Mechanical Energy Storage for Self-Destructing Motes and Jumping Microrobots

by

Joseph Greenspun

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kristofer S. J. Pister, Chair
Professor Vivek Subramanian
Professor Liwei Lin

Summer 2018

Mechanical Energy Storage for Self-Destructing Motes and Jumping Microrobots

Abstract

Mechanical Energy Storage for Self-Destructing Motes and Jumping Microrobots

by

Joseph Greenspun

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kristofer S. J. Pister, Chair

Mechanical energy storage has been studied to enable a self-destructing mote and a jumping microrobot. Just as chemical energy stored in batteries can be used for a wide array of devices, so too can mechanical energy stored in beams. In this work various energy storage elements are designed, fabricated, and tested to create these MEMS systems.

As privacy and data security become increasingly important, new means of keeping that data safe must be developed. A MEMS system has been created to allow a wirelessly enabled mote to destroy itself on command. To achieve this, a cavity is microfabricated, filled with a silicon etchant, and capped with a fracturable membrane. An energy storage device capable applying 100's of milinewtons of force across distances of 10's of microns was designed and fabricated to fracture these membranes. Additionally, an electrostatic latch was developed to electrically trigger the release of the stored energy. The voltage required to keep this latch closed was reduced using a series of lever arms to amplify the electrostatic force.

Two versions of a silicon jumping microrobots were developed as well. The first microrobot had no active force-producing components and used identical energy storage elements as the self-destructing mote project. In the second microrobot design, the energy storage elements were redesigned and optimized to work with an electrostatic inchworm motor. This motor was combined with a rack and pinion system to create a motor system capable of amplifying the force from a standard inchworm motor by a factor of 10. This microrobot was capable of storing 1.0 μJ of mechanical energy and jumping 1 mm when its motors were actuated electrically through tethered inputs. When the energy storage elements were loaded manually and latched using one of the inchworm motors, 4.0 μJ of energy were stored and the microrobot jumped 6.5 mm.

Finally, a design and simulation library was created throughout this work specifically for microrobots. This library, written in MATLAB, can be used to programmatically generate layout files as well as simulation files. While this functionality exists in other software packages, the MATLAB environment enables calculations to be done in-line with the layout. Users can easily add new functions and build upon the existing software. The simulation environment uses a solid body physics simulator to test functionality of microrobots in

software before they are fabricated. This helps ensure that new designs work as intended before going through the time intensive and costly process of fabrication in the cleanroom.

To those struggling in grad school

...it gets better.

Contents

List of Figures

List of Tables

Acknowledgments

This could easily be the longest section of this dissertation, but I will do my best to keep it brief while trying not to exclude any contributors. I apologize in advance if I leave anyone out. First, I would like to thank Professor Kris Pister, my advisor. He was exactly the kind of mentor I needed him to be every step of the way. I would not have made it through grad school without his constant support, guidance, and contagious enthusiasm. I would also like to thank my qualifying exam and dissertation committees: Professor Vivek Subramanian, Professor Sanjay Govindjee, Professor Ana Arias, and Professor Liwei Lin.

Furthermore, I would like to thank my fellow researchers in the Pister group. I feel so lucky and honored to have spent my time at Cal with such a talented group, and I feel even luckier to know that I have made some lifelong friendships along the way. The late nights in the lab working towards a deadline would have been intolerable without my fellow microroboticists by my side. Daniel Drew's dedication and drive continue to inspire me as I leave graduate school. Since the early days of building ionocraft together, I have had the pleasure of working with him on both academic and personal projects. I feel incredibly fortunate to have had him as a friend and coworker throughout my graduate career. Daniel Contreras laid much of the groundwork for this work. He developed the assembly technique and the motors used both in this work and by others in the group. Having him in the cleanroom and at conferences made both experiences far more enjoyable than they would have been otherwise. Hani Gomez has taught me more than she will ever know and I am excited to see how the rest of her graduate career unfolds. The group would not be the same without her unique perspective. I could not have asked for a more perfect peer advisee. Craig Schindler and Brian Kilberg have made a great research team and I have thoroughly enjoyed struggling through conversations in Spanish with them during lunch, and struggling through the third lap of a very uncomfortable mile at the Clark Kerr Track. As for the circuits subgroup, I guess they were alright too. Brad Wheeler was the first one to venture into the cleanroom with me and I would not have survived that experience without his calming presence. He was also an excellent GSI for EE240. David Burnett was a great prelim study partner and an even better Artemis captain. Filip Maksimovic was the best desk neighbor anyone could ask for. He took care of my plant when I was gone and was always there to talk when I needed it. To all the other members of the Pister group, past and present, thank you for the support and the lively group meetings.

