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

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 knows that it's not easy to juggle a pregnant wife and troubles in the lab, but somehow we still managed to fit in eight hours of TV a day. 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 always asking 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 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
FWHM	.....	Full-Width at Half Maximum
mLSI	.....	Microfluidic Large Scale Integration
PCB	.....	Printed Circuit Board
PDMS	.....	Polydimethylsiloxane
$\mathbb{R}^2$	.....	the Real plane
VLSI	.....	Very Large Scale Integration

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 seem to 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 was motivated by 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. 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. In light of these benefits, one is left to wonder why microfluidic devices 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.).

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 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 motivation 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 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. These results are fully presented in Chapter 5.

Chapter 2

Background

2.1 Microfluidic Design-to-Device Workflows

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*, such as VLSI, 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), and BioStream (Thies et al., 2008), which could serve as the cornerstone of true experimental automation and will be described in the subsequent section. It could serve as a useful exercise to review the methodology computer engineers use to manage complexity and then contrast these principles with the current state of microfluidic design. There are few better places to find fundamental digital design practices than in introductory textbooks, one of which lists abstraction as the most important element of managing complexity (Harris and Harris, 2013).

2.1.1 Abstraction in Fabrication

It is difficult to imagine a computer engineer having to create a VLSI layout and fabricate an 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, the

basic tenants of digital design require 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.

If the goal is true experimental automation, then the expensive and time-consuming fabrication step must 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 computer engineers 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 CAD tools to layout application specific integrated circuits (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.

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.

2.1.2 Microfluidic Design Tools

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 implies placing the scientist at only the highest levels of abstraction and masking all underlying details. Currently, even the best efforts in microfluidic tools only remove intermediate levels of abstraction, while exposing the scientist to the highest **and lowest** levels. Imagine if the only output of a C program were a circuit schematic that must first be built in order to obtain the result of the program. The use of the term "working levels" of complexity implies that the person operating within that 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 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 paper 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 for chip fabrication. Microfluidic chip fabrication is a highly unproductive in that it requires many hours to design and build a device incapable of performing diverse and repeatable experiments. Fortunately for the computer engineer there exists other prototyping options besides PCBs and ASICs, such as the use of a field programmable gate array (FPGA) or microcontroller. The microfluidic experimentalist is left with only one prototyping option that almost always requires some level of device fabrication.

2.1.3 Traditional Microfluidic Fabrication Using Soft Lithography

EDIT THIS: IT'S FROM CASSIE'S THESIS Microfluidic chips are fabricated from PDMS through soft lithography. As many detailed reviews of the process exist (Duffy et al., 1998)(Sia and Whitesides, 2003)(Weibel et al., 2007), I will only provide a brief description. Photoresist (typically SU-8) is spun out over a substrate of silicon, and a transparency with the chip design is placed over it as a mask. The sandwich of mask, photoresist, and substrate is then exposed to UV light. The mask is removed, and the photoresist washed in developing agent to obtain the master mold. PDMS layers are cast from the master through replica molding. The channels are then sealed against a substrate suitable for imaging and connected to input and control structures. A summary of the process is shown in Figure 22A. The entire fabrication process, from the creation of the photomask to the molding of the chip, takes no more than a few days including the turnaround time for printing the photomask. This process does require the experimenter to have access to a high quality cleanroom and purchase specialized fabrication equipment for soft lithography (Elveflow, 2015), or

else contract out the fabrication process to a dedicated microfluidic foundry. **STOP**

2.2 Continuous-Flow Microfluidic Routing

It is difficult to imagine a computer engineer having to create a VLSI layout and fabricate an 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, the basic tenants of digital design require 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. If the goal is true experimental automation, then the expensive and time-consuming fabrication step must 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 analogy seems strange as computer engineers can be productive without having to know how to wash chemical from printed circuit boards (PCBs), use CAD tools to layout application specific integrated circuits (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. Why is that? This paper will provide a possible answer via a historical analysis of the emergence of digital electronics design and highlight instances where microfluidics deviated, resulting in the current design topology seen today. It will also analyze various attempts to rectify microfluidic design challenges using various computer-aided tools and possibly provide the missing element that would allow the microfluidic experimentalist to be one step closer to realizing the productivity potential of true large-scale integration.

2.3 Acoustofluidic Cell Separation

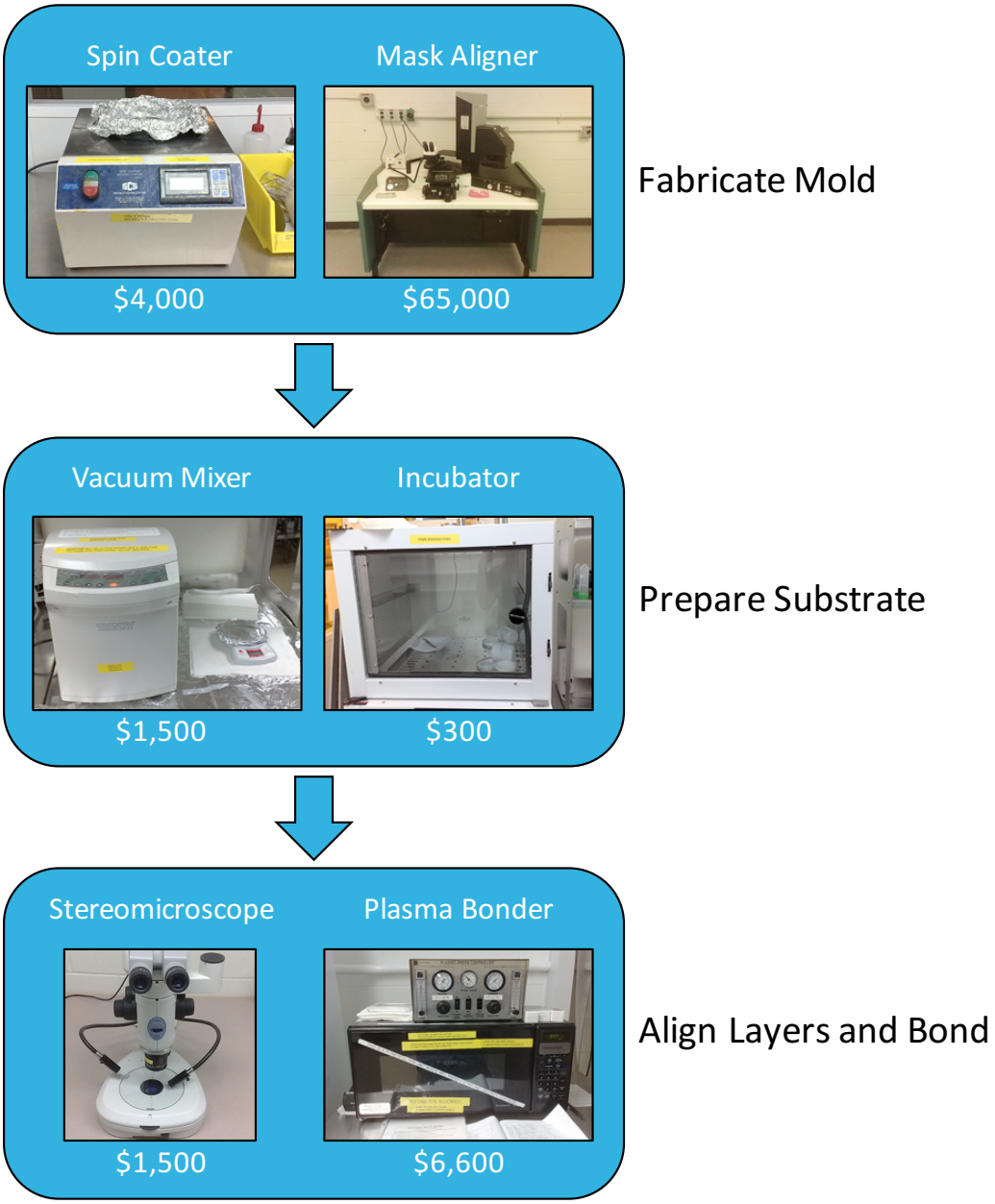


Figure 2·1: Microfluidic fabrication via multi-layer soft lithography requires about $80,000 in specialized equipment, not accounting for infrastructure, personnel, maintenance, or training. Cost estimates based on company quotes.

Chapter 3

The MakerFluidics Framework

3.1 Introduction

MakerFluidics is a microfluidic fabrication and control paradigm that operates within a set of constraints guided by the ideals of automation and the modern "maker movement". This paradigm 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 layout as inputs and generates all of 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.

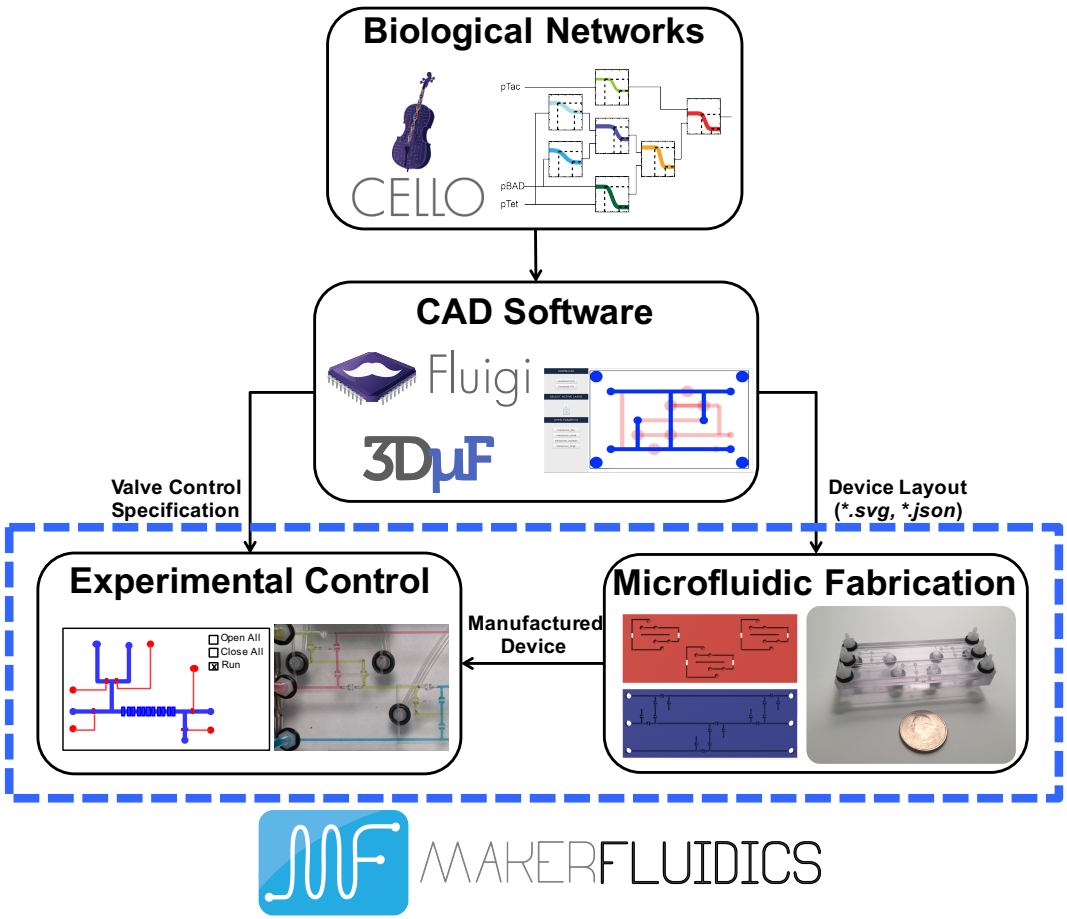


Figure 3·1: MakerFluidics is part of a larger microfluidic design flow that spans from biological specification to microfluidic fabrication and control

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 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 mill (Othermill, Othermachine Co.) and 3D printer (Ultimaker 2, Ultimaker B.V.) used in our lab are each less than $2,500. 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.

