant modes in the channel-fluid system. However, the rigid materials used in these devices are relatively expensive and slow to manufacture, have

limited compatibility with many biological samples, and pose challenges to produce as disposable laboratory tools (Nge et al., 2013). On the other hand, our recent work has demonstrated acoustophoresis in plastic, showing that acoustic separation of RBCs is possible in polystyrene, opening the door to low-cost diagnostic and therapeutic devices (Mueller et al., 2013).

However, the design of optimized plastic acoustofluidic devices is hampered by complex boundary conditions relative to those of more rigid materials, and a satisfactory predictive model is not yet established, because the one-dimensional approximations are inaccurate and the channel walls can no longer be considered ideally rigid (Mueller et al., 2013). Moreover, even for rigid materials, a sophisticated two-dimensional analysis does not appear to easily predict experimental performance in detail (Garofalo et al., 2016)(Bora and Shusteff, 2015). To further optimize the geometry of acoustic microfluidics in plastic, a parametrized experimental investigation provides an expanded database for comparison with simulation and can give performance improvements without the need for simulation.

This study used rapid prototyping, statistical design of experiments, and rapid experimental screening to obtain a better-performing device geometry when compared to the only other plastic, acoustofluidic blood–bacteria separation device in literature (Mueller et al., 2013), which will be referred to as the *baseline*. The baseline was designed in accordance with the existing one-dimensional hard-wall theory and no further optimization had been attempted. The devices tested varied in cross section dimensions.

While plastic acoustofluidic devices can be produced in volume using methods such as hot embossing or injection molding (Heckele and Schomburg, 2003), prototyping test geometries in small batches could become a costly endeavor. We minimized fabrication costs by manufacturing chips on-site through the use of an automated,

rapid prototyping software framework in conjunction with a new class of inexpensive, desktop computer numerical controlled (CNC) micromills.

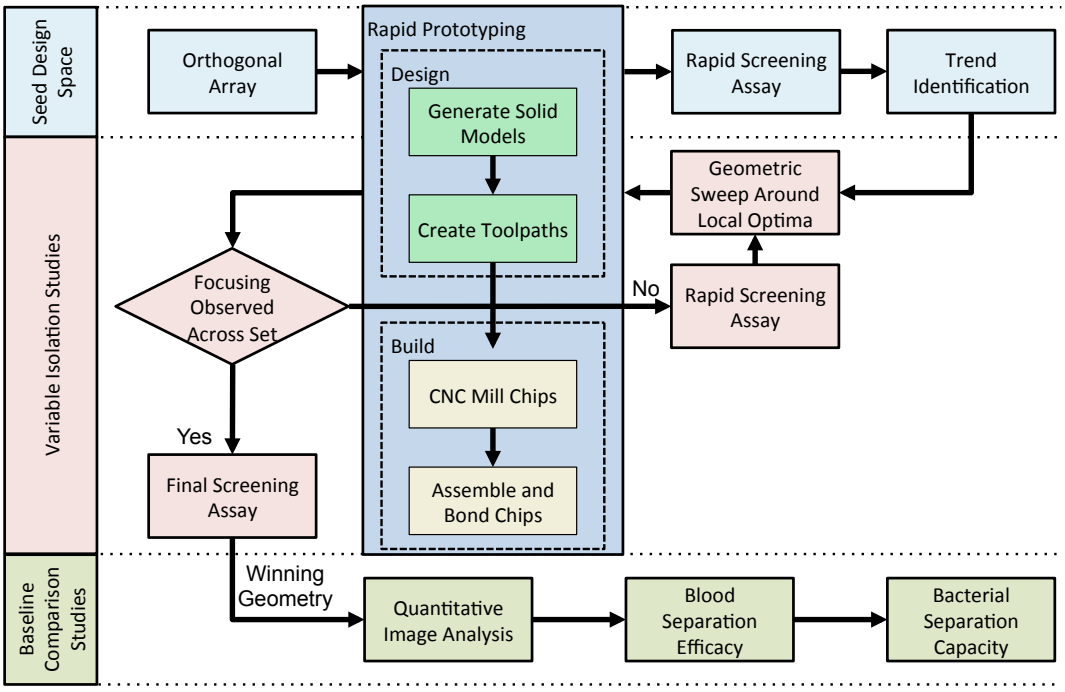


Figure 5·1: Iterative rapid prototyping and testing workflow. The design space was first seeded using a design of experiments tool known as an orthogonal array, which minimizes the number of necessary experiments when compared to a full-factorial experimental analysis (Section 5.5.2). Trends were then identified within single geometric variables and experimentally explored. These variable isolation studies were repeated until a fully functional design set was achieved (Section 5.5.3). The best performing device geometry emerging from the workflow, Chip 2.0, is then compared to the baseline using image processing, blood separation, and, finally, bacterial separation tests (Section 5.5.4).

Devices were screened in rapid succession using an image-based performance parameter of RBC acoustophoresis, described in Section 5.4.2, while varying two measures of merit: dissipated power and volumetric flow rate. As a final validation, the improved design was compared to the baseline in the task of separating bacteria from blood and shown to achieve comparable separation with significant advantages in the figures of merit. This improved device offers increased throughput and reduced

power requirements and could improve performance in future point-of-care plastic acoustofluidic devices.

Figure 5·1 illustrates the iterative workflow used in conducting the study, further described in Section 5.3. Section 5.3 summarizes the approach used to design variable chip geometries and defines the two types of tests used in the screening phase of the workflow. Section 5.4 outlines the methods used to screen separation performance from microscope images. Finally, Section 5.5 presents the results of the device screening as well as the winning design's performance when compared to that of the baseline geometry.

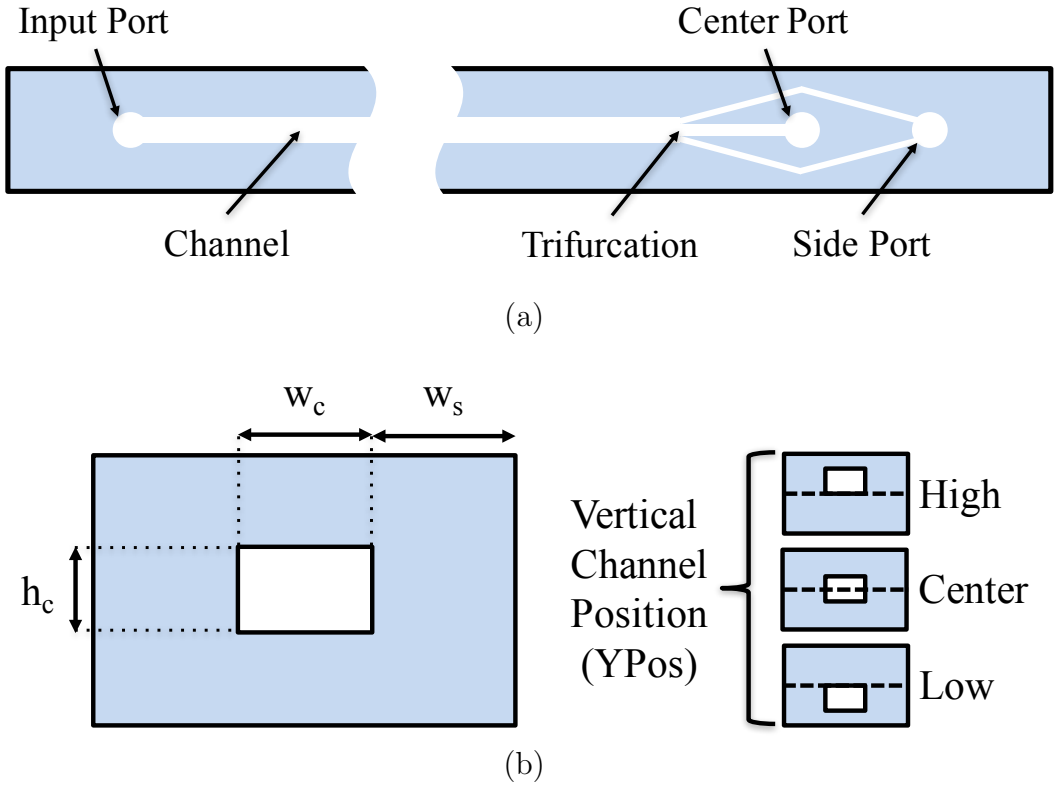


Figure 5·2: Acoustic separation device. (a) Complete trifurcated microfluidic separation chip. (b) Definitions for each two-dimensional geometry included in the study. Note that the definitions apply to the fluidic channel upstream of the trifurcation.

Figure 5·2a. is a top-down, two-dimensional drawing of the trifurcated acoustic separation device. The device functions by focusing large particles, such as RBCs and WBCs, to the center port, while smaller particles (e.g., platelets, bacteria, etc.) are collected at the side port. This trifurcated device is used in the blood and bacterial separation experiments. In order to reduce manufacturing complexity, devices screened using the methods outlined in Sections 5.3.2 and 5.3.2 consist only of the input port; fluid channel, defined as the channel upstream of the trifurcation; and a single output port. The cross-sectional geometries, defined in Figure 5·2b., apply to this simplified device design.

5.3 Experimental Methodology

5.3.1 Rapid Prototyping

This study relied upon the ability to design and fabricate iterations of chip geometries in a rapid process informed by experimental tests. The traditional workflow of conventional machining requires a fully specified mechanical drawing and changes to the design may demand regeneration of the solid model and revised setup of the milling instrument. This section describes how these limitations were mitigated using free and open-source design software in conjunction with a $3,199 USD desktop micromill (Othermill Pro, Other Machine Co., Berkeley, CA, USA). Microchannels with parameterized dimensions were systematically fabricated with a minimum of operator intervention.

Listing 5.1: The custom OpenSCAD library allows solid model creation using just a single line of code

```
chip(Wc=0.55,Hc=0.25,Ws=0.85,YPos=high);
```

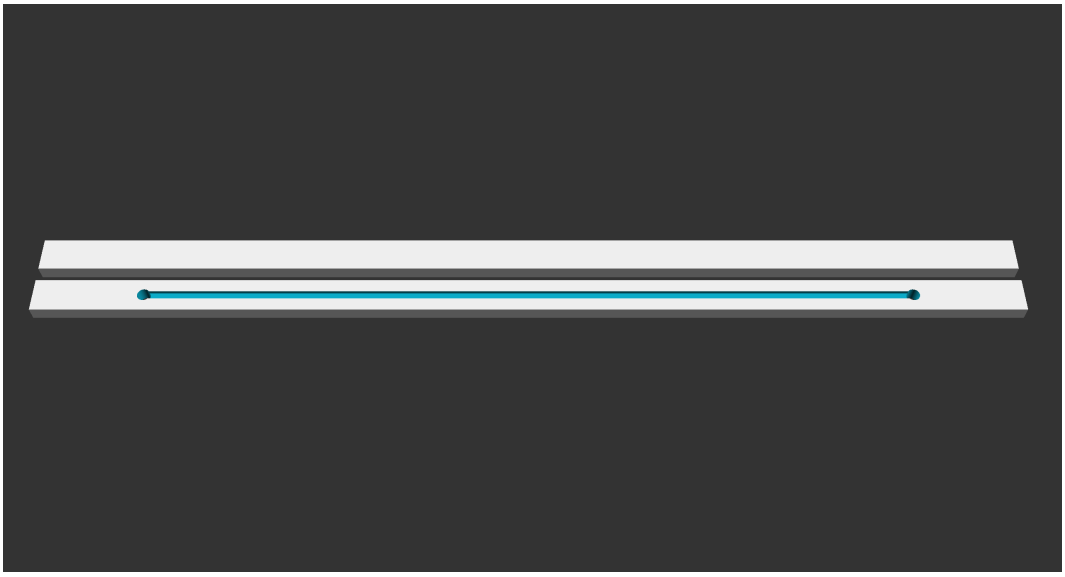


Figure 5·3: Output solid geometry of Listing 1, including bonding plate.

Listing 5.2: The single line of code in Listing 5.1 can then be iterated upon to form an array of different device geometries that can be sent directly to a CAM tool for toolpath generation

```
include<chip.scad>

wc=[1.5,1,0.5];
ws=[0.5,1.5,2.5];
hc=[0.1,0.2,0.25];
ypos=[high,high,high];
Spacing=0.79375;            //End Mill Diameter

numChips=len(wc);           //Total chips to make

module chipLayout(){
  //Iterate over number of chips
  for(i=[0:(numChips-1)]){
    let (ChipY=wc[i]+ws[i]*2){
      //Correctly space chips apart
      translate([0,2*i*(ChipY+Spacing),0])
      chip(Wc=wc[i],Hc=hc[i],Ws=ws[i],YPos=ypos[i]);
    }
  }
}
```

Figure 5·4: Output solid geometry of Listing 2, which includes three different designs and their corresponding bonding plates.