Additional thanks goes to other researchers at Cal who have been key to my success here. Bobby Schneider, was an excellent visit day host and showed me around the cleanroom. Steven Volkman was a great resource for any question I might have about research or otherwise. Nishita Deka was a pleasure to have in the classroom as well as in the cleanroom. She was an excellent co-super user and always there to help when a tool went down or a process failed. The Marvell Nanolab staff was critical to the success of this work. Bill Flounders, Greg Mullins, Jay Morford, Jason Chukes, Joanna Bettinger, Rich Hemphill, Ryan Rivers, Allison Dove, Emily Beeman, and Kim Chan were all willing to go out of their way to help in times of need.

I would also like to thank the entire physical education department at Cal. Without them and their classes I would have deteriorated both physically and mentally during the past 6 years. Special thanks goes to Toni Mar who taught me to love core conditioning. I am sorry I could never fully straighten my legs during the L's. Justin Caraway taught me how to play volleyball. I may still set with only 3 fingers, but it looks much better than it used to. Thank you for not cutting me from your classes, and thank you for not letting me purchase an entire mason jar that time at the Edge. Elmar Stefke taught me how to swim correctly and made me appreciate the Butterfly, even though it feels like drowning. Additionally I would like to thank the instructors at Hella Yoga. Ana and Kelly, I had a hella good time in your pilates classes 100% of the time.

I am deeply indebted to all of the friends I have made during my time here. They have supported me in small and big ways from day one, and I am forever grateful for that. Thomas Rembert was a constant source of support and inspiration during grad school. I cannot imaging what life here would have been like without him. From playing the lights to speaking in sentences with exactly 5 syllables, we have made some lifelong memories together. Emily Marron was a brilliant ray of sunshine that burst into my life half way through grad school. We have gone on some incredible adventures together and seen the sunrise over the most beautiful landscapes. She is someone who can immediately tell that something is wrong, and knows exactly what to say to fix it. Aaron Smyth was the friend that never should have been. What began as a joke encounter turned into one of my most cherished friendships. Though our differences abound, we bring out the best in each other and I am a better person, friend, and researcher for having known him. Keith Cheveralls was always there to remind me that I was looking at a hill, not a mountain. I will fondly remember our conversations at the rim of the fire trail and our lack of conversation immediately following the particularly steep climbs. To the volleyball crew, thank you for your patience with my bad passes and inconsistent hits. Colton Gover was less patient but made up for it off the court. He may have tagged me multiple times during games, but he is a fiercely loyal friend. Martha Deans was a team player through and through. She would kindly offer advice on the court and was the first (and only) one to volunteer to potentially push that electric car to the charging station in Spain. Roman Pina was always a pleasure to be across from at the net and is one of the sweetest men I have ever had the pleasure of knowing. To my Dungeons and Dragons family, I would not have wanted to defeat the evil Karzoog with any other gang of misfits. Thank you for providing much needed comic relief after long days in the lab. To our DM, thank you for not killing us when you probably should have. To Archie, thank you for only turning into the fighting machine once. To Gnara, thank you for usually remembering not to stand in front of Kody. To Deeba, thank you for not leaving us (me) when we (I) ruined all of your Saving Finales. To Wakunda, thank you for your fearlessness and willingness to barge into a room with no plan. To Sneg, Epona, and Yvonn thank you for being loyal companions even though we forgot about you most of the time.