3.4 Microfluidic Design

3.5 Microfluidic Fabrication

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

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 and wet etching, which require 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 3·2) 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).

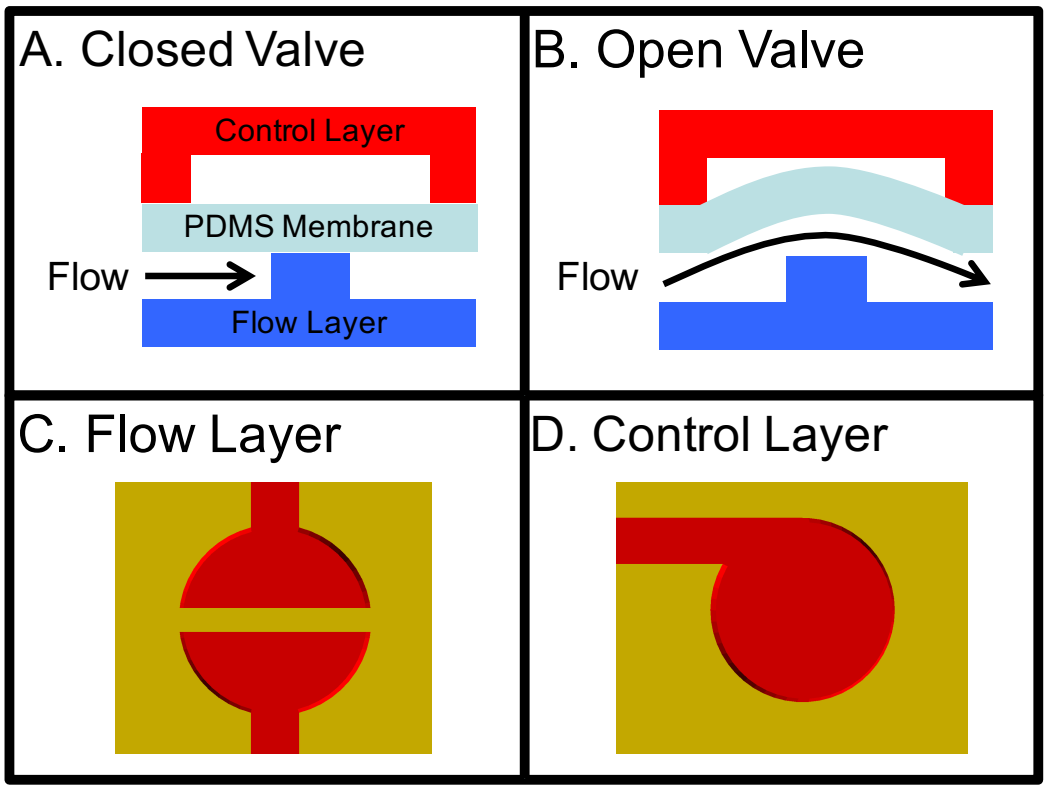


Figure 3·2: Normally-closed monolithic membrane valves (Grover et al., 2003) are realized by introducing discontinuities in the flow layer (blue) and a corresponding pneumatic cavity in the control layer (red). These two layers are separated by a PDMS membrane. To open the valve a vacuum is introduced into the cavity in the control layer.

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 5psi (McDonald and Whitesides, 2002). Using these techniques we have fabricated a number of continuous flow devices inluding the device shown in Figure 3·1 as well as centrifugal microfluidic systems.

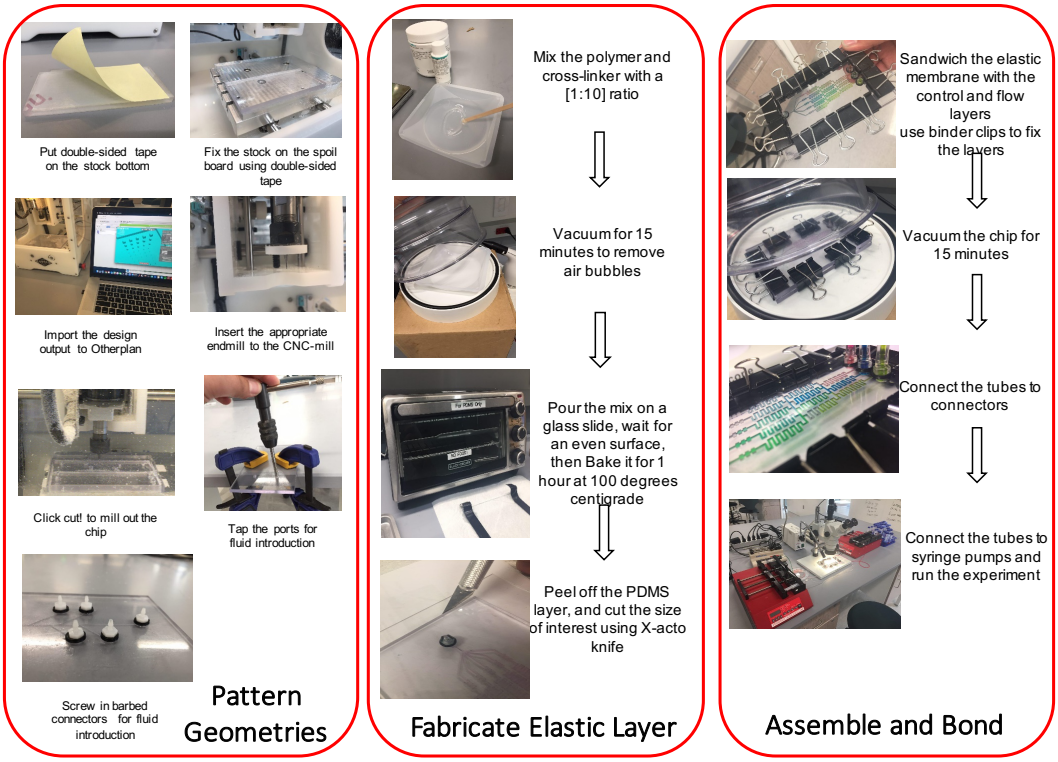


Figure 3·3: Workflow for fabricating microfluidics using the MakerFluidics framework.

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 is automatically generated by microfluidic CAD software developed in CIDAR Lab, but it can also be created manually as a series of wait statements and valve conditions shown in Figure 3·4. These valve objects are linked to the microfluidic layout provided to the fabrication software through the use of a JSON object for display in 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.

A.

```
begin;
    setValve(0, CLOSED);
    setValve(1, OPEN);
    setValve(2, OPEN);
    setValve(3, OPEN);
    setValve(4, OPEN);
    setValve(5, CLOSED);
wait(2000);
    setValve(0, OPEN);
    setValve(1, CLOSED);
    setValve(2, CLOSED);
    setValve(3, CLOSED);
    setValve(4, CLOSED);
    setValve(5, OPEN);
end;
```

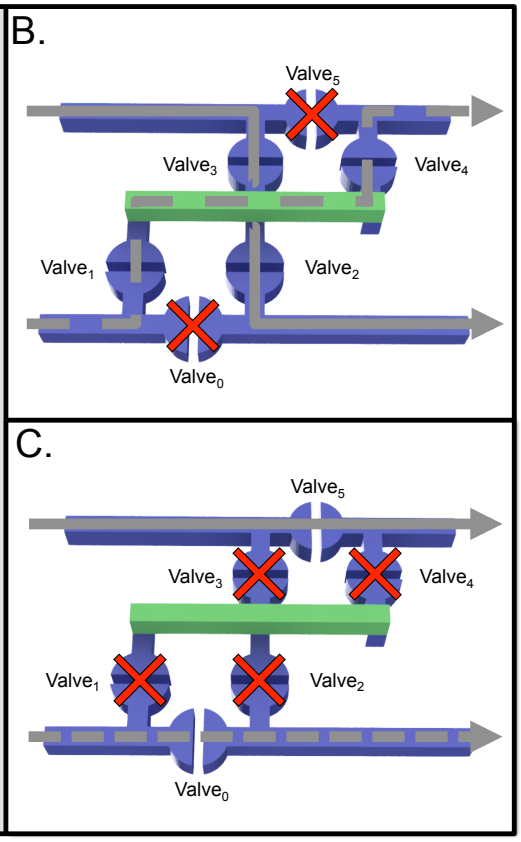


Figure 3·4: The temporal specification (A) (Thies et al., 2008) for a device containing six valves (B, C). The specification in (A) dictates the set of conditions in (B) to begin the assay. After 2000ms the valves change state to that shown in (C), thus affecting the movement of fluid through the device.

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, 2014), 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, 2014) 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.

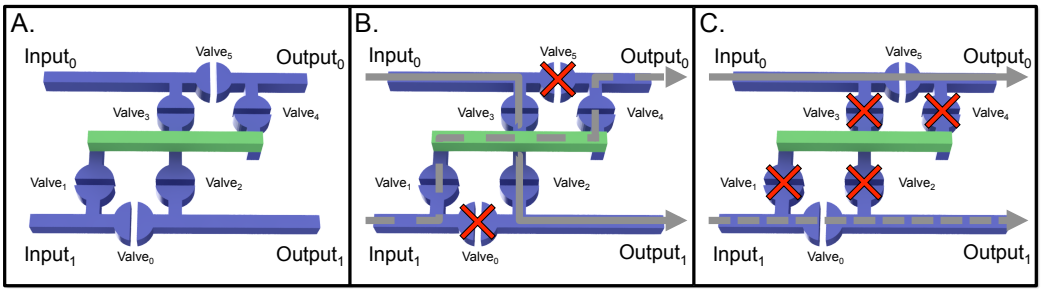


Figure 4·1: Theory of operation for a single transposer primitive. A. Normally closed elastomeric membrane valves are realized through discontinuities in flow channels and a corresponding pneumatic cavity on the control layer. The via (green channel) is located on the control layer and is accessed through 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

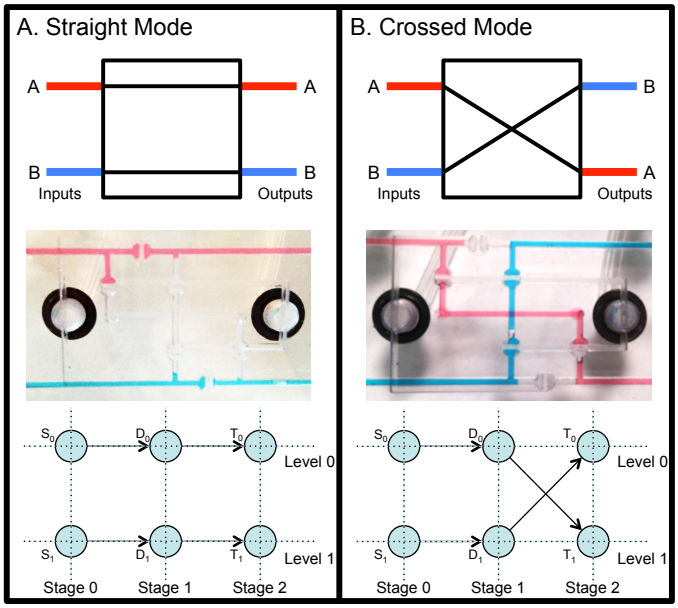


Figure 4·2: The functional, physical and graphical representations above demonstrate how the transposer can route fluids straight through the primitive (A.) or swap the inputs when viewed from the output ports (B.). Vertices labeled v_i where $v \in \{S, D, T\}$ correspond to source, decision and terminal 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,

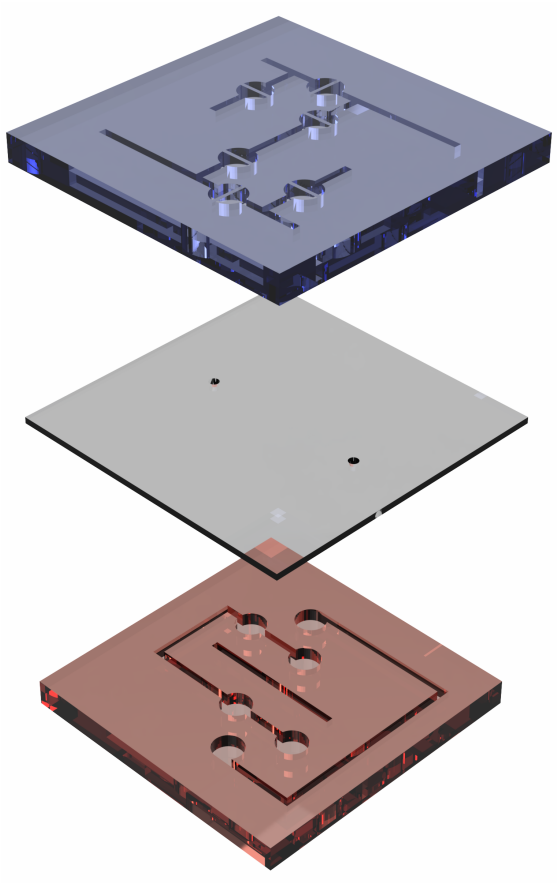


Figure 4·3: Microfluidic assembly stack. Channel and valve geometries in flow (blue) and control (red) layers were ablated from polycarbonate. The via on the control layer is accessed from the flow layer by through-holes punched in the PDMS membrane that separates the two polycarbonate layers.