Design Generation

Three levels of software are required to design and fabricate a novel chip geometry using a computer numerical control (CNC) micromill: a computer aided design (CAD) tool is used to create a solid model of the device; computer aided manufacturing (CAM) software generates the commands (also called toolpaths) that are sent directly to the micromill; and control software manages the connection between a computer and the micromill and sends individual toolpaths to the micromill.

Device designs were created using OpenSCAD (Wikibooks, 2017), a free and open source CAD tool that reads script files to generate solid models. A custom library was used to create solid models using just the geometric parameters outlined in Figure 5·2b. as inputs. Figure 5·4 shows how an array of distinct solid models can be created from a few lines of code shown in Listing 5.2. This array of designs is spaced according the size of the endmill used by the CNC to cut out each design, thus allowing for seamless processing by CAM software (Autodesk Fusion 360).

Fabrication

Micromilling has demonstrated advantages for low-volume prototyping of plastic microfluidic devices in terms of time and cost when compared to other fabrication methods such as embossing and injection molding (Guckenberger et al., 2015). While such studies claim that micromilling devices using an outside source can lower costs to $137 per batch and 11-15 days of turn-around time, these costs only consider material costs and not labor, which can drive up the cost of prototyping an order of magnitude (Guckenberger et al., 2015). Costs per device can be lowered to less than $1 if devices can be fabricated in-house; however, the costs associated with establishing such capabilities can be prohibitive. It is only very recently that high quality micromills have been available at low cost: As recently as 2015, micromills capable of achieving resolutions at or below 25μm were available for a minimum of $15k (Guckenberger et al., 2015). The large footprints and noise associated with these machines made them inappropriate for use within a microfluidics laboratory. Additionally, the cost in terms of expertise required to operate a micromill is non-trivial. The software stack associated with generating toolpaths for a micromill does not currently resemble the simplicity of other CNC machines such as 3D printers. Traditional micromilling requires a suite of CAM tools that demand extensive knowledge of various tooling strategies such as feed rate, depth of cut and spindle speed that vary with each material and tool size.

Recent advances in micromilling has led to the formation of a new class of desktop micromills, which can approach 25μm resolution at costs starting at $2,500 all in a form-factor appropriate for a typical laboratory bench (Yen et al., 2016). While these new machines still require knowledge of CAD and CAM software tools, research in automation techniques specifically for micromilling microfluidics is beginning to bear fruit (Silva et al., 2016)(McDaniel et al., 2017). This study leverages these advance-

ments to quickly manufacture distinct acoustofluidic device designs at a negligible cost when compared to outsourcing fabrication.

5.3.2 Device Evaluation

We organized this study into four types of experiments: rapid screening, final screening, blood separation, and bacterial separation, progressing toward the selected design and then validating its performance in a series of functional tests. Rapid screening tests were conducted on two initial sets of designs for which the functionality of each design in the set could not be assumed, as shown in Figure 5·1. The methodology for this test is outlined in Section 5.3.2. The results of these two initial rapid screening tests were then used to inform the parameter set of a final, more involved, screening test described in Section 5.3.2. Next, the winning geometry after final screening, hereafter referred to as Chip 2.0, was compared to that of the baseline geometry in two different experiments measuring a device's ability to focus RBCs. Finally, Chip 2.0 was compared to the baseline in experiments separating bacteria from diluted whole blood. The methodologies for each experiment are outlined in the subsections below.

The screening tests analyzed microscope images and derived pixel intensities as an indicator of RBC focusing performance. This method assumes that higher concentrations of RBCs will appear visibly darker, and hence have a higher inverse grayscale value, when better acoustic focusing is present (Barnkob et al., 2012). Figures 5·5 and 5·6, demonstrate that in conditions conducive to good focusing (e.g., lower flow rate and higher dissipated power) the observable band of blood cells will appear narrower and darker and thus have a correspondingly higher inverse grayscale value within this band.

The two types of screening tests, rapid screening and final screening, each have

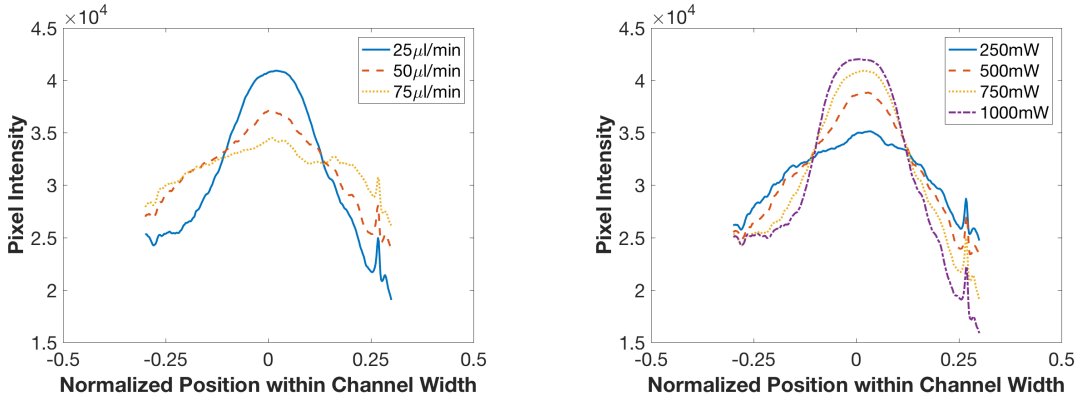
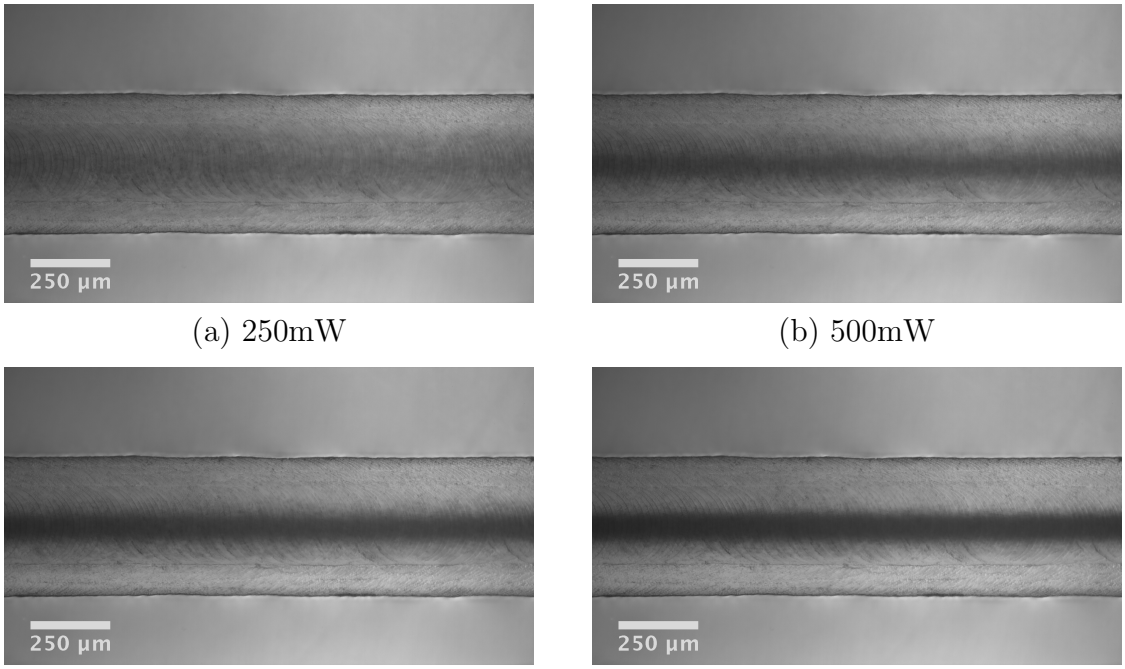


Figure 5·5: Pixel intensity as a surrogate for RBC focusing performance. Inverted pixel grayscale values (i.e., darker areas have a higher value) across the width of the fluidic channel. (a) Power is held constant at 750 mW while flow rate is varied. Note that maximum pixel intensity, and thus focusing performance, increases as flow rate is decreased. (b) Flow rate is held constant at 25 μl/min, while power is varied. Note that focusing performance increases as power is increased.

different assumptions regarding the nature of their input parameter sets. In the rapid screening test a device may not exhibit acoustic focusing at any frequency under the experimental conditions; thus, the performance parameter determines the existence of acoustic focusing, but does not discern the relative quality of focusing between devices. In contrast, the final screening test attempts to compare the relative performance of devices that exhibit some acceptable measure of focusing performance.

The latter separation experiments measured a device's ability to maintain cell separation performance while minimizing dissipated power to the transducer (i.e. amplitude of acoustic excitation) and maximizing sample throughput (i.e., volumetric flow rate). Maximizing the volumetric flow rate will enable the largest volume of input sample to be enriched in the shortest amount of time. Minimizing the power requirements of the system is another important measure of merit because heat generated in the transducer and in the PS channel during actuation may lead to delamination



(a) 250mW (b) 500mW

(c) 750mW (d) 1000mW

Figure 5·6: Microscope images of focusing performance at resonant frequency with increasing power at levels of 250mW (a), 500mW (b), 750mW (c), 1000mW (d). The plots in Figure 5·5(b) are derived directly from these images.

of the channel or may be harmful to the clinical sample. Therefore, it is best to drive the system at the smallest amplitude necessary to achieve acceptable performance.

Rapid Screening Test

The purpose of the rapid screening test was to discover functional device designs, i.e., designs that could focus blood, and the frequencies at which they operate. Functionality of an acoustofluidic device was determined by analyzing the strength of focusing bands whose maxima occurs within the center fifth of the channel width, as the purpose of the device is to focus RBCs into the center channel of the trifurcation shown in Figure 5·2. Thus, in order to affirm the decision node labeled "Focusing Observed Across Set" in Figure 5·1, the point of maximum pixel intensity must be present in

the center fifth of the channel for the device under test.

Each set of devices was tested under the same power and flow conditions while sweeping through a frequency band of 0.50 – 2.00 MHz. This range was informed by the optimum frequency of the baseline device (1.012 MHz). The transducer was selected with a resonant frequency of 2.34 MHz to avoid confounding effects of transducer resonance. The flow rate was set such that the average velocity in each chip was 1.94 cm/sec, a value that is equivalent to 100 μl/min in the baseline device. The input power to the RF amplifier was set such that the average dissipated power at the transducer was 1 W at 1 MHz. Electrical equipment settings (e.g., input voltage, RF amplifier gain, etc.) for this condition (i.e., 1 W at 1 MHz) were held constant as the frequency was varied. Within the tested bandwidth photos of the downstream end of the channel were taken in 10 kHz increments, from which a number corresponding to the maximum peak prominence was returned. Prominence is a measure of a "peak" in pixel intensity and is defined further in the Methods section. The maximum peak prominence for the entire frequency band constituted the score for that particular design. An example frequency sweep is shown in Figure 5·7.

Final Screening Test

The final screening test served to discriminate between devices that exhibited RBC focusing on the rapid screening test. This was accomplished by manually tuning the device to its optimum frequency (by visual inspection of the RBC focusing) and then modulating the measures of merit. Three flow rates (25 μl/min, 50 μl/min, and 75 μl/min) and four power levels (250 mW, 500 mW, 750 mW, and 1000 mW) were studied. A microscope image was captured for each combination of flow rate and power level, from which the ratio of peak prominence to width was calculated in accordance with the method described in Section 5.4.3. The ratio of peak prominence

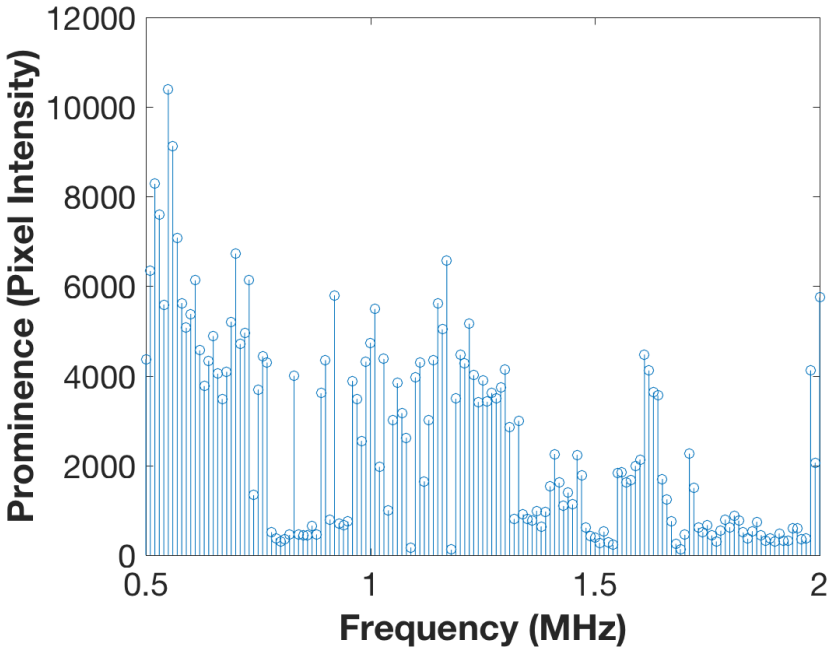


Figure 5·7: Prominence versus Frequency for a fixed input voltage for device 4L9 in Table 5.1. Images were captured in 10 kHz increments over the bandwith extending from 0.5 MHz to 2 MHz, inclusive.

to width serves as a predictor of the device's ability to separate RBCs.