This section would not be complete without a thank you to all of the friends and family who have supported me not only during grad school but during the years before as well. Michaela Papa and Lacey Vogel were always there for me when I needed them most, even

though they lived thousands of miles away. They provided help with everything from relationships to research and I am so lucky to still have them in my life. Chris Michael and Kristina Krajcik did the same, but I was fortunate enough to see them more often. The constant trips Chris made up to the bay helped me destress and talk through my problems in person. Visiting Fresno was also a nice escape from Berkeley even when it was 110 degrees outside. Kristina was always so supportive and encouraging. Her baked goods got me through the exceptionally tough times. If the three of us managed to survive that fateful bus ride home from San Francisco in 2012, I know our friendship can survive anything. Nicholas Luzod and Megan Garland provided a much needed escape from grad school with trips down into the Grand Canyon and up into the Cascades. I could not ask for better adventuring companions. Shannon Grover was always willing to drop what was going on in her life to provide love and support when it was most needed. Michael Bhatt and David Krinjak have been two people I can always count on to go for a rage run and talk about whatever is on our minds. My family has been there for me at every step of this process. My mom always offered to write a note to the conference coordinators asking for an extension. My dad was always there to listen, even if we could not agree on kilowatts versus kilowatt-hours. My sister Nic was instrumental to my success. Without her I could never have gotten through my first year. I have already learned so much from her and have so much more to learn still. My brother-in-law Jonathan was always a source of curiosity and intrigue with regard to my research. Talking to him always reenergized my excitement in my own work. My little brother Luke always made sure I had enough of my favorite kind of pen so I could do my research and once saved me and my entire cross country team when we were broken down on the side of the highway. Finally, none of this would have been possible without my boyfriend Riley. His support was unwavering even when he had his own stresses to deal with. On top of listening to my complaining, he came into the cleanroom numerous times just so I could continue my work. I cannot thank him enough for all of the time, effort, and love that he has invested into me and my work.

Chapter 1

Introduction

Energy storage is an integral part of our daily lives. Most of us plug in our devices overnight with the hope that the stored chemical energy will last us through the next day. Battery technology has enabled a wide array of portable devices from MP3 players, to cellphones, to fitness trackers. We use the same batteries to listen to music, make calls, and track our steps. And although chemical energy stored in batteries may be the energy we worry about most in a day, we also rely on stored mechanical energy to perform a variety of tasks. Just as the chemical energy in a battery enables a diverse set electronic devices, we leverage stored mechanical energy when we use a wristwatch, go for a jog, and drive a car.

This work focuses on various means of storing and releasing mechanical energy at the microelectromechanical systems (MEMS) scale as well as applications and systems that utilize this stored energy. Chapter 1 details the background information for the two main systems developed here, a mote that can self-destruct, and a jumping microrobot. The rest of the thesis details the steps taken to design, fabricate, and test these devices. Chapter 2 explains the self-destructing silicon project and introduces the basic MEMS process used throughout this work. Additionally, it explains additional process steps to enhance the performance of that latches required for the project. Chapter 3 introduces the design and testing of a jumping microrobot. Chapter 4 discusses the MATLAB library that was developed through the course of this work to generate layout and simulation files. Finally, Chapter 5 concludes the findings and lays out next steps for the jumping microrobot project.

1.1 Self-Destructing Silicon

When designing an electronic system, many factors are often considered such as device lifetime, power requirements, and temperature sensitivity. One factor almost never considered is the system's ability to destroy itself. As we delve deeper into the information age and our data and privacy become all the more important, this ability may become a core feature upon which we rely. Generally when we think about protecting our data, we rely on encryption. Encryption can go a long way in protecting our 1's and 0's, but what happens

when the hardware itself is part of the information that needs to be keep secret and safe.

There are some on-chip technologies that can be implemented to help with this problem. The eFUSE, developed by IBM, is a one time programmable fuse capable of permanently changing the functionality of a piece of circuitry. This device uses electromigration, usually an undesirable effect, to permanently change a particular electrical path. The eFUSE was originally developed for applications such as repairing memory arrays, assigning electronic Chip ID, and implementing fault tolerance [1]. This technology could theoretically be used to permanently render an integrated circuit (IC) inoperable. The idea of permanently bricking a piece of electronics with this technology was at the forefront of a Motorola controversy in 2010. A Motorola cell phone, the Droid X, used the eFUSE technology to ensure that owners did not install their own modified bootloading software onto the phone. If a non-stock bootloader was found, the eFUSE would only allow the phone to boot into safety mode. The customers, however, started rumors that Motorola would use the eFUSE technology to render their devices permanently inoperable if the bootloader was modified [2]. While these were just rumors, the eFUSE could theoretically be used to do exactly that.