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

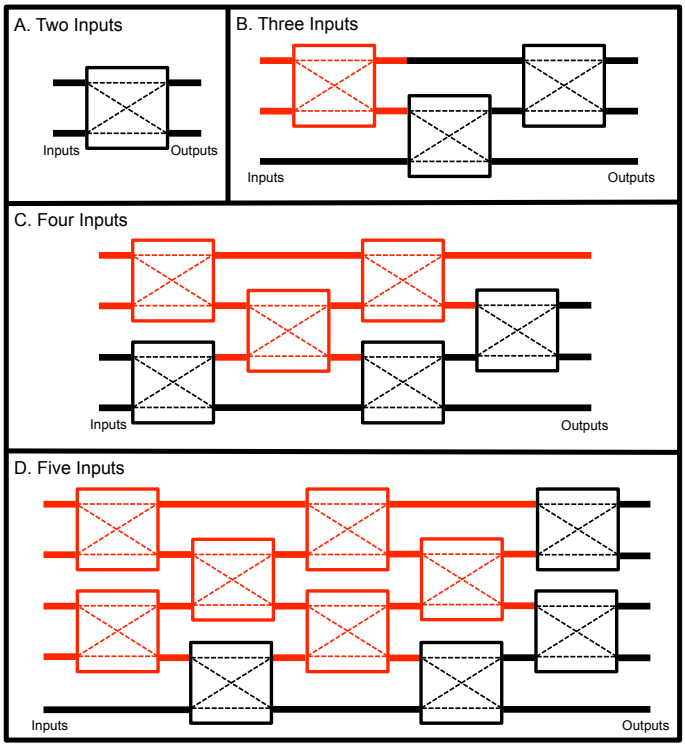


Figure 4·4: Examples of a populated routing fabric. A. shows a single primitive, represented as a box, which is able to route two fluidic inputs to two fluidic outputs. This primitive (highlighted in red in B.) can be tiled with two other primitives, thus enabling the ability to route three fluidic inputs. 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

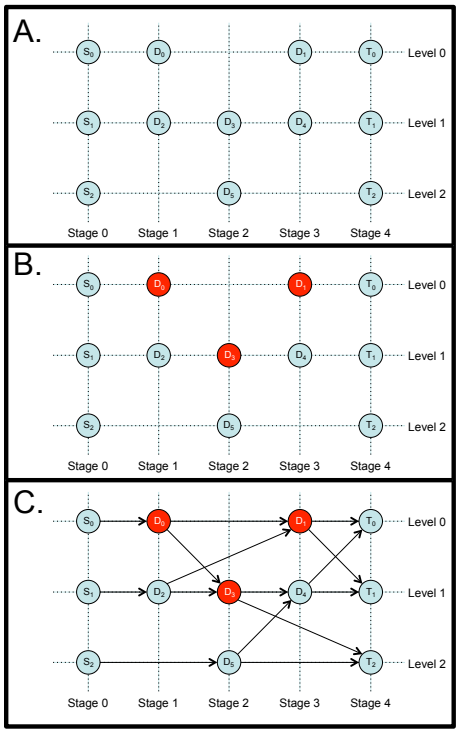


Figure 4·5: Algorithmic progression for generating an unrouted graph for a three-input ($n = 3$) routing fabric. A. shows the outcome of Algorithm 1, where vertices v_i have coordinates p and types $v \in \{S, D, T\}$ corresponding to source, decision and terminal nodes, respectively. B. shows in red 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

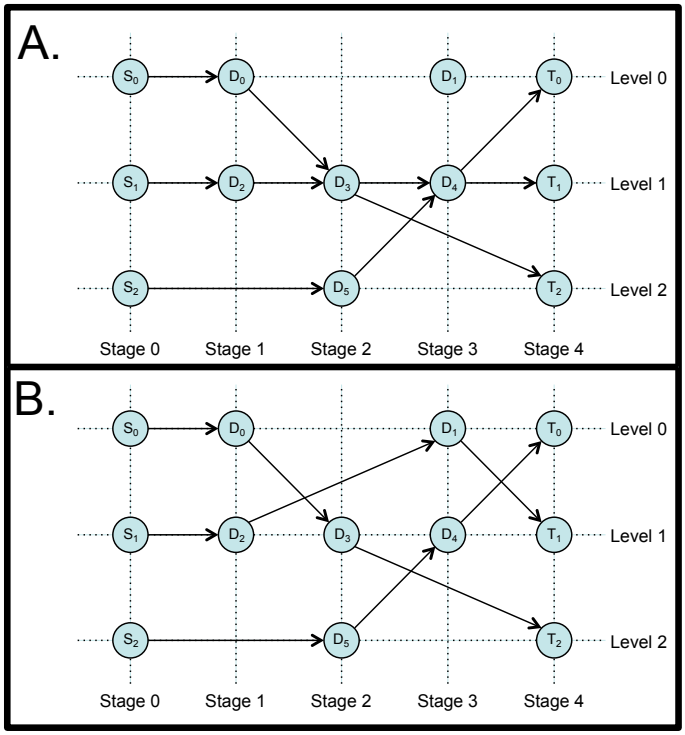


Figure 4·6: Graphical representations of illegally routed (A.) and correctly routed (B.) graphs. Each graph routes inputs from S_0 to T_2, S_1 to T_1 and S_2 to T_0. The graph in A. is illegal because vertices D_3 and D_4 appear in more than one path.

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, 2014). 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.

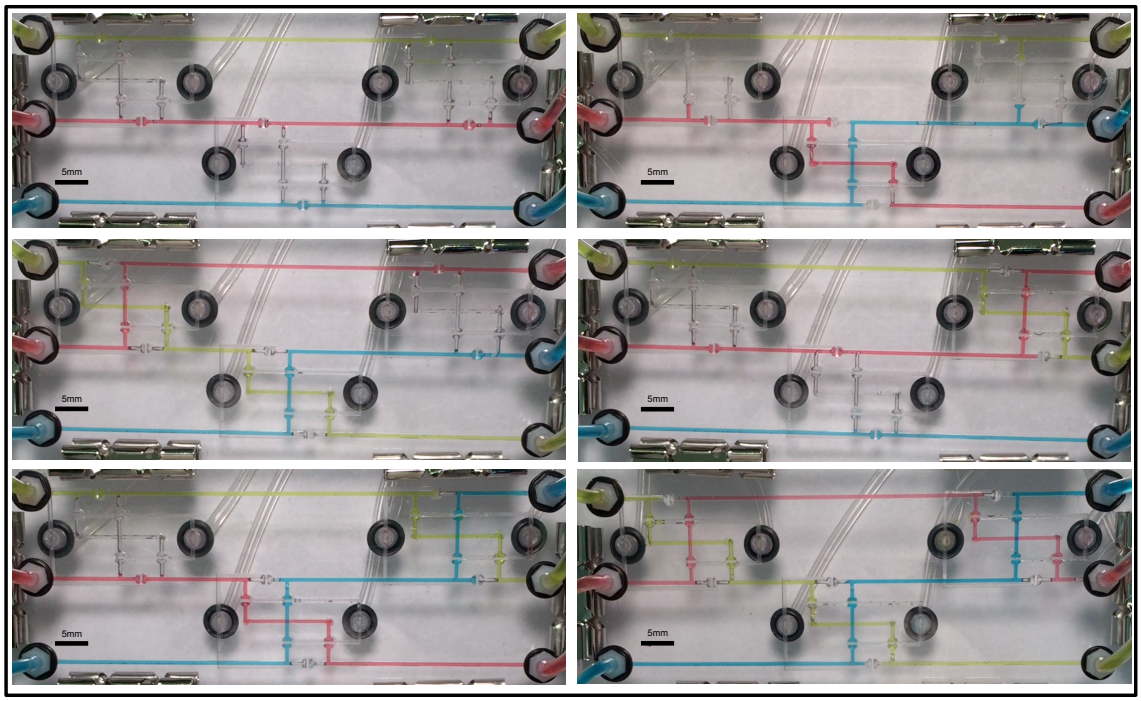


Figure 4·7: All possible permutations for a three-input routing fabric. Fluid moves from left to right in each photo. The inputs from top to bottom are always in the order: green, red, blue. The outputs of the photos correspond to all possible routing permutations for the three input fluids to the three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

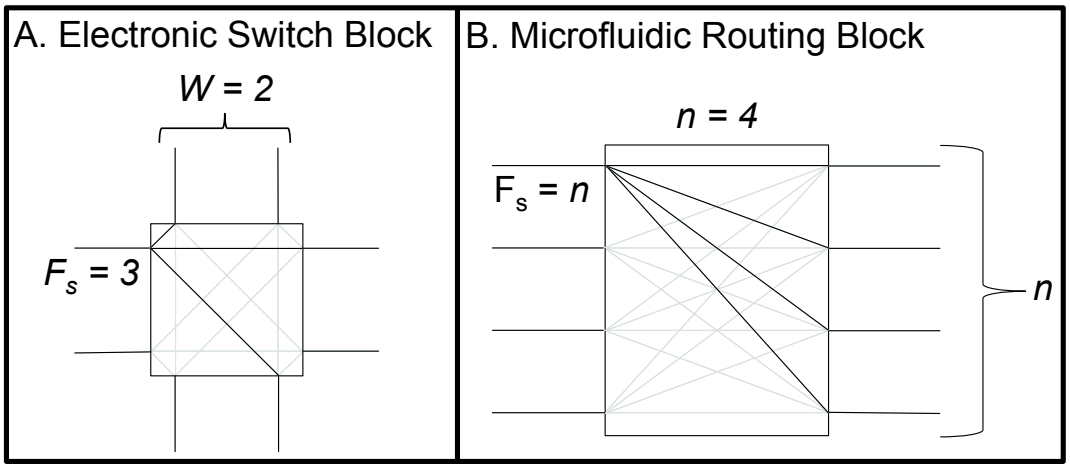


Figure 4·8: Electronic (A.) and Microfluidic (B.) programmable routing blocks. The channel width (W) and flexibility (F_s) of the electronic switching block (A.) is two and three, respectively. These switching block attributes are the same as the ones used in Figure 4·9A and 4·9B.