Blood Separation Test

The purpose of the blood separation test is to measure a device's ability to focus RBCs to the center port of the device at a series of combinations of flow rate and power settings. A test of this sort has proven useful in assessing or comparing the acoustic energy density in several device configurations for applications ranging from high-throughput cell sorting (Adams et al., 2012)(Mueller et al., 2013) to plasmapheresis (Lenshof et al., 2009). Samples of dilute whole blood were collected from both side and center ports for each setting according to the protocol established in Section 5.4.7. Performance is measured using RBC separation percentage, calculated by dividing the number of RBCs collected from the center port and dividing by the total number of

RBCs collected from both outlet ports.

Bacterial Separation Test

The goal of the bacterial separation experiments is to determine the maximum capacity of a chip design in its ability to separate bacteria from blood, as has been demonstrated in other acoustic microfluidic devices (Ohlsson et al., 2016)(Li et al., 2016). The operational capacity of a chip is evaluated in terms of the two measures of merit: volumetric flow rate and average dissipated power, each tested with the other fixed. Bacterial separation is measured at the point of minimum power or maximum flow rate required to achieve 90% separation of RBCs, measured by the ratio of colony forming units (CFU) to RBCs present in the side port. Where power was varied, flow rate was held at 50 μl/min. Where flow rate was varied, power was held at 1 W.

5.4 Methods

5.4.1 Materials and Assembly

Polystyrene (PS) was selected as an appropriate material based on its relatively low attenuation and high acoustic impedance relative to other plastics (Mueller et al., 2013)(Selfridge, 1985). The chip was sealed using a thermal bonding method previously described (Mueller et al., 2013). 0.75" lengths of polyetheretherketone (PEEK) tubing served as an interface between the PS chip and the longer lengths of polyvinyl chloride (PVC) tubing used to introduce and collect sample. The rigid PEEK tubing was inserted into machined port cavities and affixed to the chip using epoxy (Epoxy 907, Miller-Stephenson, Danbury, CT, USA).

Lengths of vinyl capillary tubing were appended to the outlet ports to divide outlet volumes at a ratio of 60% volume measured at the side port to 40% at the center, using relative resistances.

The sealed chip was mounted to a lead zirconate titanate (PZT) transducer (APC International, PZT 850) with a published resonance of 2.34 MHz using low viscosity cyanoacrylate adhesive.

5.4.2 Image Processing and Analysis

Image processing software (ImageJ (Abràmoff et al., 2004)) was used to measure pixel intensities across the width of the channel, W_c, from which prominence was calculated. The prominence of the focused stream, as described in detail in Section 5.4.2 below, served as a measure of the degree of focusing and the performance of the chip. Prominence has advantages over raw pixel intensity for the purposes of comparison due to its self-normalizing nature. Since prominence is measured relative to points on the signal itself it is robust against irregularities inherent to the signal. These irregularities can take the form of variable lighting conditions between experimental runs, such as variations in environmental lighting, and illumination variabilities within a single microscope image's region of interest, such as skewed background intensities caused by shadows.

Definition of Prominence

Suppose an ordered signal is defined as in Equation 5.1, where set D consists of N data points. Prominence is calculated by first finding all local maxima within the response set D and then determining a reference point on the signal associated with each local maxima (Freeman and Davis, 1977)(Arge et al., 2013). Briefly, this reference point is established by drawing a horizontal line in both the positive and negative directions from the local maxima (labeled as "Scan High" and "Scan Low", respectively, in Figure 5·8) until either the end of the signal is reached (as in the case of i_{high}, in Figure 5·8) or until the line intersects the signal itself (i_{low}, in Figure 5·8), thus creating two sets of data points in the positive and negative directions. Minima

Algorithm 5: Find Peaks. Finds all local maxima in D and returns them as a list of peaks P as shown in Figure 5·8(a).

```
Data: Ordered data set D
Result: A list of peaks P ⊂ D
begin
    P ⟵ ∅
    for d_i ∈ D do
        Find all neighbors for d_i | |i − i'| = 1 AppendToP(neighbors)
    end
end
```

are determined for each of the data sets, and prominence is then defined as the height of the local maxima relative to the maximum of these two established minima (s_{ref}).

$$s[n] = D \quad n = 0, 1, 2, \cdots, N-1 \tag{5.1}$$

A framework for defining prominence in a more formal terms begins by establishing the data structure for the signal of interest. Individual values are accessed by index such that $s[i] = d_i, d_i \in D$. Two discrete points $d_i \in D$ and $d_{i'} \in D$ are said to be *neighbors* if $|i - i'| = 1$. A local maxima, hereafter called a *peak*, is a point p where d_i is greater than all of its neighbors, as defined in Algorithm 5. The set of peaks $P \subset D$ consists of individual peak values $p \in P$. A peak is referenced by its index value i in D and the raw peak height is calculated by finding $s[i]$. The index in D of peak p is returned by the function $x(p)$. The height of peak p is returned by the function $s[x(p)]$. Thus if $p \leftarrow d_i$, $x(p) = i$ and $s[x(p)] = d_i$. Prominence is then determined via Algorithm 6.

5.4.3 Definition of Prominence to Width Ratio, χ

The half-prominence width of a peak of prominence *Prom* is calculated by drawing two horizontal lines extending in the negative and positive directions from the point

Algorithm 6: Calculate Prominence. Draw a horizontal line to the left (low) and right (high) of the peak until the end of the signal is reached or until the signal is intersected. Record the indices of each as i_{low} and i_{high}, respectively. Find the minima in each set and use the maximum of the minima to set the reference level. Calculate a peak's prominence by subtracting the reference level from the raw signal value of the peak (Figure 5·8(b-c))

Data: Ordered data set D and a list of local maxima $P \subset D$
Result: A list of prominence values, $Prom$, for each local maxima, p
begin
  $Prom \longleftarrow \emptyset$
  for $p_j \in P$ **do**
    `/* Scan Low */`
    for $i = 0$ *to* $x(p_j)$ **do**
      if $x(p_j) = 0$ **then**
        $i_{low} = x(p_j)$
        Exit Loop
      end
      else if $s[x(p_j) - i] \geq s[x(p_j)] \vee x(p_j) - i = 0$ **then**
        $i_{low} = x(p_j) - i$
        Exit Loop
      end
    end
    for $i = 0$ *to* $N - 1$ **do**
      if $x(p_j) = N - 1$ **then**
        $i_{high} = x(p_j)$
        Exit Loop
      end
      else if $s[x(p_j) + i] \geq s[x(p_j)] \vee x(p_j) + i = N - 1$ **then**
        $i_{high} = x(p_j) + i$
        Exit Loop
      end
    end
    $s_{low} = min(s[i_{low}] \rightarrow s[x(p_j)])$
    $s_{high} = min(s[x(p_j)] \rightarrow s[i_{high}])$
    $s_{ref} = max(s_{low}, s_{high})$
    $Prom_j = s[x(p_j)] - s_{ref}$
    `AppendToWidth`$(Prom_j)$
  end
end

of half-prominence. These lines extend in either direction until either the end of the signal is reached or the line intersects the signal itself. The indices of these events in the negative and positive directions are recorded as i^- and i^+, respectively. The peak width $HalfPromWidth$ is then defined as $|i^+ - i^-|$, as described in Algorithm 7. We reason that for equivalent separation performance, assuming a fixed ratio of flow to side and center ports (Ley and Bruus, 2016), that the prominence half width scales with the channel width, therefore it is appropriate to normalize the peak width by the width of the channel. The final equation for χ is shown in Equation 5.2

$$\chi = Prom * \frac{W_c}{HalfPromWidth} \tag{5.2}$$

5.4.4 Transducer Drive

The sinusoidal signal used to drive the transducer is generated by a function generator (AFG3022C, Tektronix, Beaverton, OR, USA) and amplified using a broadband RF amplifier (AG1021, T&C Power Conversion, Rochester, NY, USA). The instantaneous voltage and current across the transducer is monitored using an oscilloscope (DPO2024B, Tektronix, Beaverton, OR, USA). In order to determine the actual power consumed by the transducer, it is first necessary to consider the instantaneous power as follows:

$$P_{inst} = VI = V_{max} sin(\omega t) I_{max} sin(\omega t - \varphi), \tag{5.3}$$

where V_{max} and I_{max} are the maximum values of voltage and current, φ is the phase lag between the instantaneous current and voltage signals, and $\omega = 2\pi f$ is the sinusoidal drive frequency in rad/s. Using trigonometric identities and integrating over a cycle of the sinusoid we compute the average consumed power as:

Algorithm 7: Calculate Peak Width at Half Prominence. Draw a horizontal line to the left (-) and right (+) at the median of the prominence line until either the end of the signal is reached or until the signal is intersected. Record the indices of each as i^- and i^+, respectively. The absolute value of the difference between these index values is the peak width at half prominence. 5·8(d))

Data: A list of prominence values, $Prom$, for each local maxima, p
Result: A list of half-prominence peak widths $HalfPromWidth$, for each local maxima, p

begin
  $Prom \longleftarrow \emptyset$
  for $p_k \in P$ **do**
    `/* Scan Left */`
    for $i = 0$ *to* $x(p_k)$ **do**
      if $x(p_k) = 0$ **then**
        $i^- = x(p_k)$
        Exit Loop
      end
      else if $s[x(p_k) - i] \leq s[x(p_k)] - \frac{Prom_k}{2} \vee x(p_k) - i = 0$ **then**
        $i^- = x(p_k) - i$
        Exit Loop
      end
    end
    `/* Scan Right */`
    for $i = 0$ *to* $N - 1$ **do**
      if $x(p_k) = N - 1$ **then**
        $i^+ = x(p_k)$
        Exit Loop
      end
      else if $s[x(p_k) + i] \leq s[x(p_k)] - \frac{Prom_k}{2} \vee x(p_k) + i = N - 1$ **then**
        $i^+ = x(p_k) + i$
        Exit Loop
      end
    end
    $HalfPromWidth_k = i^+ - i^-$
    `AppendToWidth`$(HalfPromWidth_k)$
  end
end

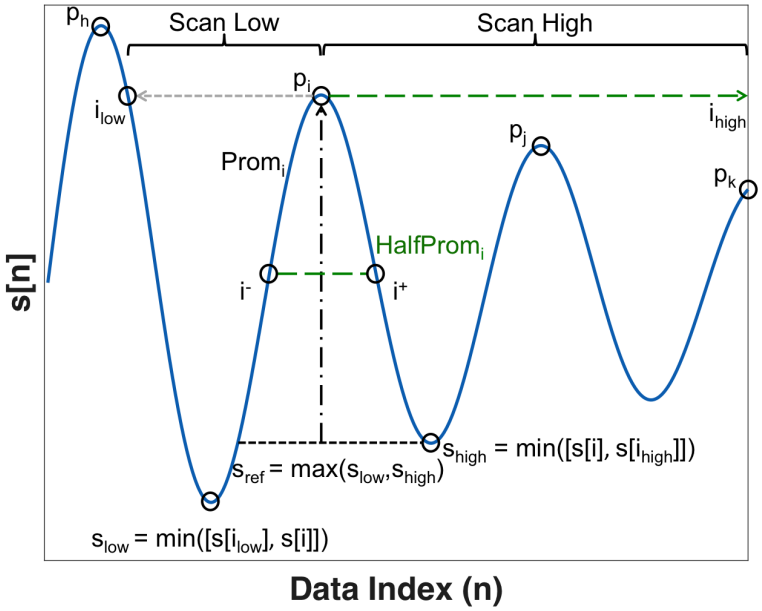


Figure 5·8: Algorithmic progression. The blue, solid line represents a signal, $s[n]$, sampled at points, n, along the x-axis. Local maxima are labeled as points $p_{\{h,i,j,k\}}$. s_{ref} marks the reference point from which the prominence, $Prom$, of peak p_i is calculated. The width of the signal at the point of half-prominence, $HalfPromWidth$, is calculated by subtracting i^- from i^+.

$$P_{avg} = V_{rms} I_{rms} cos\varphi, \tag{5.4}$$

where V_{rms} and I_{rms} are the root mean square values of voltage and current. Using the oscilloscope, we multiply the instantaneous voltage and current and compute the average of this product to find the average consumed power in real time throughout our experiments.

Voltages and currents used to achieve 1 W of dissipated power at resonant frequencies for the baseline geometry and Chip 2.0 are shown in Table 5.4. Note that the drive is not optimized for impedance matching, therefore most power is reflected.