If stopping device functionality is the goal, the eFUSE is a compelling solution, however some technology is so valuable that competitors will go to great lengths to figure out exactly how the device was made. This has historically been common among semiconductor companies in the CPU space [3]. When one company designed a new chip, the others would purchase that product and reverse-engineer it. Through various grinding, staining, and imaging methods, those companies could know the exact location of every metal trace, via, and transistor. They could directly copy the design, or use it as a starting point to create an improved version of the IC. Even if this IC were to contain an eFUSE or similar technology, it would not inhibit the reverse engineering process. To create a chip that can not be reverse engineered, an entirely new approach was adopted.

Using silicon based micromachining, a package level solution was implemented to create a self-destructing IC. The MEMS component utilizes a mechanical energy storage and rapid release device called a MEMS Hammer. This device fractures a cavity containing xenon difluoride, an aggressive silicon etchant. The etchant sublimates at room temperature and will attack any silicon based IC within the package. This etch method turns the solid silicon IC into a gaseous etch product rendering it completely non-reverse engineerable.

1.2 Jumping Microrobots

Researchers have been excited by the idea of creating, controlling, and manipulating objects at small scales since the mid 1900s [4]. From these early ideas evolved the dream of creating a fleet of tiny mobile robots that could operate autonomously, sense their surroundings, and communicate to accomplish group tasks. Today, these microrobots could be used in manufacturing, search and rescue, and medicine. However, as with many technological advances, their true niche may not be discovered until after their development. Even when

microrobots were first proposed in the early 1980s, their development was likened to that of the computer and the hard-to-foresee creation of the video game industry [5].

Although microrobots have been actively researched for decades, many challenges to their fundamental operation remain unsolved. Among these challenges is the basic ability to locomote. Researchers have had varying degrees of success creating walking, jumping, flying, and swimming microrobots. While all locomotion modalities have certain benefits and drawbacks, this work focuses on jumping microrobots. Jumping offers the ability to maneuver over obstacles many times the size of the microrobot which will be a crucial ability when navigating through many real-world environments. Additionally, this work aims to lay the groundwork for a fully autonomous jumping microrobot. That is to say, the components of this microrobot are designed with the ultimate goal of having power, control, actuation, and energy storage all on-board.

Generally, the term microrobot is not well defined. Some claim the entire robot should be on the micron scale to qualify, others include much larger robots. In this work, the term microrobot refers to any robot on the millimeter size scale. Robots at this scale have a distinct advantage over larger robots, especially if they utilize standard microfabrication techniques. Using a MEMS foundry, one could be easy to produce tens of thousands of these robots daily. Their utility comes from the fact that there are so many of them. If an individual robot breaks or veers off course, there are thousands more ready to take its place and continue the job it was doing.

Prior Work

Although the jumping microrobot shown here is not strictly biomimetic, one can look to biology for tips in designing these small scale jumpers. Generally speaking, animals on the millimeter scale that jump from place to place do not do so by firing off muscles that actively propel them into the air. Looking at the common flea as a model organism, scientists were originally puzzled at how the muscles in a flea could produce the accelerations scientists were measuring. It should have been impossible for any muscle in the flea to create such a force in a single contraction [8]. They later learned that the flea manages to jump by storing energy into a biological material called resilin. It does this slowly over a time period of a few hundred milliseconds, and releases that stored mechanical energy all at once over the course of less than 1 millisecond [9]. Motors that can be readily fabricated at the microscale suffer the same limitations as these muscles; MEMS motors cannot generate the forces and accelerations required to make a MEMS device jump. So various means of energy storage and rapid release have been used in the designing of jumping microrobots.

Noh et al. from Seoul National University designed a jumping microrobot that mimics the catapult-like jumping mechanism of fleas [6]. This microrobot is 20 mm long and weighs 1.1 g. Using shape memory alloy (SMA) springs as muscles, the researchers placed these springs in the same orientation as the flexor, extensor, and trigger muscles in a flea. When a current is passed through these springs, they contract and store mechanical energy. When the trigger spring is activated, it quickly release this energy and the microrobot jumps. It