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.

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 reaction (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.

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 con-

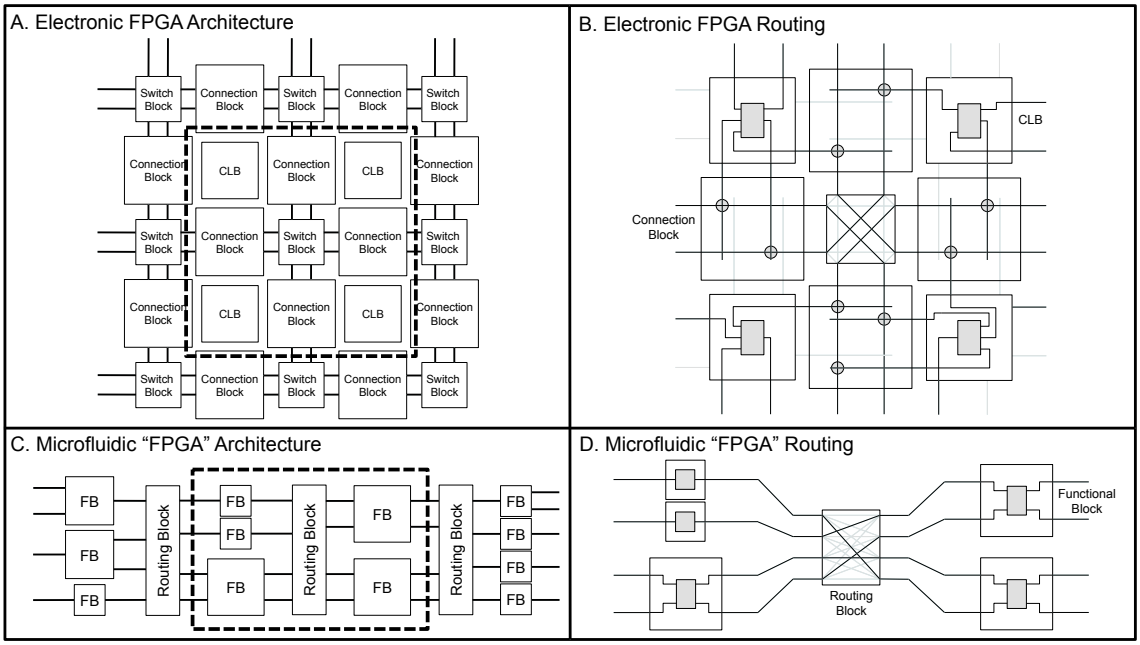


Figure 4·9: Electronic and Microfluidic programmable architectures. A 2D, island-style Electronic FPGA architecture (A.) with example routing (B.) of the area of interest indicated by a dashed line in A. A transposer-based microfluidic chip architecture (C.) with example routing of area of interest (D.) 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.

tamination 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

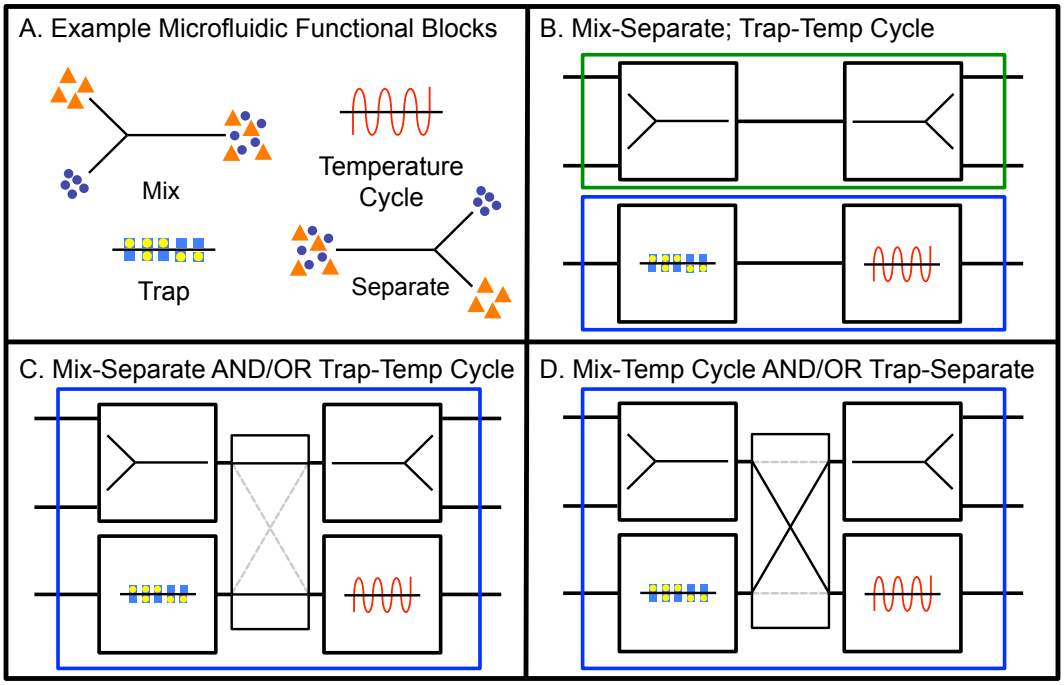


Figure 4·10: Example architecture utilizing a single transposer. Microfluidic functional blocks (A.) are connected via static channels to form two separate devices (B.) the functions of which are immutable. One device can perform mixing and then separation, while the other can perform cell trapping and 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.

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 of parameterized acoustofluidic microchannels towards isolation of bacteria for point of care diagnostics

5.1 Introduction

Acoustic manipulation has emerged as a versatile method for microfluidic separation and concentration of particles and cells. It has advantages over other methods because of high throughput, no affinity chemistry, and no electrodes or specialized structures in the microchannel. The efficiency of the separation depends only on microchannel dimensions appropriately selected for the ultrasonic excitation frequency. However, the potential of plastic acoustofluidic devices is hampered by complex boundary conditions relative to those of more rigid materials, ultimately resulting in the lack of a predictive model. In the absence of a mechanism for predicting focusing conditions, improvements in device performance must either be delayed indefinitely until a model is built or one must experimentally explore a seemingly unbounded solution space dictated by non-continuous, non-linear performance patterns. In the absence of a rapid prototyping framework, the latter choice can involve prohibitive amounts of time and fabrication costs and thus is not a practical option. This paper presents such a framework and uses it to explore the complex, multi-dimensional response

surface ultimately arriving at a device geometry better able to separate bacteria from clinical blood samples to downstream biochemical reactions that result in an optical read out. This detector uses a proprietary synthetic biology approach to optically identify target pathogens and rapidly test their susceptibility to antibiotics.

5.2 Problem Statment

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 (Laurell et al., 2007). This ideal boundary simplifies design because the resultant reduction in mathematical complexity allows for the development of models that can be used to calculate the resonant modes in the channel-fluid system (Petersson et al., 2007). However, the rigid materials previously used are expensive and slow to manufacture, have poor compatibility with many biological samples, and are unlikely to be acceptable for mass production of disposable laboratory tools (Nge et al., 2013). Recent work has demonstrated acceptable levels of acoustic separation in plastic, showing that acoustic separation is possible in polystyrene, opening the door to low-cost diagnostic and therapeutic devices (Mueller et al., 2013).

To further optimize acoustic microfluidics in plastic in the absence of a predictive model, an experimental investigation is needed. While acoustic separation in plastic is proven to work experimentally, enhancing performance is challenging because the simplified analysis developed by others no longer applies, since the channel walls can no longer be considered ideally rigid (Barnkob et al., 2010). Additionally, while plastic acoustofluidic devices can be scaled using methods such as hot embossing or injection molding (Koerner et al., 2005), prototyping geometries in small batches remains an expensive proposition.

This study seeks to find a better-performing device geometry when compared to the device currently in use, which will be referred to as the *baseline*. Fabrication costs will be minimized 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. Devices were screened in rapid succession using a performance parameter described in Section 5.7 while varying two measures of merit, dissipated power and volumetric flow rate, towards enhancing the separation performance of point-of-care plastic acoustofluidic devices.

5.3 Methods

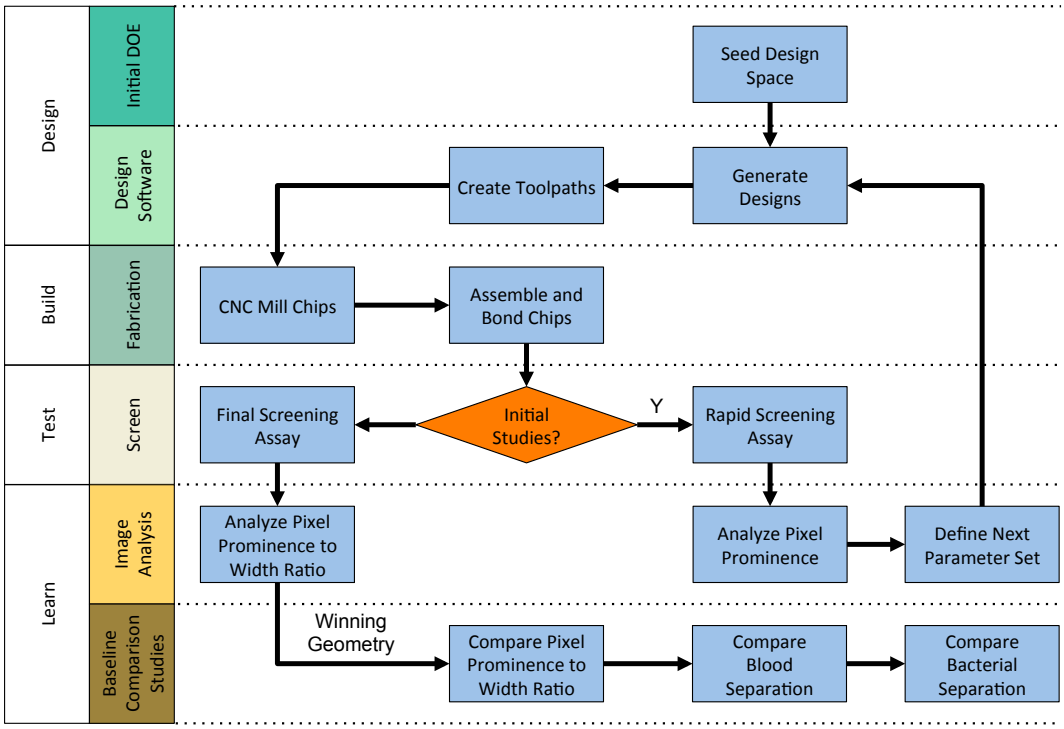


Figure 5·1: Rapid Prototyping Workflow.

Figure 5·1 illustrates the iterative workflow used in conducting the study. The basic theory of operation for the acoustofluidic separation device is explained in Sec-

tion 5.4. Section 5.5 summarizes the methods used to design and manufacture new chip geometries. Section 5.6 defines the two types of assays used in the screening phase of the workflow as well as the blood and bacterial separation experiments used to compare the performance of the new geometry to that of the baseline. Section 5.7 outlines the methodology used to screen separation performance from microscope images. Finally, Section 5.8 presents the results of the device screening as well as the winning design's performance when compared to that of the baseline geometry.

5.4 Chip Design and Theory of Operation

TODO Jason: Help here please!

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 red blood cells (RBCs) and white blood cells (WBCs), to the center port, while smaller particles (ex., platelets, bacteria, etc.) are collected at the side port. This trifurcated device is used in the blood and bacterial separation experiments described in Section 5.8.4. In order to reduce manufacturing complexity, devices screened using the methods outlined in Section 5.6.3 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.