5.4.5 Blood Sample Preparation

All experiments in this study used de-identified fresh human whole blood purchased from a vendor (Research Blood Components, Brighton MA), anticoagulated with acidcitratedextrose. In each case, the blood was diluted to 5% by volume (for rapid screening tests) or 5% hematocrit (for all other tests) in phosphate buffer solution (PBS 7.4 pH Lot Number 1832496). Cellular concentrations were measured before and after dilution using an automated hematology analyzer (XP-300, Sysmex Co., Kobe, Japan). The diluted sample was then transferred to a 10 ml plastic syringe (BD 10 ml Luer-Lok tip syringe 309604) and introduced to the chip through PVC tubing. The volumetric flow rate was regulated by a syringe pump (PhD Ultra, Harvard Apparatus).

5.4.6 Bacteria–Blood Sample Preparation

Pseudomonas aerouginosa was incubated overnight in a Lysogeny Broth (LB) culture. It was diluted by a factor of 50 and incubated until it reached a mid log phase. A whole blood sample was diluted into PBS as described in Section 5.3.2. The optical density of the *Pseudomonas* culture was taken and the appropriate dilution was calculated to create solution consisting of whole blood diluted in PBS to 5% hematocrit and 10^5 *Pseudomonas* cells/ml.

5.4.7 Sample Measurement

Blood or blood bacteria solutions were pumped through the device at its previously determined resonant frequency. For each device this was found visually by observing the focusing stream–the most compact stream reveals the resonant frequency. Outlet samples were collected in conical tubes after which they were measured and weighed for flow fraction and cell quantity calculations.

Blood content was measured via a hematology analyzer while bacterial content was measured through a plating analysis: the samples were weighed, then serially diluted in 10x steps into PBS. Each dilution was cultured onto a plate of LB agar and incubated at 37°C overnight after which the CFU were counted.

The setpoint for 90% RBC separation for variable power and fixed flow (50 μl/min) rate was accomplished by increasing power in small increments. Samples were collected from each output port and RBC counts were measured after each power increment. This process was repeated until a 90% RBC separation ratio was achieved between the side and center ports.

Determining 90% RBC separation for variable flow rate and fixed power (1 W) required that the flow rate be set to 25 μl/min and increased in increments until such a point as 90% RBC separation was achieved.

5.5 Results

5.5.1 Screening Design of Experiments

Experiments must be initialized such that the sampling of the solution space can detect curvature in the response, as described in Section 5.5.2. Curvature in the response surface for an explanatory variable implies the existence of a local maxima in performance. Driving the design towards that local maxima is accomplished by isolating the variable in question and conducting experiments at adjacent points on the response plane, as outlined in Section 5.5.3.

Figure 5·2 illustrates the explanatory variables studied. Channel Width (W_c), channel height (H_c), side-wall width (W_s) and the position of the channel on the y-axis in two dimensional space (Y_{pos}) were selected as variables to study because they account for a majority of the two-dimensional design-space and are simple to vary during fabrication.

5.5.2 Seeding of the Design Space (Rapid Screening Test)

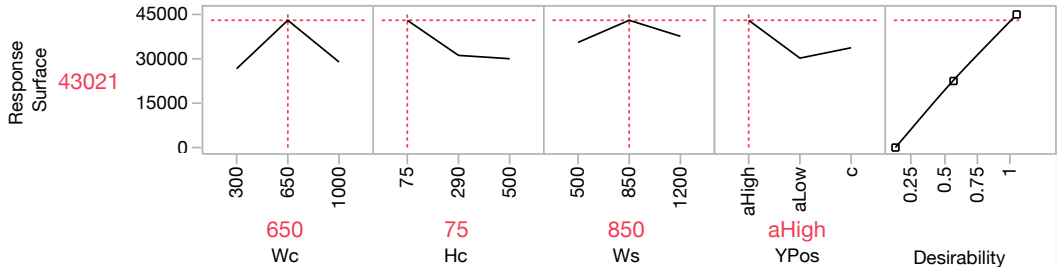


Figure 5·9: Results of design space seeding. The quantities on the vertical axis correspond to the prominence data shown in Table 5.1. The response surface is determined through a least squares fit of the aforementioned prominence values in four-dimensional space. The red dashed lines demarcate the points of maximum desirability.

In order to economize the number of rapid screening experimental runs, an orthogonal array was chosen to generate a useful parameter set for the initial experiments. Orthogonal arrays are are used in design optimization to provide the most coverage of the solution space while minimizing the number of experimental runs (Yokoyama et al., 1993). This is accomplished through the creation of an experimental set such that each combination of the array's strength appears equally often (Hedayat et al., 2012). The orthogonal array used to seed the design space is known as the $L_9(3^4)$ array, which has a strength of two and is used to probe a solution space consisting of four explanatory variables at three settings in nine experimental runs, as opposed to the 81 experimental runs required to conduct the full-factorial experiment (Hedayat et al., 2012). The use of three explanatory variable settings was chosen in order to detect curvature in the response surface. The parameter set tested while seeding the design space, in accordance with an $L_9(3^4)$ orthogonal array, is shown in Table 5.1.

The response surface generated from the data points shown in Table 5.1 was analyzed using statistical software (JMP, Version 13.0.0. SAS Institute Inc., Cary, NC, 1989-2017) and is shown in Figure 5·9. Maximum desirability of the performance

Table 5.1: Geometries tested for L9 design array with corresponding Chip ID. All dimensions are given in units of μm. Prominence values are given in units of pixel intensity and correspond to each chip's maximum prominence value for the tested bandwidth. The given frequencies indicate the point at which the maximum prominence was observed.

ID	W_c	H_c	W_s	Y_{Pos}	Prominence	$f\ (MHz)$
1L9	300	75	500	Low	6375	1.58
2L9	300	290	850	Center	5496	1.22
3L9	300	500	1200	High	8240	0.67
4L9	650	75	850	High	43021	0.55
5L9	650	290	1200	Low	12950	0.66
6L9	650	500	500	Center	13251	0.57
7L9	1000	75	1200	Center	14215	0.74
8L9	1000	290	500	High	9574	0.73
9L9	1000	500	850	Low	3144	0.71

parameter (prominence) is achieved by setting W_c to 650μm, H_c to 75μm, W_s to 850μm and placing the channel in the high vertical position (i.e., Chip 4L9 in Table 5.1). Additionally, the response surface shows significant curvature while modulating the width of the channel, leading into the next iteration of the study outlined in Section 5.5.3.

5.5.3 Variable Isolation Studies

Channel Width Study (Rapid Screening Test)

Proceeding from the results of the initial seeding of the design space, we fixed the other parameters from the best geometry (4L9) and varied the channel width in 50μm increments. Although the response surface indicates that thinner channel height may be preferable, we selected a height of 100μm for this variable isolation study, anticipating that a slightly greater channel height would have practical advantages.

As shown in Table 5.2, the highest performing channel width, holding other factors constant, was 550μm. Channels having a width of 600μm and above demonstrated the ability to focus RBCs to the center fifth of the channel, however the focusing was a result of driving the chip at a higher order mode thus creating more than one

Table 5.2: Geometries tested for an isolation study of channel width with corresponding Chip ID. All dimensions are given in units of μm. Prominence values are given in units of pixel intensity and correspond to each chip's maximum prominence value for the tested bandwidth. The given frequencies indicate the point at which the maximum prominence was observed. The frequency range scanned spanned from 500 kHz to 2 MHz. No focusing was observed within this range for the design with a channel width of 700μm.

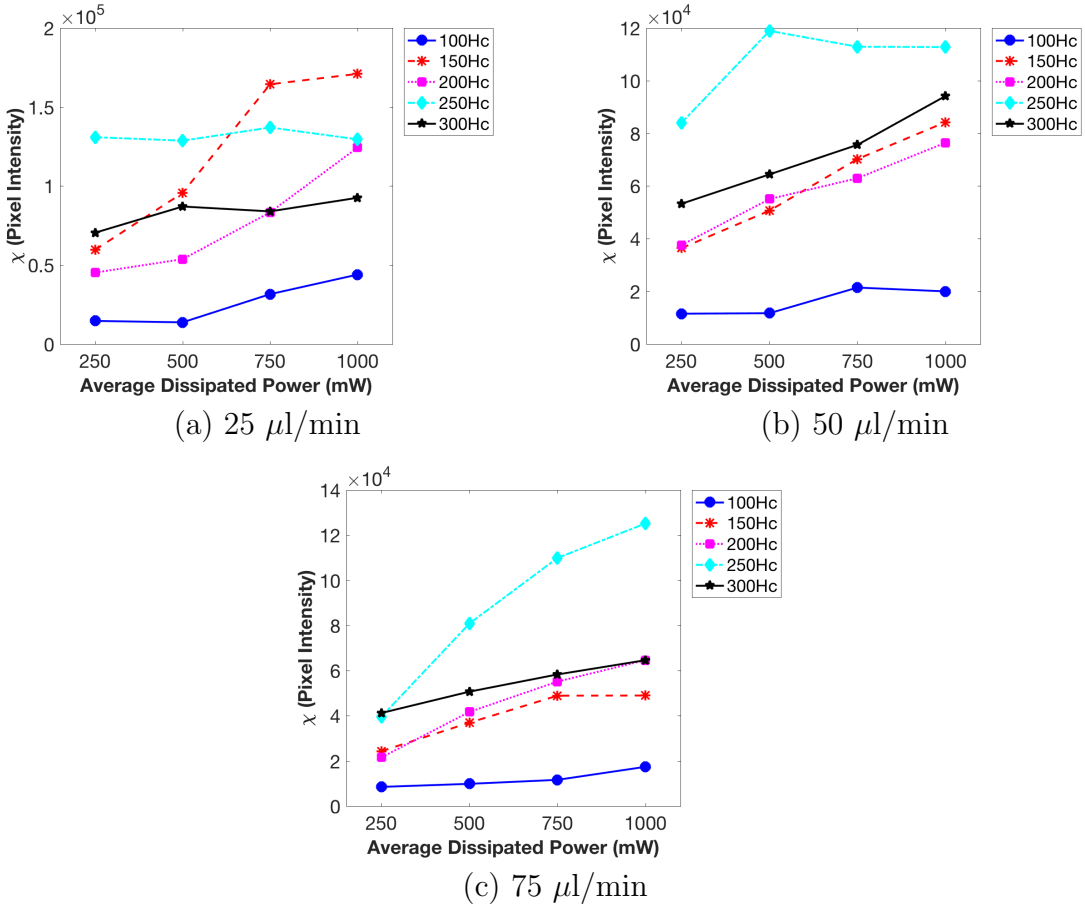
ID	W_c	H_c	W_s	Y_{Pos}	Prominence	f (MHz)
1Wc	500	100	850	High	12143	0.64
2Wc	550	100	850	High	15200	0.51
3Wc	600	100	850	High	4770	1.58
4Wc	650	100	850	High	5470	1.99
5Wc	700	100	850	High	-	-

focusing band across the channel width.

As a significant performance shift was observed as a result of a small adjustment to H_c, channel height was chosen as the variable to isolate for the next design set.

Channel Height Study (Final Screening Test)

This design set fixed values for W_c, W_s, and Y_{Pos} while varying H_c resulting in the geometries shown in Table 5.3. All devices in this set demonstrated focusing of blood; the frequencies for which are shown in Table 5.3. This study was conducted in accordance with the final screening test specified in Section 5.3.2. The results shown in Figure 5·10 demonstrate that, for flow rates higher than 25 μl/min, the chip with a channel height of 250μm performed better than the other chips tested in terms of χ at all studied power levels. Thus the winning geometry of the study's device screening stage has a channel width of 550μm, a channel height of 250μm, a side-wall width of 850μm, with the channel in the High vertical position (i.e., Chip 4Hc in Table 5.3).



(a) 25 μl/min

(b) 50 μl/min

(c) 75 μl/min

Figure 5·10: Performance comparison of channel height study using image analysis. (a-c) plot the performance of the geometries in Table 5.3 for three volumetric flow rates: 25 μl/min, 50 μl/min and 75 μl/min, respectively. Performance is calculated from microscope images in the manner described in Section 5.4.3. Datasets are shown for each channel height from 100 to 300μm.

5.5.4 Comparison of Chip 2.0 versus Baseline Geometry

The results of the screening tests shown in Figure 5·10 yield a geometry that outperformed the other devices in terms of RBC focusing analyzed by image analysis. However, in order to gauge the ultimate success of this geometry it was tested against the baseline geometry. This section presents results from three experiments that com-

Table 5.3: Geometries tested for a final screening study of channel height with corresponding Chip ID. All dimensions are given in units of μm. The listed frequencies coincide with the locations of the corresponding device's fundamental odd resonant mode.