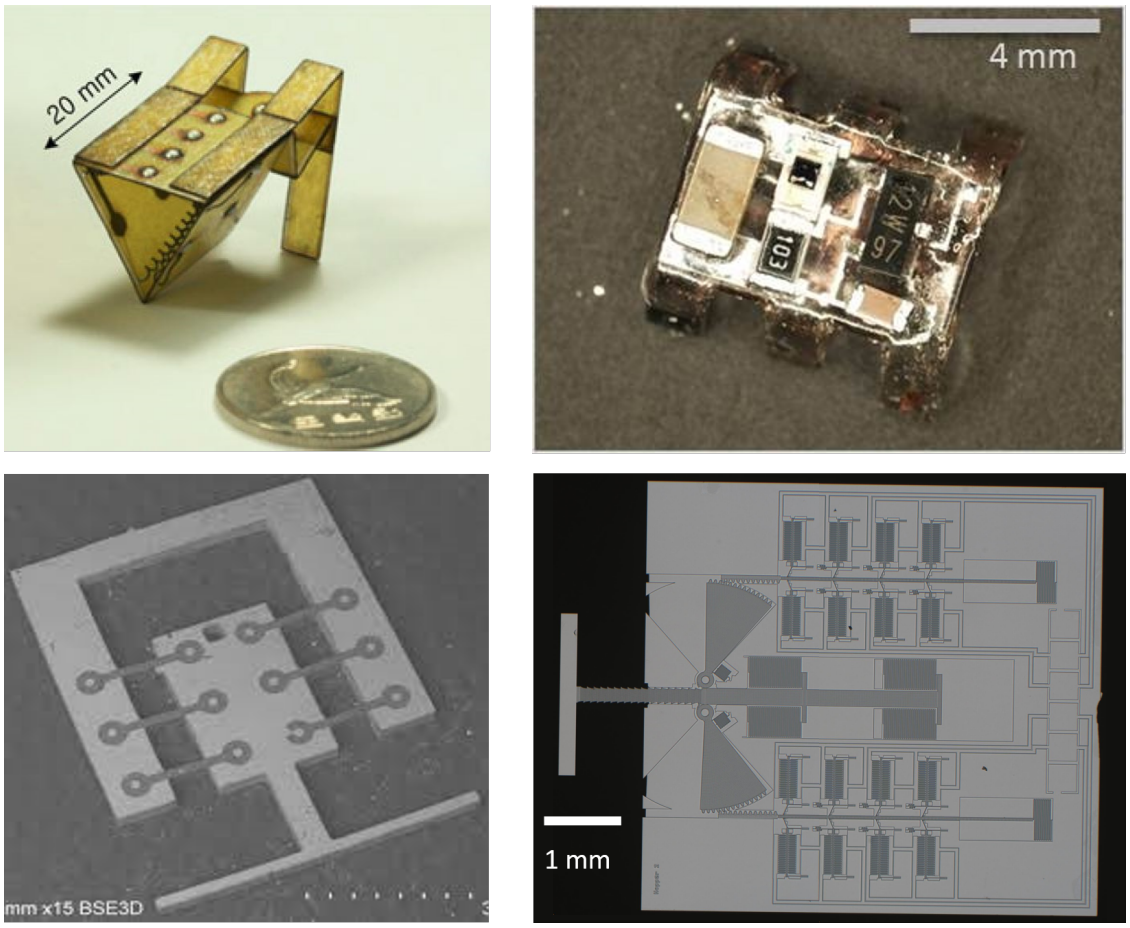

Figure 1.1: Prior work and current research in jumping microrobotics. Top left, a catapult-like jumping microrobot that uses shape memory alloy to jump 64 cm [6]. Top right, a light sensitive jumping microrobot that uses chemical energy to propel itself 8 cm [7]. Bottom left, a microfabricated silicon based jumper that incorporates PDMS springs for energy storage and is capable of being launched 32 cm [7]. Bottom right, the microrobot designed and fabricated in this work which uses silicon beams to store enough energy to jump 6.5 mm.