5.5 Rapid Prototyping

The success of this study relied upon the ability to quickly design and fabricate iterations of chip geometries. Previous design optimization efforts were hampered by the process of out-sourcing device fabrication. Each design iteration required a fully specified mechanical drawing and any changes to the design demanded regeneration

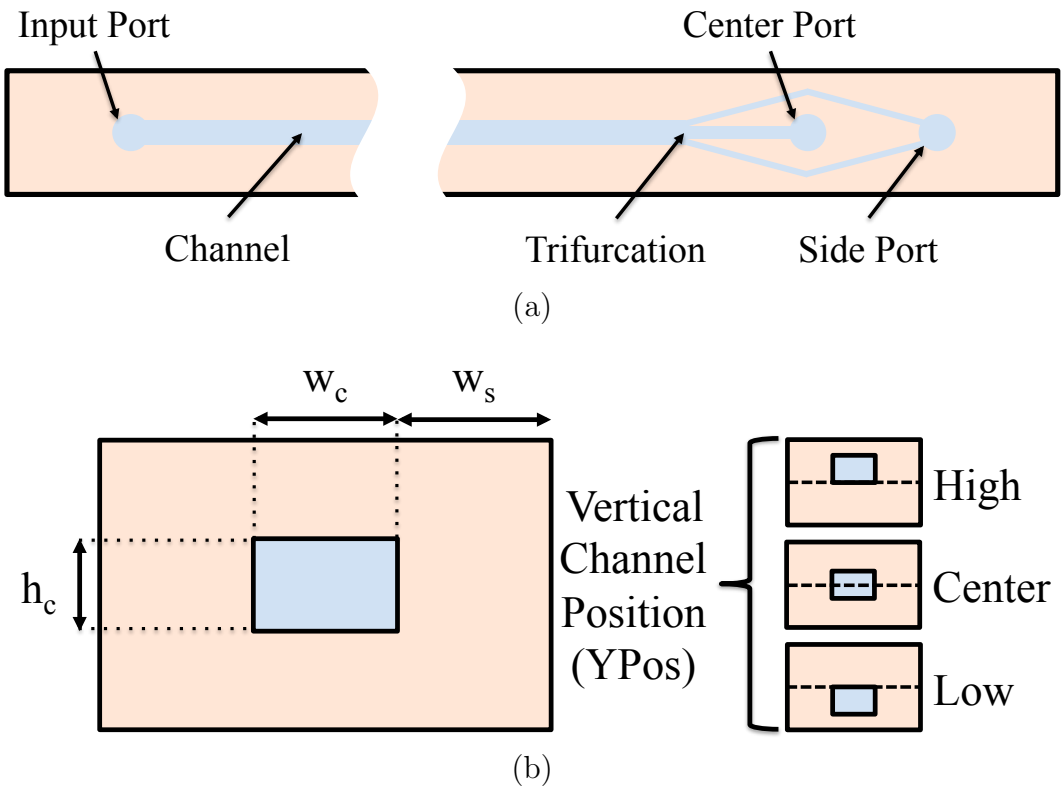


Figure 5·2: (a) Complete trifurcated microfluidic separation chip. (b) Definitions for each two-dimensional geometry included in the study. Note that the definitions apply to fluidic channel upstream of the trifurcation.

of the solid model. Once the drawing for a single design iteration was prepared and submitted to a machine shop lead times in excess of two weeks were not uncommon while costs soared as high as $3,000 for a batch of 20 identical chips. This section describes how these limitations were mitigated using free and open-source design software in conjunction with a $2,500 desktop micromill.

5.5.1 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 De-

sign (CAD) tool is used to create a solid model of the device. Second, Computer Aided Manufacturing (CAM) software is responsible for generating the commands (also called toolpaths) that are sent directly to the micromill. Finally, a piece of control software is required to manage the connection between a computer and the micromill. This software sends individual toolpath commands from the computer 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 5·3 shows the solid model for a single device design created using the function call in Listing 5.1. 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).