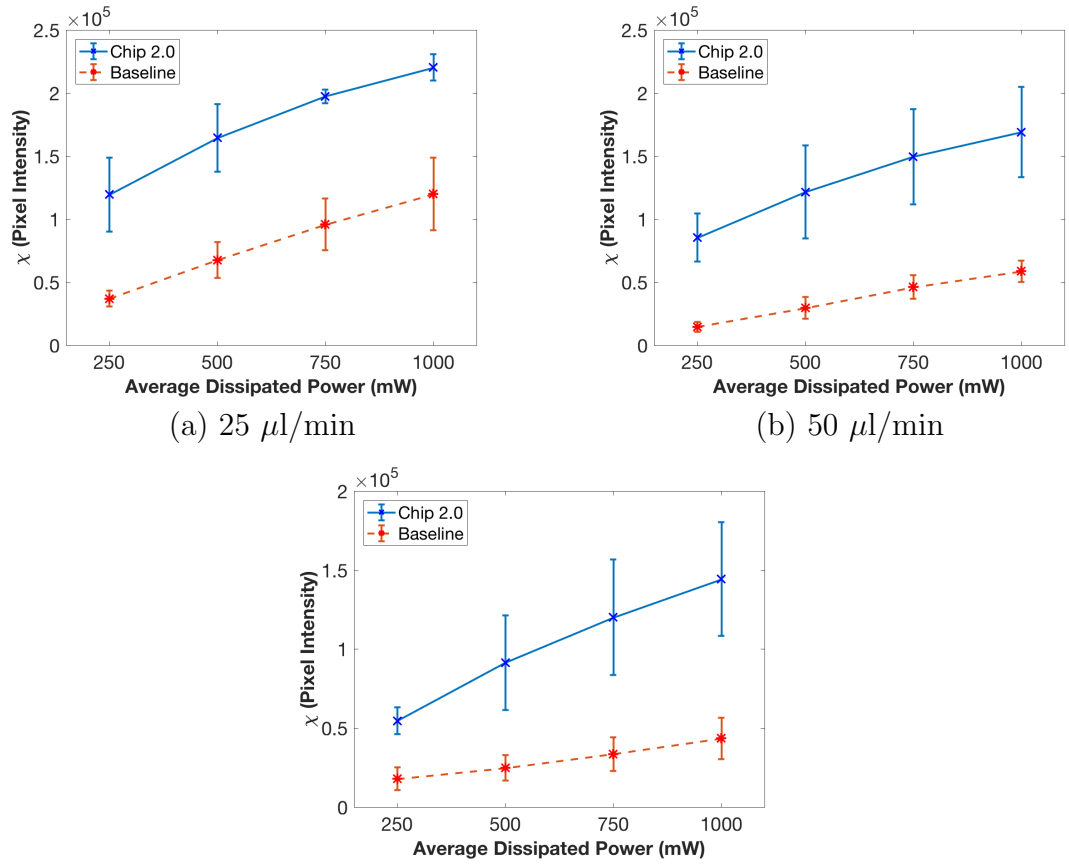
ID	W_c	H_c	W_s	Y_{Pos}	$f\ (MHz)$
1Hc	550	100	850	High	0.684
2Hc	550	150	850	High	0.640
3Hc	550	200	850	High	0.634
4Hc	550	250	850	High	0.632
5Hc	550	300	850	High	0.640

Table 5.4: Baseline versus Chip 2.0. All dimensions are given in units of μm. Standard operating conditions, measured at the transducer, are provided for frequencies, input voltages and currents required to achieve 1 W of average dissipated power. These conditions are the values for which the fundamental resonant odd modes for each chip were achieved.

Chip	W_c	H_c	W_s	Y_{Pos}	$f\ (MHz)$	Voltage (V_{pp})	Current (mA_{pp})
Chip 2.0	550	250	850	High	0.632	56.44 ($\pm$2.4)	1039 ($\pm$53)
Baseline	430	200	1055	High	1.012	45.78 ($\pm$1.86)	1340 ($\pm$70)

pared the winning geometry of the screening tests (i.e., Chip 2.0) against the baseline geometry through image analysis, blood separation, and bacteria separation.

Comparative Focusing



(a) 25 μl/min

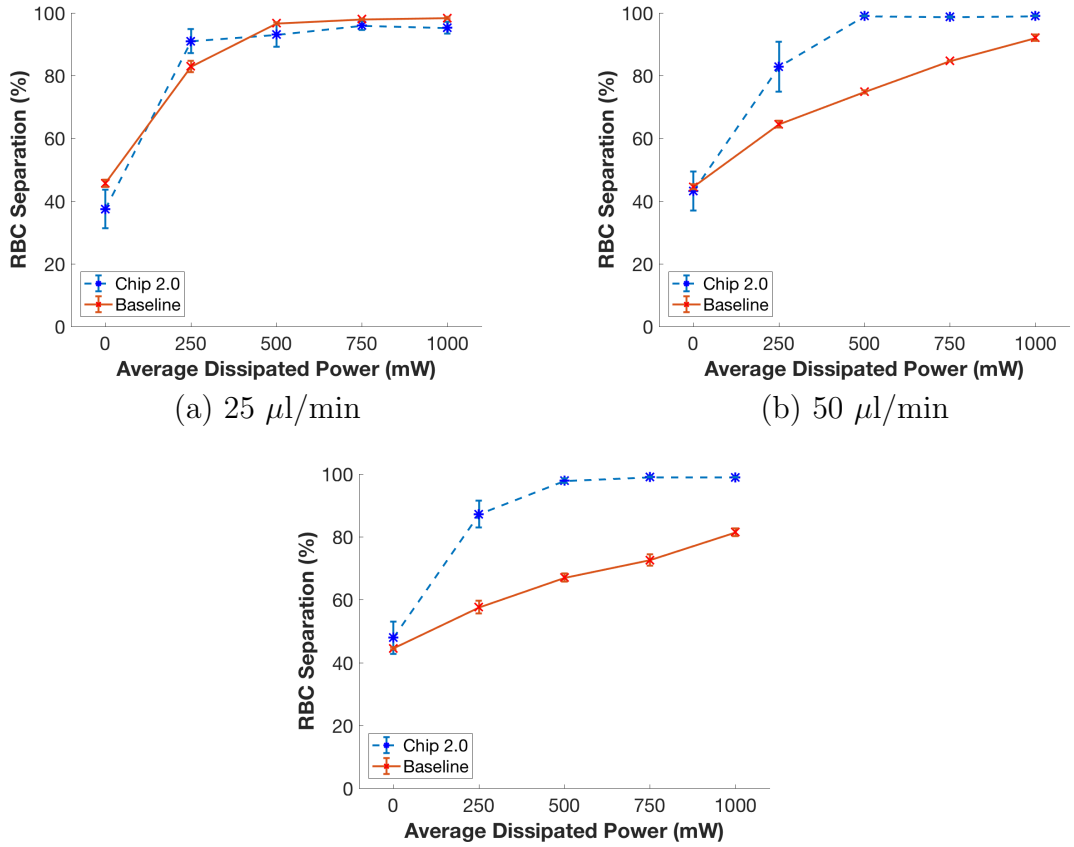
(b) 50 μl/min

(c) 75 μl/min

Figure 5·11: Performance comparison versus baseline using image analysis. (a-c) plot the performance of the baseline versus the winner of the study for three volumetric flow rates: 25 μl/min, 50 μl/min and 75 μl/min, respectively. Performance is calculated from microscope images in the manner described in Section 5.4.3.

Figure 5·11(a–c) plots the performance of Chip 2.0 against the baseline in terms χ. The results show that Chip 2.0 outperforms the baseline for all tested combinations of flow and power settings.

Comparative Blood Separation



(a) 25 μl/min

(b) 50 μl/min

(c) 75 μl/min

Figure 5·12: Blood separation performance comparison versus baseline. (a-c) plot the performance of the baseline versus the winner of the study for three volumetric flow rates: 25 μl/min, 50 μl/min and 75 μl/min, respectively. Performance is defined based on each design's ability to focus RBCs into the middle channel of the trifurcation shown in Figure 5·2. RBC separation is defined as the number of RBCs measured at the center port divided by the total number of RBCs measured at the input port.

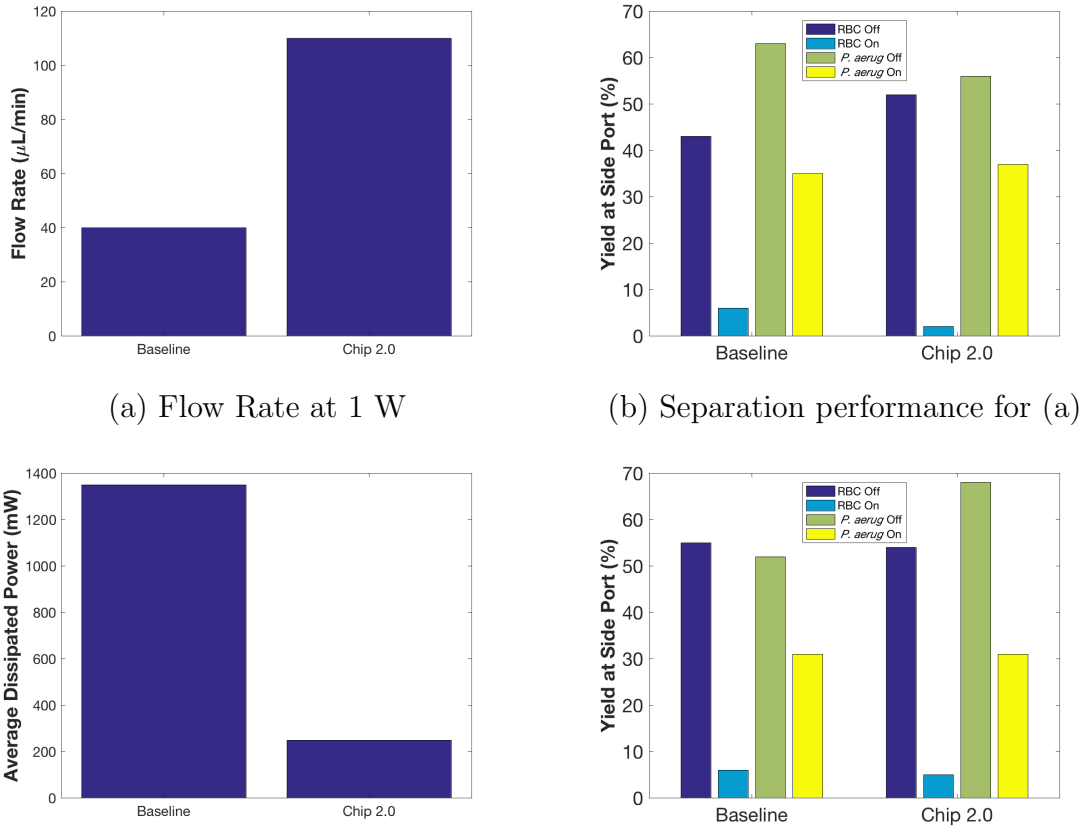
Figure 5·12 plots the performance of Chip 2.0 against the baseline in terms of each device's ability to focus RBCs to the center port. The dynamic range of each measurement was limited between the control measurement (i.e., zero average dissipated

power) and 100% RBC concentration in the center channel. As the chip has two outlet ports (Figure 5·2), RBCs will be approximately equally distributed between them in the acoustics-off (0 W input power) condition. The baseline and Chip 2.0 demonstrated comparable performance at a flow rate of 25 μl/min across all power settings; however, at higher flow rates Chip 2.0 outperformed the baseline across all non-control power settings.

Comparative Bacterial Separation

Four experiments were conducted in three technical replicates, two for each device design (baseline and Chip 2.0), in order to determine the optimal value for each measure of merit while holding the other constant. Optimality was defined as the maximum flow rate or minimum power required to maintain at least 90% RBC separation between the side and center ports while bacteria recovery in the side port was measured.

The results in Figure 5·13 show that Chip 2.0 achieved a 175% increase in throughput relative to the baseline. Additionally, Chip 2.0 was able to decrease the average dissipated power by 81.63% when compared to that of the baseline geometry. The actual average RBC separation across all four experiments was 95.25% ($\pm$1.89%). The yield of the bacterial samples collected at the side port had a standard deviation of 0.05%. For both devices the yield of bacteria in the side port was 33.5% ($\pm$3%) after separation, showing that separation performance in the 2.0 design is equal to that of the baseline while enabling lower driving power and/or higher throughput.



(a) Flow Rate at 1 W

(b) Separation performance for (a)

(c) Power at 50 μl/min

(d) Separation performance for (b)

Figure 5·13: Levels of each measure of merit required to achieve equivalent separation performance. (a) shows the relative performance of the baseline as it compares to Chip 2.0 in terms of flow rate with power held constant at 1 W. (b) illustrates the actual performance of each chip at the levels specified in (a) in both the acoustics "off" (0 W and established chip flow rate) and "on" conditions (1 W and established flow conditions). (c) summarizes the performance of the two chip designs holding flow rate at 50 μl/min while varying power. Constant values were maintained for RBC separation and bacterial recovery for all chip designs and experiments. (d) depicts the performance of each device at the power levels specified in (c); "off" being the control condition of 0 W and 50 μl/min.