has managed to jump 64 cm high, which is over 30 times its body size. This jump height can be tuned by changing the amount of current passed through the shape memory alloy springs during the energy storage phase of the jump. This means of actuation is beneficial because it minimizes the number of components required to create the jumper. The shape memory alloy can be used as both a muscle that applies force, and a spring that stores mechanical energy. The drawback of this approach however is the power required to actuate the shape memory alloy springs. For the microrobot to jump, the following sequence must occur: 0.4 A of current is passed through the flexor spring for 4 seconds, 0.6 A is passed through the extensor spring for 15 seconds, and finally 0.4 A of current is passed through the trigger

spring for 2 seconds. In total, this requires over 1 W of power to be constantly supplied for 21 seconds for the microrobot to jump. This means the microrobot requires over 21 J of energy to jump 64 cm. The minimum energy this jump could require is 6.8 mJ, which means this jumping method is less than 0.1% efficient. This is not a deal breaker for a tethered setup, however if a battery or energy scavenging system were to be used, the microrobot would not be able to jump many times on a single charge. The researchers propose adding a 10 mAh battery to their device in the future, which would only allow the robot to jump 3 times at most.

Moving down in size scale, Churaman, Gerratt, and Bergbreiter from the University of Maryland created a 4x7x4 mm^3 jumping microrobot that utilized stored chemical energy to jump [7]. This 314 mg microrobot is especially exciting because it has integrated power, sensing, and control all on-board. The microrobot uses nanoporous silicon combined with an oxidizer (sodium perchlorate) for its thrust. When heat is applied to the nanoporous silicon through a bridge wire, the system ignites and the microrobot accelerates upwards. This microrobot has successfully jumped 8 cm. The circuitry consists of two capacitors, a resistor, a phototransistor, and a MOSFET. All of the aforementioned components are assembled onto a polymer chassis that is created using standard rapid prototyping techniques. To arm the robot, external wires charge the capacitors and are subsequently removed. Once the robot is exposed to light, the phototransistor turns on and sinks current from one of the capacitors through the bridge wire, which in turn ignites the oxidizer and releases the energy stored within. Upon release of this energy, the microrobot jumped 8 cm into the air. The energy required to initiate a jump is exceedingly low. It requires 150 mA at 6 V for less than 100 μs, which is a power of 0.9 W. While this power number is comparable to that of the previously mentioned microrobot, it only needs to be sourced for 100 μs. This means it requires only 90 μJ of electrical energy to initiate a jump, compared to over 21 J of electrical energy in the flea-like hopper. It was estimated that 250 μJ of energy was successfully transferred into the microrobot, and that most of the energy released by the oxidizer was not successfully transferred. The drawback is that this microrobot can only jump one time. It could theoretically jump multiple times if there were additional areas of nanoporous silicon that could be ignited individually. However, the size of the microrobot would need to scale with the number of jumps it could perform, and unless it finds a way to scavenge sodium percholrate in the wild, the total number of jumps this microrobot could ever perform would be determined at its time of fabrication.

The same researchers from the University of Maryland developed a 4x4x0.3 mm^3 microrobot that uses stored mechanical energy to jump 32 cm into the air [7]. This microrobot weighs only 8 mg and is fabricated using standard MEMS microfabrication techniques, with the addition of one non-standard microfabrication step. When developing this jumper, the researchers looked at resilin, the elastomeric protein found in fleas, as inspiration for their energy storage material. Resilin can undergo hundreds of percent strain before failing. This high fracture strain material can be desirable over a low fracture strain material, such a silicon which has a fracture strain of a fraction of up to 1%, because it allows the energy storage structure to be operated a safe margin away from where it will fracture, while not sacrific-

	Noh [6]	Churaman [7]	Churaman [7]	Greenspun
Mass [mg]	1100	314	8	43
Jump Height [cm]	64	8	32	0.6
Energy Storage Material	SMA	$NaClO_4$	PDMS	Silicon
Actuation Mechanism	SMA	Bridge Wire	Manual	Inchworm Motor
Input Power [mW]	1000	900	-	0.034
Time Between Jumps [s]	21	-	-	120

Table 1.1: Comparison of the microrobot proposed in this work with previously developed jumping microrobots.

ing too much stored energy. A biomatieral such as resilin would be exceedingly difficult to incorporate into a MEMS process, but polydimethylsiloxane (PDMS) has similar elasticity and fracture strain. The PDMS was added to a standard silicon MEMS fabrication process to create a rigid silicon backbone, a rigid silicon foot, and stretchy PDMS springs connecting the two silicon pieces. The microrobot is actuated manually using a pair of tweezers. The tweezers grip around the silicon frame and depress the foot onto a flat surface. This stores energy in the PDMS springs which accelerates the microrobot upwards when the tweezers are released. There are no motors, power, or sensing on this microrobot. Table 1.1 compares the microrobots mentioned with the microrobot proposed in this work.