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);
```

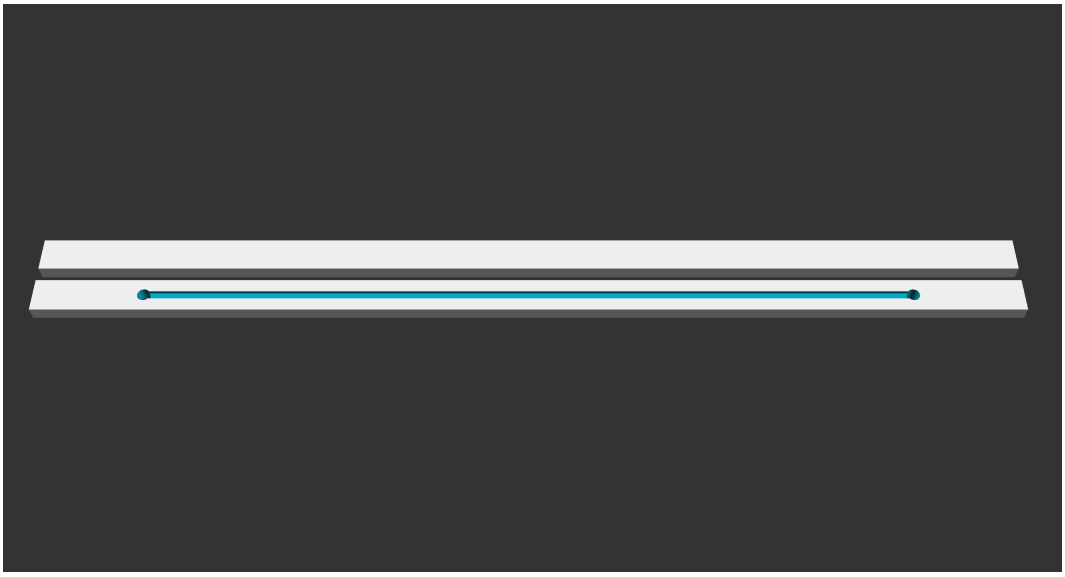


Figure 5·3: Output solid geometry of Listing 1, including bonding plate.

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);

module chipLayout(){
  for(i=[0:(numChips-1)]){
    let (ChipY=wc[i]+ws[i]*2){
      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.

5.5.2 Fabrication

Micromilling has demonstrated potential for prototyping 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 staggering. Micromills capable of achieving resolutions at or below 25 μm start at $15k. The large footprints and noise associated with these machines can make them inappropriate for use within a microfluidics laboratory. Self-contained micromills drive costs up another order of magnitude. 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, which only require a solid model generated from a CAD tool, such as SolidWorks, and a slicer which can generate toolpaths with a minimal amount of user input. 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 offer 25 μm 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.,). This study leverages these advancements to quickly manufacture distinct acoustofluidic device designs at a negligible cost when compared to outsourcing fabrication.

5.5.3 Materials and Assembly

Polystyrene (PS) was selected as an appropriate material based on its low attenuation and high impedance of ultrasonic energy (Wang et al., 2001). 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).

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.

5.6 Experimental Methodology

This study required four types of experiments: rapid screening, final screening, blood separation and bacterial separation. Rapid screening assays were conducted on two initial parameter sets labeled as "Initial Studies" in Figure 5·1. The methodology for this assay is outlined in Section 5.6.3 and the initial parameter sets upon which this assay were conducted are defined in Sections 5.8.2 and 5.8.3. The results of these two initial rapid screening assays were used to inform the parameter set of a final, more involved, screening assay described in Section 5.6.3 and the parameter set upon which this study was conducted is defined in Section 5.8.3.

Ultimately, the performance of the final screening assay's winning geometry was compared to that of the baseline geometry in three different experiments measuring a device's ability to focus blood using a quantitative visual inspection, focus blood by measuring RBC concentration in the center port, as well as separate bacteria from whole blood. Performance was scored based on parameters defined for each experiment. Later experiments (i.e., after rapid screening) measured a device's efficacy while minimizing dissipated power and maximizing throughput (i.e., volumetric flow rate). These measures of merit are fully defined in Section 5.6.1.

5.6.1 Measures of Merit

The two measures of merit used in this study are the volumetric flow rate and the average electrical power. Maximizing the volumetric flow rate will enable the largest volume of input sample to be enriched in the shortest amount of time. This is typically favorable in most diagnostic and therapeutic applications, as long as performance (i.e., enrichment of sample) is not reduced considerably. In this study, the volumetric

flow rate through a device under test was regulated by a syringe pump (PhD Ultra, Harvard Apparatus). Minimizing the power requirements of the system is another important measure of merit which has multiple implications. First, heat generated in the transducer and in the PS channel during actuation may lead to delamination 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. Second, if better performance can be achieved at lower power levels, this may enable an increase in the volumetric flow rate. We quantify dissipated power as described in section 5.6.1 below.

Volumetric Flow Rate

The volumetric flow rate, measured in μl/min, through a device under test was regulated by a syringe pump (PhD Ultra, Harvard Apparatus).

TODO: Anything else to say about flow rate? If not, then this need not be a numbered section

Average Dissipated Power

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:

$$P_{inst} = VI = V_{max} sin(\omega t) I_{max} sin(\omega t - \varphi), \tag{5.1}$$

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:

$$P_{avg} = V_{rms} I_{rms} cos\varphi, \tag{5.2}$$

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.

5.6.2 Blood Sample Preparation

All experiments in this study used human whole blood (Interstate Labs) anticoagulated with acidcitratedextrose. In each case, the blood was diluted to 5% hematocrit in phosphate buffer solution (PBS 7.4 pH Lot Number 1832496). Cellular concentrations were measured before and after dilution using an automated hematology analyzer (Sysmex XXX). The diluted sample was then transferred to a 10 mL syringe (BD 10 mL Luer-Lok tip syringe 309604) and introduced to the chip through PVC tubing.

5.6.3 Screening Assays

Two types screening assays were conducted, both of which analyzed device performance through the use of microscope images and derived pixel intensities as a surrogate for focusing performance. This method is based on the assumption that higher concentrations of red blood cells will appear visibly darker, and hence have a higher grayscale value, when better acoustic focusing is present. This assumption is based

on observations shown in Figures 5·5 and 5·6, which demonstrate that in conditions conducive to good focusing (ex., lower flow rate and higher dissipated power) the observable band of blood cells will appear narrower and darker and thus have a correspondingly higher grayscale value within this band. Data confirming this assumption is presented in Section 5.8.4, which measures the relative RBC concentrations between the center and side ports of the trifurcated chip design.

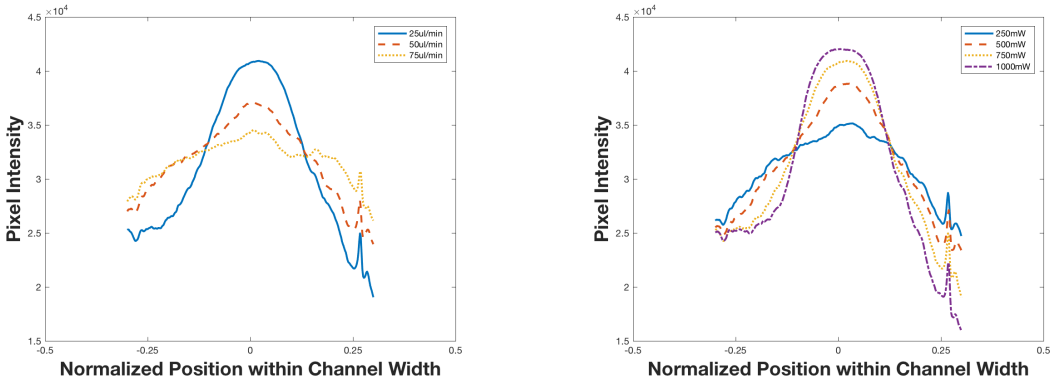


Figure 5·5: Pixel grayscale values across the width of the fluidic channel. (a) Power is held constant at 750mW, while flow rate is varied. (b) Flow rate is held constant at 25ul/min, while power is varied. Note that maximum pixel intensity, and thus focusing performance, increases as flow rate is decreased.

The two types of screening assays, rapid screening and final screening, each have different assumptions regarding the nature of their input parameter sets. The rapid screening assay does not make any assumptions regarding the functionality of a device under test. This means that it is reasonable to assume that a device will not exhibit acoustic focusing at any frequency within the tested bandwidth under the experimental conditions; thus, the performance parameter by which devices are scored seeks to determine the existence of acoustic focusing, rather than discern the relative quality of focusing between devices. The performance parameter used in this case is called

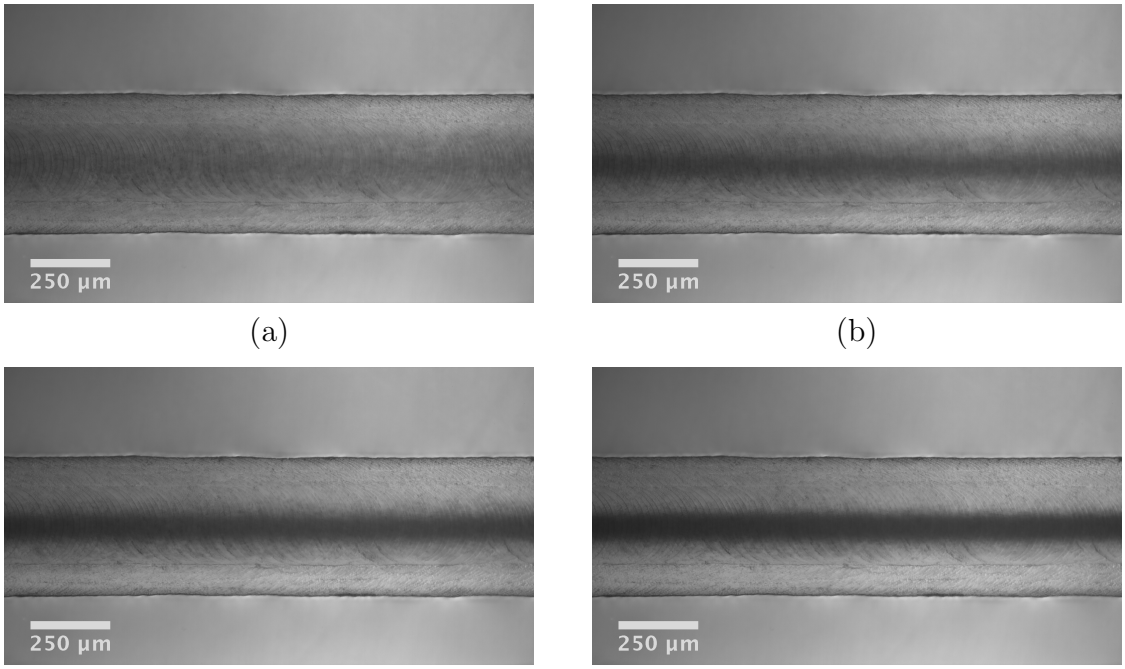


Figure 5·6: Microscope images of focusing performance at resonant frequency with increasing power at levels of 250mW (a), 500mW (b), 750mW (c), 1000mW (d). The plots in Figure 5·5(b) are derived directly from these images.

prominence and is defined in Section 5.7.1.

In contrast, the final screening assay attempts to compare the relative performance of devices that are assumed to exhibit some acceptable measure of focusing performance. Devices are scored based on the ratio of prominence to the width of the focusing band, as defined in Section 5.7.2. The relative merits between the two performance parameters are discussed in Section 5.7.

Rapid Screening Assay

The purpose of the rapid screening assay was to discover functional device designs, i.e., designs that could focus blood, and the frequencies at which they operate. This was accomplished by testing many devices under the same power and flow conditions while

sweeping through a frequency band around which the baseline geometry operates while remaining below the piezoelectric transducer's resonant frequency. The flow rate was set such that the particle velocity of each chip design was normalized to that of the baseline device operating at a flow rate of 100 μl/min. The input power was normalized by setting the average dissipated power to 1 W at 1 MHz, the operating frequency of the baseline geometry, and then maintaining this input power throughout the frequency sweep.