It is worth noting that the Chip 2.0 design achieved better RBC separation for both the maximum flow rate and minimum power experiments (98% vs 94% and 95%

vs 94%, respectively) and equivalent bacterial recovery relative to the baseline across all four experiments; thus, for each Chip 2.0 experiment the measure of merit (flow or power) could show greater advantages if the RBC yield were adjusted to match that of the baseline.

5.6 Discussion

The resonant response of acoustophoretic microfluidic devices has been shown to be non-linear (Garofalo et al., 2016)(Bora and Shusteff, 2015)(Glynne-Jones et al., 2009)(Hahn et al., 2014). Complex response patterns with numerous explanatory variables have been successfully optimized using a fractional factorial experimental design approach in conjunction with a response surface methodology (Khuri and Mukhopadhyay, 2010); however, these methodologies assume that the function $f(x)$ modeling the response variable y to the explanatory variables $x = (x_1, x_2, \ldots, x_n)$ accounting for some experimental error ϵ, as shown in Equation 5.5, is a low-degree polynomial (Khuri and Mukhopadhyay, 2010),

$$y = f(x) + \epsilon \tag{5.5}$$

In the absence of a near-quadratic response, multiple rounds of experiments are required to adequately detect a positive gradient in performance (Carley et al., 2004). Even after multiple rounds of experiments, the nature of a complex response surface means that no claim of optimality can be made (Box, 2006). Since this study seeks a performance enhancement, as opposed to a rigorous optimal geometry, this iterative method is acceptable.

This study successfully improved the performance of the acoustic separator chip for both measures of merit. However, Chip 2.0 may not represent an optimal design, and further improvement is likely possible. The experimental design selected parameters

that were most accessible to systematic variation. For example, the impact of total height of the device was not explored because changes in sheet stock thickness would require sourcing or custom fabricating those sheets, and the bond process would have to be adjusted for each thickness. Likewise, the selection of the values tested assumes a more or less smoothly varying response. Further analysis is needed to rigorously support that assumption.

The frequency range studied for the rapid screening tests (0.5 – 2.0 MHz) was chosen based on the known resonance of the baseline device (approximately 1 MHz) and the expectation that small variations in dimensions should result in comparably small variations in resonant frequency. Additionally, the lower limit was selected with the knowledge that acoustic radiation force scales with frequency and lower frequencies could limit the ability of the device to focus small cells or particles such as platelets or bacteria, if desired in other applications. Furthermore, there were practical experimental limitations at the upper end of the range; a transducer with a higher resonant frequency (e.g., 5 MHz) would allow a wider range of frequencies to be explored, however the fragility of such transducers increases the difficulty of mounting and testing the devices. Nevertheless reproducing the screening with a wider frequency range, particularly at the low end could yield further improvements.

Despite these and other limitations, this study was meant not only to identify an improved geometry, but also to establish a method for empirical development of devices. The resulting performance improvement, as measured by the bacteria separation task, suggest that the initial screening using only image-based analysis of RBC focusing provides a useful approach for assessing devices. To perform the setup, operation, output sample collection, and cell counting in the final bacteria separation takes several hours at least with all parameters fixed, whereas hundreds of image-based prominance measurements could be made at a rate of one every 15

seconds.

Future explorations of device designs can be enabled by this parametric, rapid prototyping framework, which is part of a larger workflow (Lashkaripour et al., 2017)(Lippai et al., 2017)(Sanka et al., 2017). The 19 devices designed in this manuscript can take advantage of this process with a high level description of the device functionality, microfluidic primitives, and automated fabrication and control.

5.7 Conclusion

Enabled by a parametric, rapid prototyping framework, we were able to screen 19 device designs, distinct in four geometric parameters, towards optimizing the performance of a blood-bacteria separation device. Compared to the previously published plastic design, we demonstrate that the optimized device geometry can separate blood and bacteria while operating at 175% greater flow rate as well as reducing the power requirement by 81.63% for equivalent separation performance. The improved device will offer increased throughput and reduced power requirements and could aid performance of future point-of-care plastic acoustofluidic devices.

Chapter 6

Conclusions and Future Work

One major goal of this thesis was to lower the barrier of entry into the field of continuous flow microfluidics. This was achieved by developing a rapid, cost-effective, CAD-friendly, design-to-fabrication framework for microfluidics using accessible CAM technologies such as desktop CNC mills and 3D printers. Proving the efficacy of this framework involved demonstrating its capacity to solve relevant and complex experimental problems. The scalability of the framework was proven by its ability to design, fabricate, and control a complex network of novel microfluidic primitives, as shown in Chapter 4. Its experimental relevance was established in Chapter 5 by optimizing an acoustofluidic blood–bacteria separation device in a thermoplastic substrate — something that has never been done before.

Having demonstrated this new rapid prototyping framework is capable of designing, fabricating, and controlling complex and experimentally relevant devices, the next logical step would be to bring this research back to its genesis. This work was motivated by the desire to orchestrate large networks of novel synthetic biological systems. I presented a framework capable of designing a device that separates smaller genetic logic components into disparate reaction chambers and connecting these chambers using microchannels, through which biological signals could be sent. My framework provides a mechanism to describe and execute temporally specified valve control sequences, which can control the flow of biological signaling information such that larger functionality can be achieved.

Additionally, the state of the genetic circuits can be monitored using on-chip diagnostics such as the integration of sensors and optical reporting areas. This on-chip feedback can be integrated into the temporal valve-control specification, thus necessitating an extension to the Biostream language beyond "wait" and "set" statements to include some element of temporal logic such as "wait until".

All of these extensions should be fully integrated into the larger CAD workflow shown in Figure 3·1. This would require forgoing the use of OpenSCAD and replacing it with domain-specific CAD tools such as Fluigi Place and Route (Hu et al., 2014) using MIcrofluidic NeTlist (MINT) (Sanka et al., 2017) descriptions. Smaller devices could be designed using a microfluidic-specific drawing tool such as 3DμF (Lippai et al., 2017). All designs could be informed by a design-automation platform driven by fluid mechanics (Lashkaripour et al., 2017), which would provide aspects of automation to device parameter estimation. The design tools listed are compatible with MakerFluidics in that they can export two-dimensional graphics files (i.e., SVG files), however these tools could be extended to output designs in three dimensions. It must be noted that all of the preceding design tools are made relevant by the ability to fabricate the designs they create; this relevance is boosted by the accessibility MakerFluidics brings to continuous-flow microfluidics.

My results show that low-cost microfluidic fabrication techniques are possible and relevant. The integration of continuous-flow microfluidics into day-to-day life in a typical wet lab has the ability to change the way experimentalists look at the nature of their work. Freed from the mundane and costly burdens of device design, fabrication, and running protocols, experimentalists could spend more time on data analysis and experimental documentation — effectively boosting the quality and quantity of research. MakerFluidics and the capabilities it enables represent an important step towards this integration.

Appendix A

Othermill Standard Operating Procedure

1. **Purpose:** The purpose of this document is to provide a detailed set of instructions to operate the Othermill, for microfluidic device and prototyping.

2. **Scope:** Operating Othermill and required programs in preparation for rapid prototyping of microfluidic devices.

3. **Responsibilities:** Includes any trained personnel attempting to rapid prototype parts using the Othermill and software required for operation.

4. **Reference Documents:**

   (a) Othermill Manufacturer's Instruction Manual

   (b) Othermill Feeds and Speeds Database

5. **Definitions:**

   (a) CAM – Computer-Aided Manufacturing

   (b) CAD – Computer-Aided Design

6. **Equipment and Materials:**

   (a) Othermill Pro

   (b) Computer/Laptop

   (c) Otherplan Software

(d) Fusion 360 CAM Software

(e) Material Stock

(f) End Mills and Drill Bits

7. **Procedure**

(a) Upon completion of CAD model in OpenSCAD, save the CAD model as a .STL file.

(b) Open Autodesk Fusion 360 software and import the .STL file by clicking on the "Insert" drop down menu and selecting "New Design From File".

(c) Once the file is imported, click on the MODEL drop down menu and select the CAM option.

(d) From the SETUP drop down menu, select New Setup.

(e) On the SETUP pop-up window, change the Orientation to Select Z axis/-plane & X axis.

(f) Select the appropriate Z Axis, so that the Z Axis runs perpendicular to the surface being milled, the positive direction of the Z Axis should be pointing from the bottom to the top surface of the part.

(g) The X Axis should be selected to be orientated left to right, when looking down on the top milling surface.

(h) The origin should be selected as the X, Y coordinate (0,0), and the highest selectable point along the Z Axis.

(i) Select the Stock tab within the Setup pop-up and change the Stock Top Offset to 0 mm.

(j) From the 2D drop down menu select the appropriate milling function for the cut, 2D Pocket is for partial depth milling, and 2D Contour is for through cut milling.

(k) For 2D Pocket:

i. Select appropriate tool for the cut being programmed from the 2D Pocket pop-up screen, and make sure to change the Coolant to Disabled.

ii. Put the appropriate tool feeds and speeds from the Othermill Feeds and Speeds Database.

iii. Select the Geometry tab on the 2D Pocket pop-up window, once this screen is active select the appropriate contours to be milled.

iv. Select the Heights tab on the 2D Pocket pop-up window, on the Top Height option, change the From drop down to Model Top, then the Bottom Height option should read Selected contour(s).

v. Select the Passes tab on the 2D Pocket pop-up window, under the Passes option, change the Sideways Compensations to Right (conventional milling). The Maximum Stepover should be changed to 10% the diameter of the tool being used.

vi. Uncheck the Stock to Leave option.

vii. Check the Multiple Depths option, and change the Maximum Roughing Stepdown to the value present in the Othermill Feeds and Speeds Database.

(l) For 2D Contour.

i. Select appropriate tool for the cut being programmed from the 2D Contour pop-up screen, and make sure to change the Coolant to Dis-

abled.

ii. Put the appropriate tool feeds and speeds from the Othermill Feeds and Speeds Databse.

iii. Select the Geometry tab on the 2D Contour pop-up window, once this screen is active, select the appropriate contours to be milled.

iv. Select the Heights tab on the 2D Contour pop-up window, on the Top Height option change the From drop down to Model Top, then the Bottom Height option should be changed to Model Bottom.

v. Select the Passes tab on the 2D Contour pop-up window, under the Passes option, change the Sideways Compensations to Right (conventional milling).

vi. Make sure Stock to Leave is unchecked.

vii. Check the Multiple Depths option, and change the Maximum Roughing Stepdown to the value present in the Othermill Feeds and Speeds Database.

(m) Once the tool paths were completed, a height test tool path should be performed.

i. Using the 2D Contour protocol, select the outline contour of the part being milled.

ii. From the Heights tab on the 2D Contour pop-up, the Top Height and Bottom Height option should be both set to Model Top.

(n) Under the ACTIONS tab, select Simulate and confirm tool paths are viable and that no errors occur.

(o) Once simulations show feasibility, select the Post Process option under ACTIONS.

i. Save each tool path as its own respective .gcode file.

ii. Typical naming structure follows this structure

A. NContour/Pocket/FaceTool, where N is the ordered number of the tool path, Contour/Pocket/Face is the type of milling, and Tool is the type of end mill or drill bit being used.

(p) Connect the Othermill to a computer/laptop containing Otherplan software.

(q) Turn on the Othermill via the power switch located on the back of the system.

(r) Open the Otherplan software.

i. Home the system by selecting Home.

ii. Move the spoilboard to loading position by pressing the Loading button, this will make it easier to place new material on the spoilboard.

A. Secure the material to the spoilboard by placing strips of 3M 1 wide, rubber polypropylene film tape along the left edge, center, and right edge (if required) of the material.

B. Be sure to press down on material to ensure it properly adheres to the double sided tape.

iii. Select the Open Files option located on the right side of the screen.

iv. Select all appropriate .gcode files required to mill the part.

v. Assign proper tools for each milling step.

vi. Set the tool required for the current step by pressing the Change button, and select the proper tool.

A. The system will then be required to establish the proper tool height. After pressing Continue, make sure the tool will not come

in contact with the material by moving the head with the position buttons located on the left side of the Verify tool position pop-up window. Once the tool head is properly positioned, press the Locate Tool button.

vii. Set the correct material size by changing the Width (X), Height (Y), and Thickness (Z) settings in mm.

   A. NOTE: Add 0.2 mm to the Thickness (Z) setting to account for the thickness of the double sided tape.

viii. Run the XXHeightToolXX .gcode file, if the tool does not mill the material adjust the material thickness in the material size settings by decreasing in increments of 200 microns until the XXHeightToolXX .gcode file comes in contact with the material.

ix. Once the material thickness is established, select the Placement option for each of the Contour cuts and change the Z value to 0.20mm instead of 0.00mm to prevent the tool from contacting the double sided tape or spoilboard.

x. Run each .gcode file by pressing the Start Milling button, you will be prompted to change tools if the required milling tool does not match the current tool listed at the top.

References

Abràmoff, M. D., Magalhães, P. J., and Ram, S. J. (2004). Image processing with imagej. *Biophotonics international*, 11(7):36–43.