1.3 Silicon Microfabrication and Simulation

Silicon microfabrication has steadily become more sophisticated and refined since it was first established in the 1960s. While these techniques were originally developed for the IC industry, the MEMS community used and built from these techniques. Today there are separate IC and MEMS foundries as well as foundries that will run combined MEMS and IC processes. The ability to design many different types of systems in a single process is one reason for the success of these two industries.

Both the Self-Destructing Silicon and jumping microrobot projects rely on the same silicon micromachining process which is run in the UC Berkeley Marvell Nanolab. This process very closely parallels the SOIMUMPS process at MEMSCAP, a MEMS foundry. It was important that the process used in this research had a logical path towards mass production and commercialization. Many of the microrobot projects up to this date have used highly specialized process that incorporate non-standard materials and processing steps. If these microrobots are ever going to be produced at high volumes, it would be advantageous to use a standard process at a foundry that has a throughput of hundreds to thousands of wafers a day. The process used here is a two mask silicon-on-insulator (SOI) process. The details of this process will be further discussed in Chapter 2.

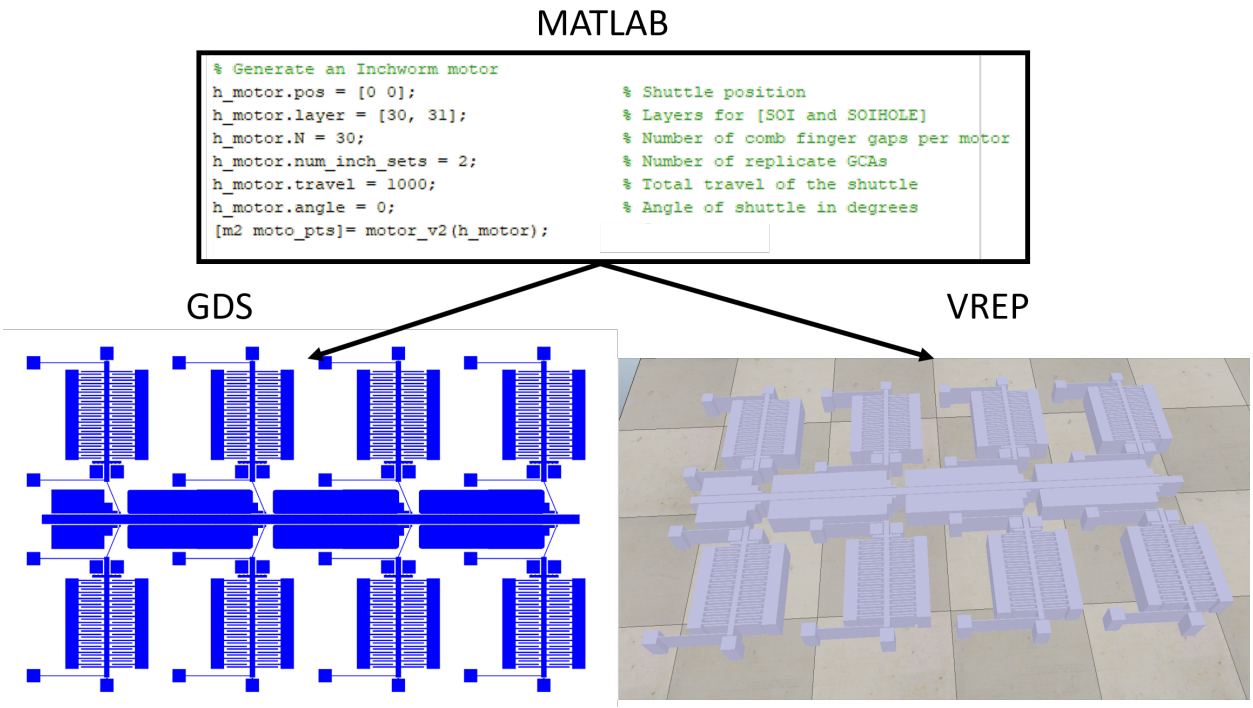

Figure 1.2: An example of high level MATLAB code being run to generate design and simulation files. Both a layout, GDS, file as well as a robotics environment simulation file, VREP, are generated behind the scenes by the MATLAB Microrobotics Library.