Taking into account the operating frequency of the baseline device (1 MHz) as well as the resonant frequency of the transducer (2.34 MHz), the chosen frequency band spanned from 0.50 – 2.00 MHz, inclusive. Within this frequency band pictures were taken in 10 kHz increments, from which a number corresponding to the maximum peak prominence was returned. The maximum peak prominence for the entire frequency band constituted the score for that particular design.

Final Screening Assay

The final screening assay served to discriminate between devices of assumed functionality. This was accomplished by manually tuning the device to its operating frequency 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 5.7.2.

5.6.4 Blood Separation Assay

Due to differences between side and center channel outlet dimensions at the point of trifurcation, outlet tubing length was cut to match the ratio of hydrolic diameters of each channel. Once flow fractions matched outlet dimensions, the channel was

primed with deionized (DI) water, followed by the sample. Once the sample being tested occupied the total volume of the microfluidic chip and inlet/outlet tubing, the flow rate was set to appropriate setting on the syringe pump. Control measurements were taken for each flow rate with the acoustics off to validate flow fraction as well as cellular behavior within microfluidic chip. Samples were collected in conical tubes and measured for flow fraction and cell quantity calculations. The resonant frequency was found visually by observing the focusing stream - the most compact stream reveals the resonant frequency. Once the resonant frequency was found, powers were swept by increasing RF amplifier gain to encompass the experimental parameters. For each sample, weights and hematology data was taken and entered into a database. MATLAB was used for further data analysis and plotting.

5.6.5 Bacterial Separation Assay

TODO Parker: Please fill in this section with a paragraph describing the experimental conditions for the plating experiment

5.7 Image Analysis

Image processing software (ImageJ (Abràmoff et al., 2004)) was used to create a plot profile of pixel intensities across the width of the channel, W_c. Devices were screened using the prominence of local maxima, as defined in Section 5.7.1, as a performance parameter. 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

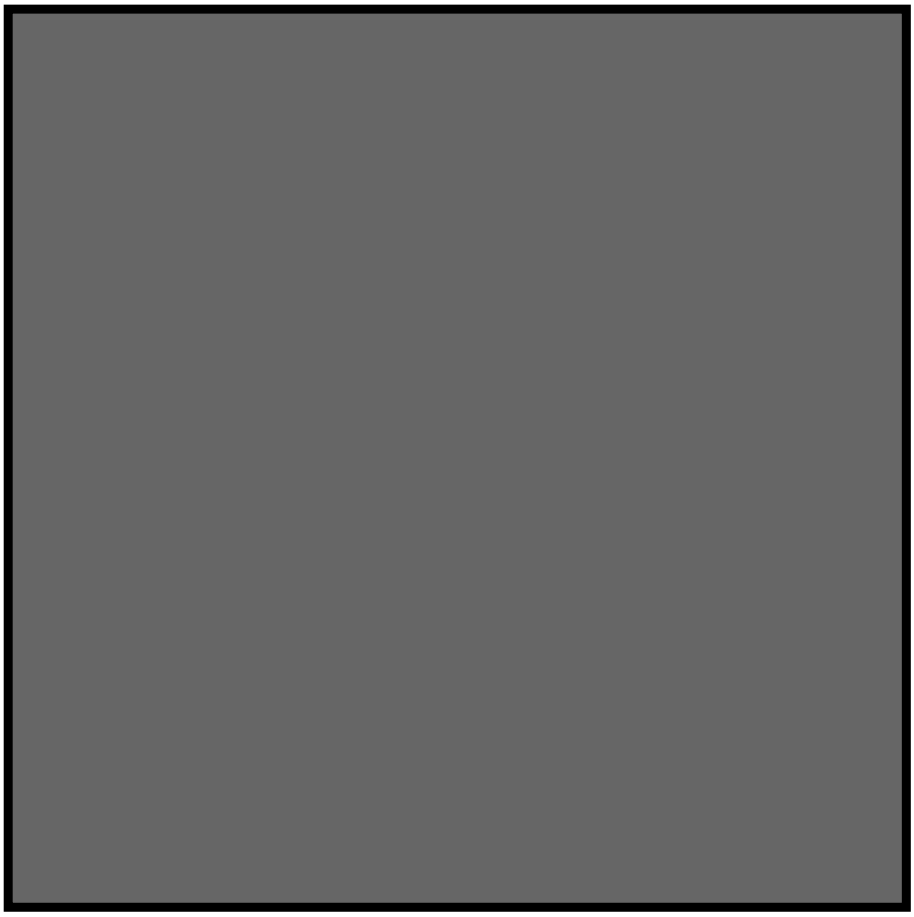


Figure 5·7: Render of the experimental setup including custom stage, piezoelectric transducer and microfluidic chip.

intensities caused by shadows.

Peak prominence is used as a direct measure of merit for device screening studies in which the assumption of a functional geometry (i.e. a geometry capable of focusing particles) cannot be made. When a device is determined to function well based on its prominence score it is compared to the baseline geometry using the ratio of peak prominence to half-prominence width, χ (Figure 5·8c.). χ cannot be used during device screening as the metric can skew results by rewarding peaks with relatively small prominence values and correspondingly small widths. However, this metric is useful for comparing the quality of the most prominent peaks among different,

commensurate, devices such as the winner of a design screening iteration and the baseline geometry.

Once a device compares favorably against the baseline in terms of χ, this final design's separation performance is then tested head-to-head against the baseline geometry using the full, trifurcated, designs shown in Figure 5·2(a). by comparing each design's ability to focus blood to the center channel, as shown in Figure 5·2, as well as each design's ability to separate bacteria from blood. These experiments are described in further detail in Section 5.8.4.

5.7.1 Definition of Prominence

Suppose an ordered signal is defined as in Equation 5.3, 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 (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 until either the end of the signal is reached or until the line intersects the signal itself, thus creating two sets of data points in the positive and negative directions. Minima 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.

$$s[n] = D \quad n = 0, 1, 2, \cdots, N-1 \tag{5.3}$$

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 $P \subset D$

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 $P \subset D$
begin
  $P \longleftarrow \emptyset$
  for $d_i \in D$ **do**
    Find all $neighbors$ for $d_i \mid |i - i'| = 1$ `AppendToP`$(neighbors)$
  end
end

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.

5.7.2 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 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 $|i^+ - i^-|$. It has been shown that for equivalent focusing performance, the prominence scales with the channel width (Ley and Bruus, 2016), therefore it is appropriate to normalize the peak width by the width of the channel. The final equation for χ is shown in Equation 5.4

$$\chi = Prom * \frac{W_c}{HalfPromWidth} \tag{5.4}$$

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

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

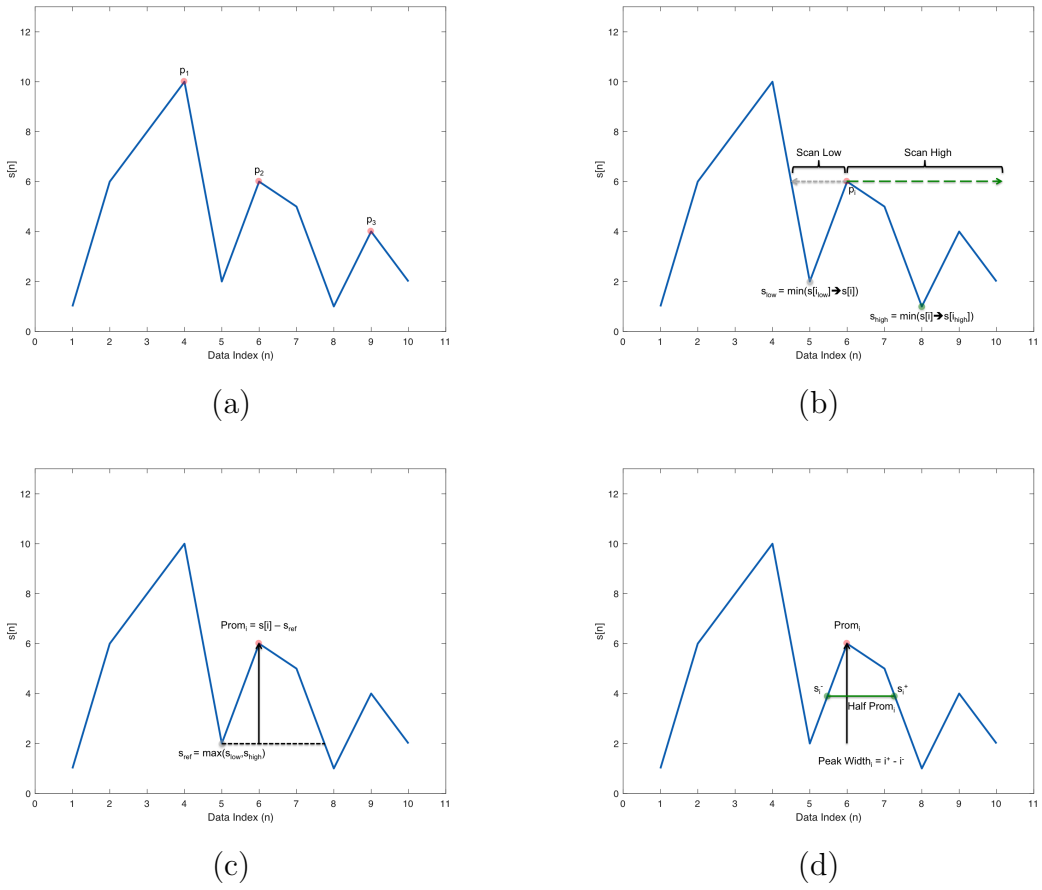


Figure 5·8: Algorithmic progression for calculating prominence and peak width at the point of half-prominence

5.8 Results

5.8.1 Screening Design of Experiments

The resonant response of acoustophoretic microfluidic devices has been shown to be non-linear and discontinuous (Garofalo et al., 2016). 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 (RSM) (Muriithi, 2015); 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 an optimal geometry, this iterative method is acceptable.

Experiments must be initialized such that the sampling of the solution space can detect curvature in the response, as described in Section 5.8.2. 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 5.8.3.

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.

5.8.2 Seeding of the Design Space

In order to economize the number of initial experimental runs, an orthogonal array was chosen in order to generate a useful parameter set for the initial iteration of 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 (Ting et al., 2004). 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.

Since the parameters generated during the initial seeding of the design space cannot assume a functional geometry (i.e., a geometry capable of focusing blood), the purpose of the initial iteration of experiments is directed towards discovering functional designs, as opposed to a comparison between functional designs. As such, the performance parameter used during this initial phase is peak prominence (Figure 5·8c.). This is because χ (Figure 5·8d.) can skew the results by rewarding peaks with small prominence values (i.e., poor focusing) and correspondingly small widths. The

Table 5.1: Geometries tested for L9 design array. All dimensions are given in units of μm. Prominence values are given in units of pixel intensity

W_c	H_c	W_s	Y_{Pos}	Prominence	Freq (MHz)
300	75	500	Low	6375	1.58
300	290	850	Center	5496	1.22
300	500	1200	High	8240	0.67
650	75	850	High	43021	0.55
650	290	1200	Low	12950	0.66
650	500	500	Center	13251	0.57
1000	75	1200	Center	14215	0.74
1000	290	500	High	9574	0.73
1000	500	850	Low	3144	0.71

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 parameter is achieved by setting W_c to 650 μm, H_c to 75μm, W_s to 850 μm and placing the channel in the high vertical position. 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 5.8.3.

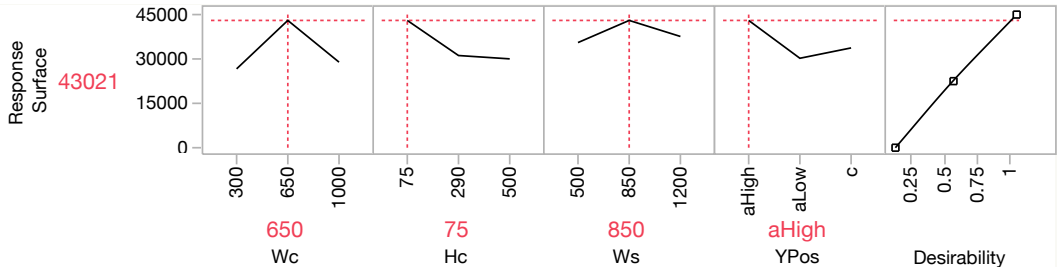


Figure 5·9: Results of initial experimental run. The response surface is determined through a least squares fit of the prominence data shown in Table 5.1 in four-dimensional space.

5.8.3 Variable Isolation Studies

Channel Width Study

Difficulties manufacturing chips with H_c of 75 μm, namely the total collapse of the channel during bonding, resulted in an increase of H_c from 75 μm to 100 μm for the subsequent variable isolation study; the assumption being that such a small change to the height of the channel would not result in a large change in performance. However, Tables 5.1 and 5.2 show a significant performance degradation as a result of the change for the designs having a channel width of 650 μm. As all other parameters were held constant, it can be assumed that the performance degradation is a result of increasing the channel height from 75 μm to 100 μm.

As shown in Table 5.2, the highest performing channel width, holding other factors