Adams, J. D., Ebbesen, C. L., Barnkob, R., Yang, A. H., Soh, H. T., and Bruus, H. (2012). High-throughput, temperature-controlled microchannel acoustophoresis device made with rapid prototyping. *Journal of Micromechanics and Microengineering*, 22(7):075017.

Alanis, A. J. (2005). Resistance to antibiotics: are we in the post-antibiotic era? *Archives of medical research*, 36(6):697–705.

Alexander, M. J., Cohoon, J. P., Colflesh, J. L., Karro, J., and Robins, G. (1995). Three-dimensional field-programmable gate arrays. In *Proc. Eighth Annual IEEE Int. ASIC Conf. and Exhibit*, pages 253–256.

Amin, N., Thies, W., and Amarasinghe, S. (2009). Computer-aided design for microfluidic chips based on multilayer soft lithography. In *IEEE International Conference on Computer Design, 2009. ICCD 2009.*, pages 2–9. IEEE.

Anderson, J. R., Chiu, D. T., Wu, H., Schueller, O., and Whitesides, G. M. (2000). Fabrication of microfluidic systems in poly (dimethylsiloxane). *Electrophoresis*, 21(1):27–40.

Antfolk, M. and Laurell, T. (2017). Continuous flow microfluidic separation and processing of rare cells and bioparticles found in blood–a review. *Analytica Chimica Acta*.

Araci, I. E. and Brisk, P. (2014). Recent developments in microfluidic large scale integration. *Current Opinion in Biotechnology*, 25:60–68.

Arge, L., De Berg, M., and Tsirogiannis, C. (2013). Algorithms for computing prominence on grid terrains. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 254–263. ACM.

Barnkob, R. and Bruus, H. (2009). Acoustofluidics: theory and simulation of radiation forces at ultrasound resonances in microfluidic devices. In *Proceedings of Meetings on Acoustics 157ASA*, volume 6, page 020001. ASA.

Barnkob, R., Iranmanesh, I., Wiklund, M., and Bruus, H. (2012). Measuring acoustic energy density in microchannel acoustophoresis using a simple and rapid light-intensity method. *Lab on a Chip*, 12(13):2337–2344.

Beaglehole, R., Irwin, A., and Prentice, T. (2004). The world health report: 2004: changing history.

Bhagat, A. A. S., Bow, H., Hou, H. W., Tan, S. J., Han, J., and Lim, C. T. (2010). Microfluidics for cell separation. *Medical & biological engineering & computing*, 48(10):999–1014.

Bora, M. and Shusteff, M. (2015). Efficient coupling of acoustic modes in microfluidic channel devices. *Lab on a Chip*, 15(15):3192–3202.

Box, G. E. (2006). *Improving almost anything: Ideas and essays*, volu 629. Wiley-Interscience.

Bruus, H. (2012). Acoustofluidics 2: Perturbation theory and ultrasound resonance modes. *Lab on a Chip*, 12(1):20–28.

Carley, K. M., Kamneva, N. Y., and Reminga, J. (2004). Response surface methodology echnical report, DTIC Document.

Chang, Y.-W., Wong, D., and Wong, C. (1996). Universal switch modules for fpga design. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 1(1):80–101.

Cho, M. and Pan, D. Z. (2008 A high-performance droplet routing algorithm for digital microfluidic biochips. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(10):1714–1724.

Compton, K. and Hauck, S. (2002). Reconfigurable computing: a survey of systems and software. *ACM Computing Surveys (csuR)*, 34(2):171–210.

Curtis, C. and Brisk, P. (2015). Simulation of feedback-driven pcr assays on a 2d electrowetting array using a domain-specific high-level biological programming language. *Microelectronic Engineering*, 148:110–116.

Damian, D. and Chisan, J. (2006). An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. *Software Engineering, IEEE Transactions on*, 32(7):433–453.

Duffy, D. C., McDonald, J. C., Schueller, O. J., and Whitesides, G. M. (1998). Rapid prototyping of microfluidic systems in poly (dimethylsiloxane). *Analytical chemistry*, 70(23):4974–4984.

Duncan, P. N., Ahrar, S., and Hui, E. E. (2015). Scaling of pneumatic digital logic circuits. *Lab on a Chip*, 15(5):1360–1365.

Duncan, P. N., Nguyen, T. V., and Hui, E. E. (2013). Pneumatic oscillator circuits for timing and control of integrated microfluidics. *Proceedings of the National Academy of Sciences*, 110(45):18104–18109.

El-Ali, J., Sorger, P. K., and Jensen, K. F. (2006). Cells on chips. *Nature*, 442(7101):403–411.

Elowitz, M. B. and Leibler, S. (2000). A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338.

Fair, R. B. (2007). Digital microfluidics: is a true lab-on-a-chip possible? *Microfluidics and Nanofluidics*, 3(3):245–281.

Farooq, U., Marrakchi, Z., and Mehrez, H. (2012). Fpga architectures: An overview. In *Tree-based Heterogeneous FPGA Architectures*, pages 7–47. Springer.

Fidalgo, L. M. and Maerkl, S. J. (2011). A software-programmable microfluidic device for automated biology. *Lab on a Chip*, 11(9):1612–1619.

Freeman, H. and Davis, L. S. (1977). A corner-finding algorithm for chain-coded curves. *IEEE Transactions on Computers*, 26(3):297–303.

Garofalo, F., Laurell, T., and Bruus, H. (2016). Performance study of acoustophoretic microfluidic silicon-glass devices by characterization of material and geometry dependent frequency spectra. *arXiv preprint arXiv:1610.02794*.

Glynne-Jones, P., Boltryk, R. J., Hill, M., and Harris, N. R. (2009). A [illegible] thin-reflector mode for ultrasonic particle manipulation in layered resonators. In *Ultrasonics Symposium (IUS), 2009 IEEE International*, pages 2137–2140. IEEE.

Grover, W. H., Ivester, R. H., Jensen, E. C., and Mathies, R. A. (2006). Development and multiplexed control of latching pneumatic valves using microfluidic logical structures. *Lab on a Chip*, 6(5):623–631.

Grover, W. H., Skelley, A. M., Liu, C. N., Lagally, E. T., and Mathies, R. A. (2003). Monolithic membrane valves and diaphragm pumps for practical large-scale integration into glass microfluidic devices. *Sensors and Actuators B: Chemical*, 89(3):315–323.

Guckenberger, D. J., de Groot, T. E., Wan, A. M., Beebe, D. J., and Young, E. W. (2015). Micromilling: a method for ultra-rapid prototyping of plastic microfluidic devices. *Lab on a Chip*, 15(11):2364–2378.

Hahn, P., Schwab, O., and Dual, J. (2014). Modeling and optimization of acoustofluidic micro-devices. *Lab on a Chip*, 14(20):3937–3948.

Harris, D. M. and Harris, S. L. (2013). *Digital design and computer architecture*. Morgan Kaufmann Publishers, Cop., Amsterdam, Boston, second edition edit.

Heckele, M. and Schomburg, W. (2003). Review on micro molding of thermoplastic polymers. *Journal of Micromechanics and Microengineering*, 14(3):R1.

Hedayat, A. S., Sloane, N. J. A., and Stufken, J. (2012). *Orthogonal arrays: theory and applications*. Springer Science & Business Media.

Herbsleb, J. D. and Goldenson, D. R. (1996). A systematic survey of cmm experience and results. In *Proceedings of the 18th International Conference on Software Engineering*, ICSE '96, pages 323–330, Washington, DC, USA. IEEE Computer Society.

Hill, M., Townsend, R. J., and Harris, N. R. (2008). Modelling for the robust design of layered resonators for ultrasonic particle manipulation. *Ultrasonics*, 48(6):521–528.

Hu, K., Dinh, T. A., Ho, T.-Y., and Chakrabarty, K. (2014). Control-layer optimization for flow-based mvlsi microfluidic biochips. In *Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2014 International Conference on*, pages 1–10. IEEE.

Huang, H. and Densmore, D. (2014a). Fluigi: Microfluidic device synthesis for synthetic biology. *J. Emerg. Technol. Comput. Syst.*, 11(3):26:1–26:19.

Huang, H. and Densmore, D. (2014b). Integration of microfluidics into the synthetic biology design flow. *Lab on a Chip*.

Huft, J., Haynes, C. A., and Hansen, C. L. (2013). Microfluidic integration of parallel solid-phase liquid chromatography. *Analytical chemistry*, 85(5):2999–3005.

Hung, P. J., Lee, P. J., Sabounchi, P., Lin, R., and Lee, L. P. (2005). Continuous perfusion microfluidic cell culture array for high-throughput cell-based assays. *Biotechnology and bioengineering*, 89(1):1–8.

Jensen, E. C., Stockton, A. M., Chiesl, T. N., Kim, J., Bera, A., and Mathies, R. A. (2013). Digitally programmable microfluidic automaton for multiscale combinatorial mixing and sample processing. *Lab on a Chip*, 13(2):288–296.

Jensen, K. F., Reizman, B. J., and Newman, S. G. (2014). Tools for chemical synthesis in microsystems. *Lab on a Chip*, 14(17):3206–3212.

Khuri, A. I. and Mukhopadhyay, S. (2010). Response surface methodology. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):128–149.

Kim, C.-J. (2001). Micropumping by electrowetting. *ASME PUBLICATIONS-HTD*, 369:55–62.

Kim, J., Stockton, A. M., Jensen, E. C., and Mathies, R. A. (2016). Pneumatically actuated microvalve circuits for programmable automation of chemical and biochemical analysis. *Lab on a Chip*, 16(5):812–819.

Kleeman, L. and Cantoni, A. (1987). Metastable behavior in digital systems. *IEEE Design & Test of Computers*, (6):4–19.

Kuon, I., Tessier, R., and Rose, J. (2008). Fpga architecture: Survey and challenges. *Foundations and Trends in Electronic Design Automation*, 2(2):135–253.

Kwok, R. (2010). Five hard truths for synthetic biology. *Nature*, 463(7279):288–290.

Lashkaripour, A., Sanka, R., Lippai, J., and Densmore, D. (2017). Design automation based on fluid dynamics. In *The Proceedings of the 9th International Workshop on Bio-Design Automation*.

Lauesen, S. and Vinter, O. (2001). Preventing requirement defects: An experiment in process improvement. *Requir. Eng.*, 6(1):37–50.

Laxminarayan, R., Duse, A., Wattal, C., Zaidi, A. K., Wertheim, H. F., Sumpradit, N., Vlieghe, E., Hara, G. L., Gould, I. M., Goossens, H., et al. (2013). Antibiotic resistancethe need for global solutions. *The Lancet infectious diseases*, 13(12):1057–1098.

Lenshof, A., Ahmad-Tajudin, A., Jarås, K., Sward-Nilsson, A.-M., Åberg, L., Marko-Varga, G., Malm, J., Lilja, H., and Laurell, T. (2009). Acoustic whole blood plasmapheresis chip for prostate specific antigen microarray diagnostics. *Analytical chemistry*, 81(15):6030–6037.

Ley, M. W. and Bruus, H. (2016). Continuum modeling of hydrodynamic particle–particle interactions in microfluidic high-concentration suspensions. *Lab on a Chip*, 16(7):1178–1188.

Li, S., Ma, F., Bachman, H., Cameron, C. E., Zeng, X., and Huang, T. J. (2016). Acoustofluidic bacteria separation. *Journal of Micromechanics and Microengineering*, 27(1):015031.

Linshiz, G., Jensen, E., Stawski, N., Bi, C., Elsbree, N., Jiao, H., Kim, J., Mathies, R., Keasling, J. D., and Hillson, N. J. (2016). End-to-end automated microfluidic platform for synthetic biology: from design to functional analysis. *Journal of biological engineering*, 10(1):1.

Lippai, J., Sanka, R., Lashkaripour, A., and Densmore, D. (2017). Function-driven, graphical design tool for microfluidic chips: 3duf. In *The Proceedings of the 9th International Workshop on Bio-Design Automation*.

Madou, M. J. and Kellogg, G. J. (1998). Labcd: a centrifuge-based microfluidic platform for diagnostics. In *BiOS'98 International Biomedical Optics Symposium*, pages 80–93. International Society for Optics and Photonics.

Martinez-Duarte, R. (12 Apr 2012). Easy and inexpensive fabrication of pdms films of different thicknesses. *Lab on a Chip: Chips and Tips*.

McDaniel, J., Baez, A., Crites, B., Tammewar, A., and Brisk, P. (2013). Design and verification tools for continuous fluid flow-based microfluidic devices. In *18th Asia and South Pacific Design Automation Conference (ASP-DAC 2013)*, pages 219–224.

McDaniel, J., Grover, W. H., and Brisk, P. (2017). The case for semi-automated design of microfluidic very large scale integration (mvlsi) chips. In *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1793–1798. IEEE.

McDonald, J. C. and Whitesides, G. M. (2002). Poly (dimethylsiloxane) as a material for fabricating microfluidic devices. *Accounts of chemical research*, 35(7):491–499.

Melin, J. and Quake, S. R. (2007). Microfluidic large-scale integration: the evolution of design rules for biological automation. *Annu. Rev. Biophys. Biomol. Struct.*, 36:213–231.

Minhass, W. H., Pop, P., Madsen, J., and Blaga, F. S. (2012). Architectural synthesis of flow-based microfluidic large-scale integration biochips. In *Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems*, pages 181–190. ACM.

Minhass, W. H., Pop, P., Madsen, J., and Ho, T.-Y. (2013). Control synthesis for the flow-based microfluidic large-scale integration biochips. In *18th Asia and South Pacific Design Automation Conference (ASP-DAC 2013)*, pages 205–212.

Mueller, A., Lever, A., Nguyen, T., Comolli, J., and Fiering, J. (2013). Continuous acoustic separation in a thermoplastic microchannel. *Journal of Micromechanics and Microengineering*, 23(12):125006.

Nge, P. N., Rogers, C. I., and Woolley, A. T. (2013). Advances in microfluidic materials, functions, integration, and applications. *Chemical reviews*, 113(4):2550–2583.

Nguyen, T. V., Duncan, P. N., Ahrar, S., and Hui, E. E. (2012). Semi-autonomous liquid handling via on-chip pneumatic digital logic. *Lab on a Chip*, 12(20):3991–3994.

Nielsen, A. A., Der, B. S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E. A., Ross, D., Densmore, D., and Voigt, C. A. (2016). Genetic circuit design automation. *Science*, 352(6281):aac7341.

Ohlsson, P., Evander, M., Petersson, K., Mellhammar, L., Lehmusvuori, A., Karhunen, U., Soikkeli, M., Seppa, T., Tuunainen, E., Spangar, A., et al. (2016). Integrated acoustic separation, enrichment, and microchip polymerase chain reaction detection of bacteria from blood for rapid sepsis diagnostics. *Analytical chemistry*, 88(19):9403–9411.

Otto, K. N. and Antonsson, E. K. (1991). Trade-off strategies in engineering design. *Research in engineering Design*, 3(2):87–103.

Pamme, N. (2007). Continuous flow separations in microfluidic devices. *Lab on a Chip*, 7(12):1644–1659.

Petersson, F., Åberg, L., Swärd-Nilsson, A.-M., and Laurell, T. (2007). Free flow acoustophoresis: microfluidic-based mode of particle and cell separation. *Analytical chemistry*, 79(14):5117–5123.

Qin, D., Xia, Y., and Whitesides, G. M. (2010). Soft lithography for micro-and nanoscale patterning. *Nature protocols*, 5(3):491–502.

Ro, D.-K., Paradise, E. M., Ouellet, M., Fisher, K. J., Newman, K. L., Ndungu, J. M., Ho, K. A., Eachus, R. A., Ham, T. S., Kirby, J., et al. (2006). Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, 440(7086):940–943.

Sanka, R., Lippai, J., and Densmore, D. (2017). mlsi design with mint. In *The Proceedings of the 9th International Workshop on Bio-Design Automation*.

Schaller, T., Bohn, L., Mayer, J., and Schubert, K. (1999). Microstructure grooves with a width of less than 50 μm cut with ground hard metal micro end mills. *Precision Engineering*, 23(4):229–235.

Schmit, H. (2005). Extra-dimensional island-style fpgas. In *New Algorithms, Architectures and Applications for Reconfigurable Computing*, pages 3–13. Springer.

Schmit, H. and Chandra, V. (2002). Fpga switch block layout and evaluation. In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pages 11–18. ACM.

Schrock, A. R. (2014). Education in disguise: Culture of a hacker and maker space. *InterActions: UCLA Journal of Education and Information Studies*, 10(1).

Selfridge, A. R. (1985). Approximate material properties in isotropic materials. *IEEE transactions on sonics and ultrasonics*, 32(3):381–394.

Settnes, M. and Bruus, H. (2012). Forces acting on a small particle in an acoustical field in a viscous fluid. *Physical Review E*, 85(1):016327.

Shields IV, C. W., Reyes, C. D., and López, G. P. (2015). Microfluidic cell sorting: a review of the advances in the separation of cells from debulking to rare cell isolation. *Lab on a Chip*, 15(5):1230–1249.

Sia, S. K. and Whitesides, G. M. (2003). Microfluidic devices fabricated in poly (dimethylsiloxane) for biological studies. *Electrophoresis*, 24(21):3563–3576.

Silva, R., Sanka, R., and Densmore, D. (2016). Makerfluidics: Microfluidics for the masses. In *The Proceedings of the 8th International Workshop on Bio-Design Automation*, pages 63–64.

Soe, A. K., Fielding, M., and Nahavandi, S. (2013). Lab-on-a-chip turns soft: Computer-aided, software-enabled microfluidics design. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*, pages 968–971. IEEE.

Swain, J., Lai, D., Takayama, S., and Smith, G. (2013). Thinking big by thinking small: application of microfluidic technology to improve art. *Lab on a Chip*, 13(7):1213–1224.

Sweatt, W., Gill, D., Ada, D., Vasile, M., and Claudet, A. (2008). Diamond milling of micro-optics. *IEEE Aerospace and Electronic Systems Magazine*, 23(1):13–17.

Tamsir, A., Tabor, J. J., and Voigt, C. A. (2011). Robust multicellular computing using genetically encoded nor gates and chemical/wires/'. *Nature*, 469(7329):212–215.

Teh, S.-Y., Lin, R., Hung, L.-H., and Lee, A. P. (2008). Droplet microfluidics. *Lab on a Chip*, 8(2):198–220.

Thies, W., Urbanski, J. P., Thorsen, T., and Amarasinghe, S. (2008). Abstraction layers for scalable microfluidic biocomputing. *Natural Computing*, 7(2):255–275.

Thorsen, T., Maerkl, S. J., and Quake, S. R. (2002). Microfluidic large-scale integration. *Science*, 298(5593):580–584.

Todman, T. J., Constantinides, G. A., Wilton, S. J., Mencer, O., Luk, W., and Cheung, P. Y. (2005). Reconfigurable computing: architectures and design methods. In *Computers and Digital Techniques, IEE Proceedings-*, volume 152, pages 193–207. IET.

Unger, M. A., Chou, H.-P., Thorsen, T., Scherer, A., and Quake, S. R. (2000). Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science*, 288(5463):113–116.

Vaidyanathan, P., Der, B. S., Bhatia, S., Roehner, N., Silva, R., Voigt, C. A., and Densmore, D. (2015). A framework for genetic logic synthesis. *Proceedings of the IEEE*, 103(11):2196–2207.

Wang, B. L., Ghaderi, A., Zhou, H., Agresti, J., Weitz, D. A., Fink, G. R., and Stephanopoulos, G. (2014). Microfluidic high-throughput culturing of single cells for selection based on extracellular metabolite production or consumption. *Nature biotechnology*, 32(5):473.

Weibel, D. B., DiLuzio, W. R., and Whitesides, G. M. (2007). Microfabrication meets microbiology. *Nature Reviews Microbiology*, 5(3):209–218.

Whitesides, G. M. (2006). The origins and the future of microfluidics. *Nature*, 442(7101):368–373.

Wikibooks (2017). Openscad user manual — wikibooks, the free textbook project.

Wohlwend, H. and Rosenbaum, S. (1993). Software improvements in an international company. In *Proceedings of the 15th International Conference on Software Engineering*, ICSE '93, pages 212–220, Los Alamitos, CA, USA. IEEE Computer Society Press.

Xia, Y. and Whitesides, G. M. (1998). Soft lithography. *Annual review of materials science*, 28(1):153–184.

Yen, D. P., Ando, Y., and Shen, K. (2016). A cost-effective micromilling platform for rapid prototyping of microdevices. *Technology*, 4(04):234–239.

Yokoyama, Y. et al. (1993). *Taguchi methods: design of experiments*, volume 4. Amer Supplier Inst.

Zhang, C., Xu, J., Ma, W., and Zheng, W. (2006). Pcr microfluidic devices for dna amplification. *Biotechnology advances*, 24(3):243–284.

Zhao, Y. and Chakrabarty, K. (2012). Cross-contamination avoidance for droplet routing in digital microfluidic biochips. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(6):817–830.

Ryan Silva

610 Commonwealth Ave.
Boston University
Boston, MA 02215
rjsilva@bu.edu

Education

Boston University, Boston, MA 02215
PhD, Electrical and Computer Engineering GPA: 4.0/4.0 2014 - 2017
Advisor: Douglas Densmore
Thesis: MakerFluidics: Low Cost Microfluidics for Synthetic Biology

Air Force Institute of Technology (AFIT), Dayton, OH, 45433
Master of Engineering, Electrical Engineering GPA: 3.8/4.0 2005 - 2007
Advisor: Rusty Baldwin
Thesis: Optimizing the Advanced Encryption Standard for an 8-bit Datapath

United States Air Force Academy (USAFA), Colorado Springs, CO, 80841
Bachelor of Science, Electrical Engineering GPA: 3.7/4.0 2002 - 2005

Awards

Charles Stark Draper Laboratory Fellow	2016 - 2017
USAFA Faculty Pipeline Fellow	2014 - 2017
Great Minds in STEM Most Promising Engineer	2013
IEEE Design Excellence Award	2013
Outstanding Academy Educator	2013

Professional Affiliations

Association for Computing Machinery VLSI (ACMVLSI)	Reviewer 2017
American Society of Engineering Education (ASEE)	Member 2011
Tau Beta Pi - National Engineering Honor Society	Inducted 2006
Eta Kappa Nu - National Electrical Engineering Honor Society	Inducted 2006
Institute of Electrical and Electronics Engineers (IEEE)	Member 2004
United States Air Force Academy Association of Graduates (USAFA AOG)	Member 2001

Invited Talks

- "*MakerFluidics: Microfluidics for the Masses*"
  2nd Workshop on Molecular Communications
  Dublin, Ireland 9 May 2017

Publications

1. R. Dubay, **R. Silva** and J. Fiering, "High Throughput Acoustophoresis in an Array of Parallel Plastic Microchannels" Acoustofluidics 2017.

2. **R. Silva**, P. Dow, R. Dubay, C. Lissandrello, J. Holder, D. Densmore and J. Fiering, "Rapid prototyping and parametric optimization of plastic acoustofluidic devices for blood–bacteria separation" Biomedical Microdevices 2017.

3. **R. Silva**, S. Bhatia, and D. Densmore, "A reconfigurable continuous-flow fluidic routing fabric using a modular, scalable primitive," Lab on a Chip 2016.

4. **R. Silva**, R. Sanka and D. Densmore, "MakerFluidics: Microfluidics for the Masses" Eighth International Workshop on Bio-Design Automation (IWBDA) 2016.

5. **R. Silva**, A. Heuckroth, H. Huang, A. Rolfe and D. Densmore, "MakerFluidics: Microfluidics for All", SynBERC 2015.

6. P. Vaidyanathan, B. Der, S. Bhatia, N. Roehner, **R. Silva**, C. Voigt and D. Densmore, "A Framework for Genetic Logic Synthesis." Proceedings of the IEEE 2015.

7. H. Huang, A. Heuckroth, **R. Silva**, S. Bhatia, and D. Densmore, "Fluigi: An Automated Framework for Creating Bioelectronic Devices", Seventh International Workshop on Bio-Design Automation (IWBDA), 2015.

8. A. Pacheco, **R. Silva**, S. Iverson, T. Haddock and D. Densmore, "Multicellular Logic through Conversion of Genetic Circuit Outputs into Electrical Signals", SynBERC 2015.

9. A. Heuckroth, H. Huang, **R. Silva**, S. Iverson, T. Haddock, A. Pacheco, and D. Densmore, "Accessible microfluidic mold fabrication using 3d printing", SynBERC 2015.

10. C. Hendrix, D. Neebel and **R. Silva**, "A Breadth First Course in Electrical and Computer Engineering." ASEE 2014.

11. M. Roberts, R. Deppensmith and **R. Silva**, "Observations from First-Year Instructors: What We Wish We Knew Before We Began." ASEE 2012.

12. **R. Silva**, "Implementation and Optimization of the Advanced Encryption Standard Algorithm using an 8-bit Datapath", Defense Technical Information Center 2007.

Teaching Experience

United States Air Force Academy, Assistant Professor of ECE

ECE382 Embedded Systems I, Course Director	AY2012-2014
ECE281 Digital Design and Computer Architecture, Course Director	AY2012-2014
ECE210 Principles of Air Force Electronic Systems(Maj), Course Director	AY2013-2014
ECE315 Principles of Air Force Electronic Systems(Core), Instructor	AY2011-2013
ENGR101 Introduction to Air Force Engineering, Instructor	AY2012-2014
ECE463/464 Capstone Design Project, Mentor	AY2012-2014

Patents

- J. Fiering, **R. Silva** and P. Dow, "*Acoustophoresis Device Having Improved Dimensions*" U.S. Provisional Pat. Ser. No. 6038.US.00, filed 14 June 2017.