Table 5.2: Geometries tested for an isolation study of channel width. All dimensions are given in units of μm. The frequency range scanned spanned from 500 kHz to 2 MHz. No fundamental odd mode was observed within this range for the design with a channel width of 700 μm.

W_c	H_c	W_s	Y_{Pos}	Prominence	Freq (MHz)
500	100	850	High	12143	0.64
550	100	850	High	15200	0.51
600	100	850	High	4770	1.58
650	100	850	High	5470	1.99
700	100	850	High	-	-

constant, was 550 μm. 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 final screening assay.

Channel Height Study

The final screening assay, as specified in Section 5.6.3, was conducted by fixing W_c, W_s, and Y_{Pos} while varying H_c. The geometries studied are shown in Table 5.3. The results shown in Figure 5·10 demonstrate that, for flow rates higher than 25 μl/min, the chip with a channel height of 250 μm 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 550 μm, a channel height of 250 μm, a side-wall width of 850 μm, with the channel in the High vertical position.

Table 5.3: Geometries tested for an isolation study of channel height. All dimensions are given in units of μm.

W_c	H_c	W_s	Y_{Pos}
550	100	850	High
550	150	850	High
550	200	850	High
550	250	850	High
550	300	850	High

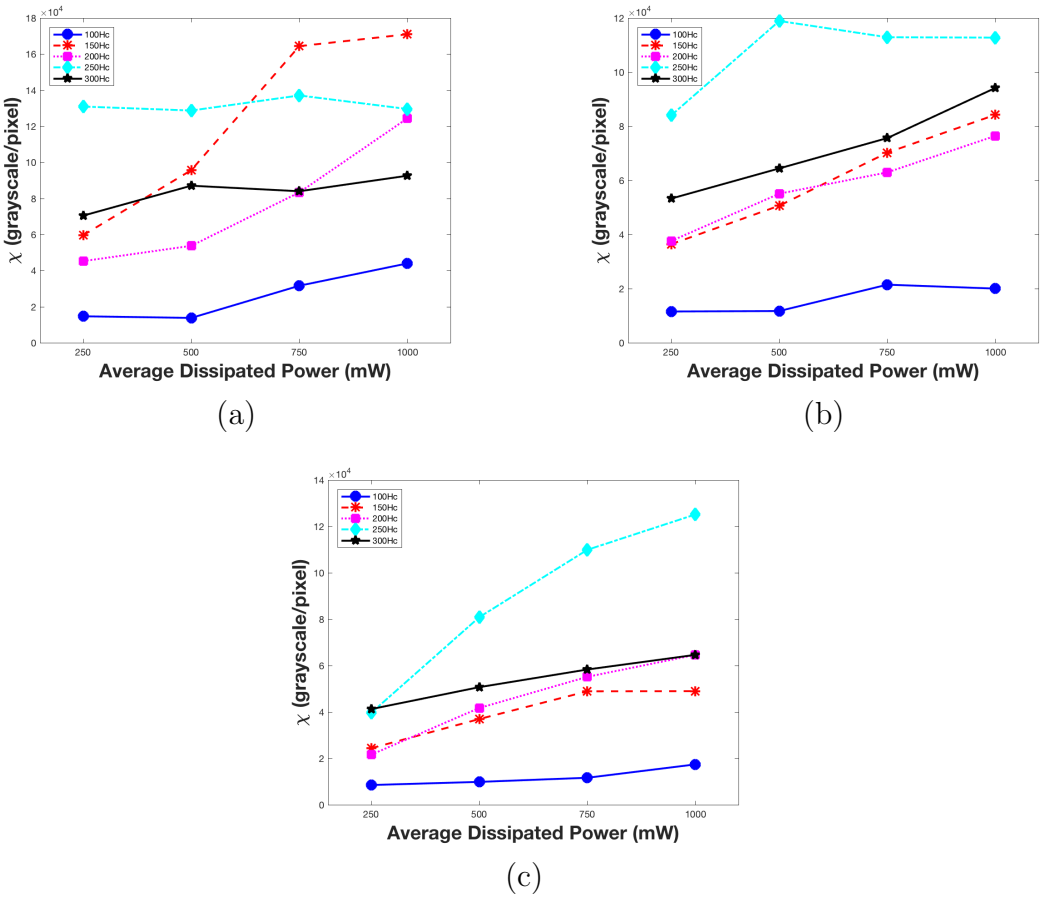


Figure 5·10: Performance Comparison of Channel Height Study using Image Analysis. (a-c) plot the performance of the geometries in Table 5.3 for three volumetric flow rates: 25 μl/m, 50 μl/m and 75 μl/m, respectively. Performance is calculated from microscope images in the manner described in Section 5.7.2.

5.8.4 Comparison of Winning Geometry versus Baseline Geometry

The results of the screening assays yielded a geometry that outperformed the other devices under test. However, in order to gauge the ultimate success of this geometry it must be tested against the baseline geometry. This section presents results from three experiments that compared the winning geometry of the screening assays against the baseline geometry through image analysis, blood separation, and bacteria separation.

Comparative Focusing

Figure 5·11(a–c) plots the performance of the winning geometry, hereafter referred to as 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.

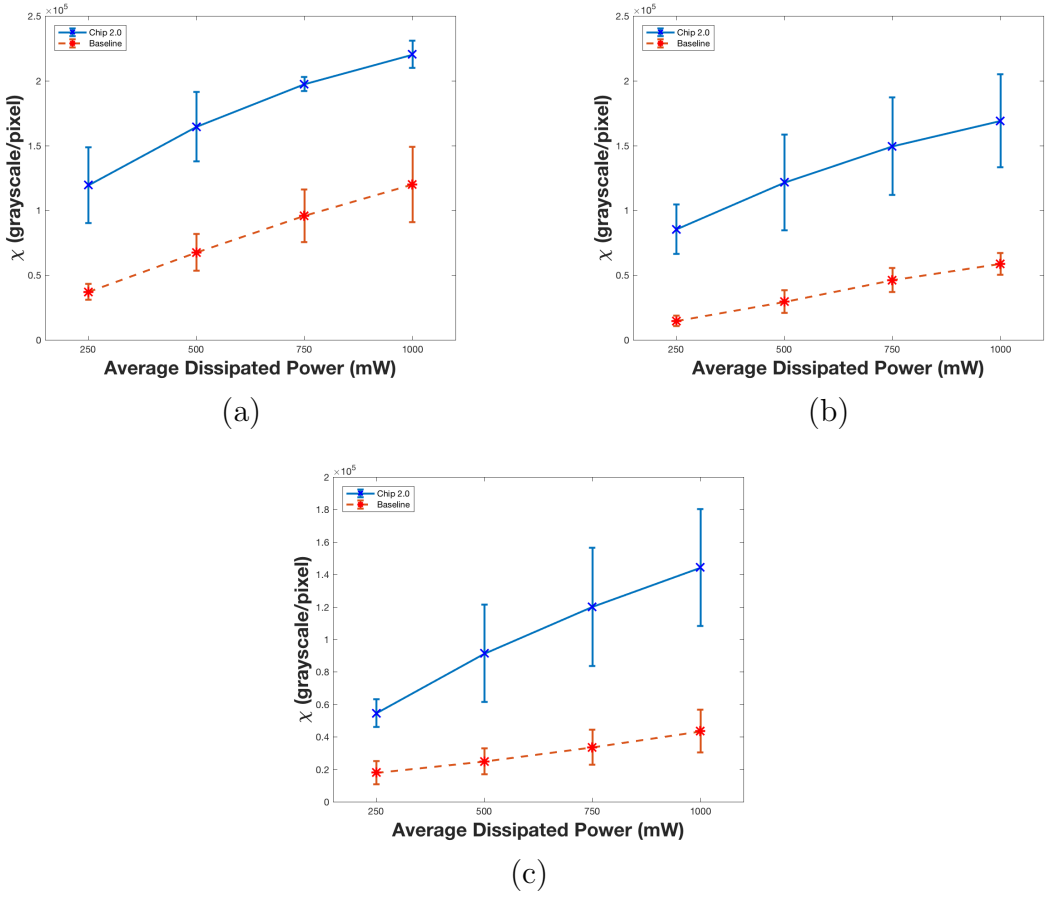


Figure 5·11: Performance Comparison Versus Baseline using Image Analysis. (a-c) plot the performance of the baseline versus the winner of the study for three volumetric flow rates: 25 μl/m, 50 μl/m and 75 μl/m, respectively. Performance is calculated from microscope images in the manner described in Section 5.7.2.

Comparative Blood Separation

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 to that which falls between the performance at the control measurement (i.e., zero average dissipated power) and 100% RBC concentration in the center channel. As the chip design used, shown in Figure 5·2, has a single input port and two outlet ports, RBCs will be equally distributed between the two outlet ports 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. **How can I quantify the percent change in performance across all experimental conditions?**.

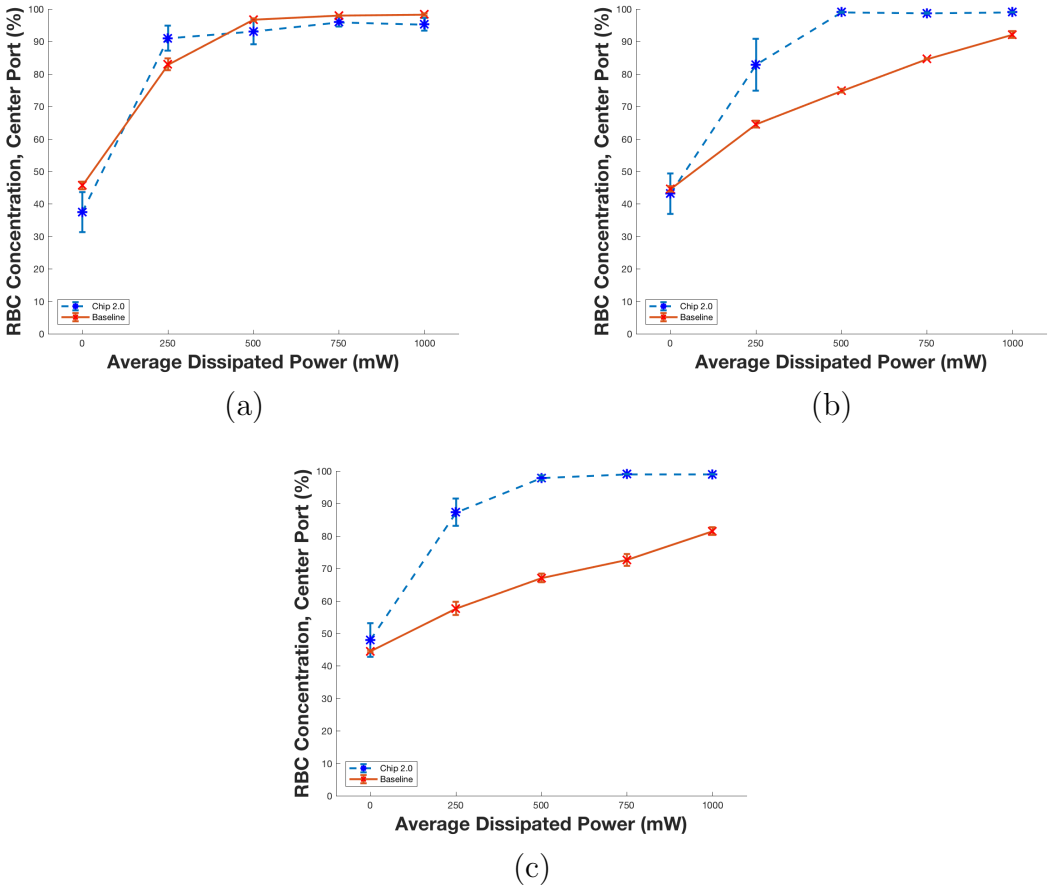


Figure 5·12: Separation Performance Comparison Versus Baseline. (a-c) plot the performance of the baseline versus the winner of the study for three volumetric flow rates: 25 μl/m, 50 μl/m and 75 μl/m, respectively. Performance is defined based on each design's ability to focus red blood cells into the middle channel of the trifurcation shown in Figure 5·2.

Comparative Bacterial Separation

TODO Parker: Please populate this section with the results of the plating experiment. We can work together on the discussion

5.9 Conclusion

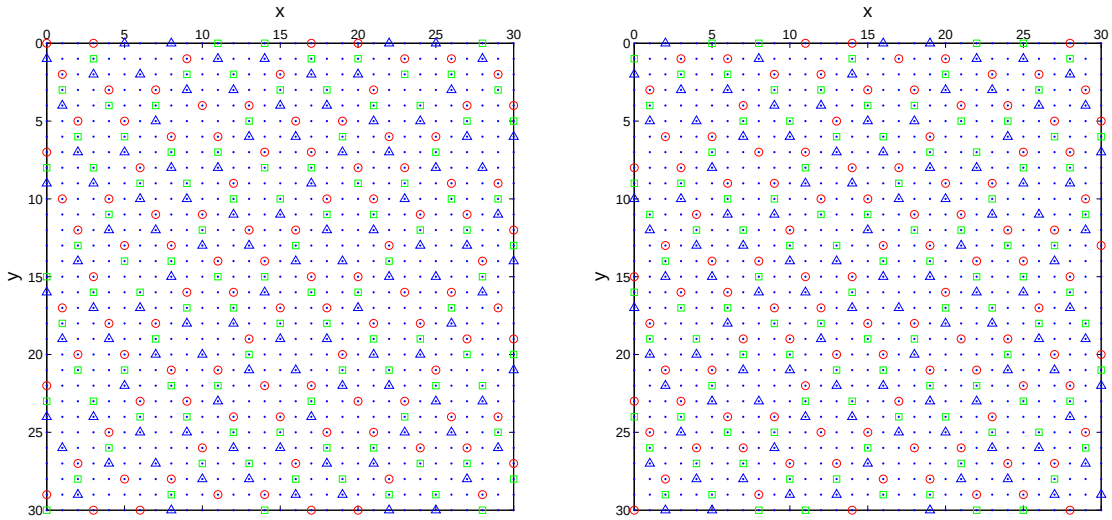
The results of the comparison experiments show that Chip 2.0 improves performance relative to the baseline geometry by **XXX**% across all three metrics: focusing, blood separation and bacterial separation.

Chapter 6

Body of my thesis

6.1 Some results

Here goes all the important stuff, likely with a lot of graphics like this:



(a) (b)

Figure 6·1: Assignment of single-view intensities to RGB components: (a) view #1; and (b) view #2.

You will also be using a lot of citations. Here is the format required in the dissertation: (Lamport, 1985),(Debreuve et al., 2001).

In all likelihood, you will need to insert tables. See one example on the next page.

Table 6.1: Absolute disparity error per pixel for the test data from Fig. 6·1 and different parameter values. In each experiment one parameter is adjusted while other parameters are unchanged.

$\eta = 6000$, $\mu = 2000$

K	u_1	u_2
3	0.52	0.46
7	0.47	0.43
10	0.35	0.36
12	0.37	0.36

$K = 10$, $\mu = 2000$

η	u_1	u_2
1000	0.54	0.45
3000	0.43	0.40
6000	0.35	0.36
9000	0.37	0.37

$K = 10$, $\eta = 6000$

μ	u_1	u_2
100	1.00	1.16
1000	0.53	0.47
2000	0.35	0.36
3000	0.44	0.43

Of course, there must be a Table of Contents at the beginning of the thesis.

Chapter 7

Conclusions

7.1 Summary of the thesis

Time to get philosophical and wordy.

IMPORTANT: In the references at the end of thesis, all journal names must be spelled out in full, except for standard abbreviations like IEEE, ACM, SPIE, INFOCOM, ...

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.

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.

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., Augustsson, P., Laurell, T., and Bruus, H. (2010). Measuring the local pressure amplitude in microchannel acoustophoresis. *Lab on a chip*, 10(5):563–570.

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

Carley, K. M., Kamneva, N. Y., and Reminga, J. (2004). Response surface methodology. Technical 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.

Debreuve, E., Barlaud, M., Aubert, G., Laurette, I., and Darcourt, J. (2001). Space-time segmentation using level set active contours applied to myocardial gated SPECT. *IEEE Trans. Med. Imag.*, 20(7):643–659.

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.

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*.

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.

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

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.

Huang, H. and Densmore, D. (2014). 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.

Koerner, T., Brown, L., Xie, R., and Oleschuk, R. D. (2005). Epoxy resins as stamps for hot embossing of microstructures and microfluidic channels. *Sensors and Actuators B: Chemical*, 107(2):632–639.

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.

Lamport, L. (1985). *LaTeX—A Document Preparation System—User's Guide and Reference Manual*. Addison-Wesley.

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

Laurell, T., Petersson, F., and Nilsson, A. (2007). Chip integrated strategies for acoustic separation and manipulation of cells and particles. *Chemical Society Reviews*, 36(3):492–506.

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.

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.

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.

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.

Muriithi, D. K. (2015). Application of response surface methodology for optimization of potato tuber yield. *American Journal of Theoretical and Applied Statistics*, 4(4):300–304.

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.

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.

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.

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).

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. 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.

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.

Ting, J.-H., Shiau, S.-H., Chen, Y.-J., Pan, F.-M., Wong, H., Pu, G. M., and Kung, C.-Y. (2004). Preparation and properties of sputtered nitrogen-doped cobalt silicide film. *Thin solid films*, 468(1):155–160.

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.

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.

Wang, H., Ritter, T., Cao, W., and Shung, K. K. (2001). High frequency properties of passive materials for ultrasonic transducers. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 48(1):78–84.

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*, pages 1–6.

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

8 St. Mary's St.
PHO 209
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: Exploring the Boundaries of Computer Engineering, Biology and Microfluidics

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