Although the MEMS industry adopted many of the physical etching and processing techniques from the IC industry, the simulation and design tools were not as easily transferable. It has taken a considerable amount of time for MEMS simulation and design packages to mature and be of significant utility. Even then, each field within MEMS has needed to develop individual packages to accurately model the types of devices commonly fabricated in their field. Today, there are excellent software suites for PMUTs and many kinds of resonators, but essentially no technology to design and simulate microrobots and their components.

Additionally, although the field of microrobotics has been actively studied and researched since the 1980s, progress has been slow and nonlinear. Even within the same research group, institutional knowledge seems to be lost from generation to generation. Arguably the most successful autonomous microrobot ever to be fabricated came out of the Pister group in 2002 [10]. When that group of MEMS students left, they took with them all knowledge of how those processes were run and how to design their microrobot. Barring the papers and dissertations they had published, there were no tools left to the next group of MEMS students who started in the group in 2012.

To fix both of these problems, a MATLAB library has been generated throughout the course of this work. First, it allows the current graduate students to create standard cells that

contain parameterized models of the devices and mechanisms they have developed. These standard cells take high level commands as inputs, such as how much energy something should store or how much force a motor should output, and produce the layout file (GDS) for that corresponding component. Tutorial files are easy to create and run that contain different configurations and specifications for these devices. This allows a new graduate student to see the exact design and layout for all the devices and microrobots that have been made in the group previously. It also allows for an easy transfer of knowledge to researchers at other universities. Collaboration is a fundamental building block of academia, and work should be done to make it easier for institutions to build from and expand upon the designs of others.

Additionally, this library allows for the mechanical simulation of microrobot movement directly from the layout files. When the MATLAB script is run to generate an inchworm motor, as shown in Figure 1.2, both the layout file and a simulation file are generated. This simulation file contains the information of the fabrication process and generates a scaled up version of the device in question. This file also includes the programming required to run the generated device. For example, in Figure 1.2, an inchworm motor was automatically generated. This simulation file contains the programming steps necessary to move the motor and apply a force to any component that is attached to its shuttle. This library aims to allow more design and simulation iterations before the designs are turned into masks and fabricated. This will cut down on fabrication costs as well as student time and frustrations by reducing the number of errors on masks.

Chapter 2

Self-Destructing Silicon

In 2014, the Defense Advanced Research Projects Agency (DARPA) started a program called Vanishing Programmable Resources (VaPR) to study various means of creating transient electronics, that is, electronics that can disappear in a controlled and triggerable manner. Contracts were awarded in industry and academia. The companies that had the most success used a strained glass substrate to achieve transience. A thinned IC was adhered to a heavily strained piece of glass that could be triggered using an electrical signal. This would fracture the glass and the IC along with it [11]. At Cornell, Gund et al. developed a hybrid silicon-polymer process in which individual electrical components could be detached from the overall structure by selectively burning away parts of the polymer [12]. Here at Berkeley, we used a package-level solution that utilized xenon difluoride (XeF_2) etching of silicon with the goal of completely vaporizing any silicon based IC. The remainder of this chapter details the work done to create a self-destructing silicon-based system for the DARPA VaPR project.

2.1 System Overview

To create a fully transient mote that relies solely on silicon etching, a system needed to be developed that could fulfill a variety of design constraints. First, the IC needs to be fully operational with zero external components because these external components may be unaffected by the silicon etchant. This constraint, as well as others, led to the development of the single chip micro-mote (SCμM), a wireless transceiver with an embedded Cortex microcontroller that requires no external components for operation. While not the focus of this work, SCμM is an essential piece of the self-destructing silicon project as well as the jumping microrobot described later. Secondly, the package must contain two chambers that are hermetically sealed from each other. One of these chambers will contain SCμM and the other chamber will hold the XeF_2. Only when the self-destruct signal is sent should SCμM come into contact with the XeF_2. Lastly, there must be a means of removing or breaking the barrier between these two chambers. This event should be triggered by SCμM itself. Ideally this barrier and breaking mechanism should be made of silicon as well so all components can

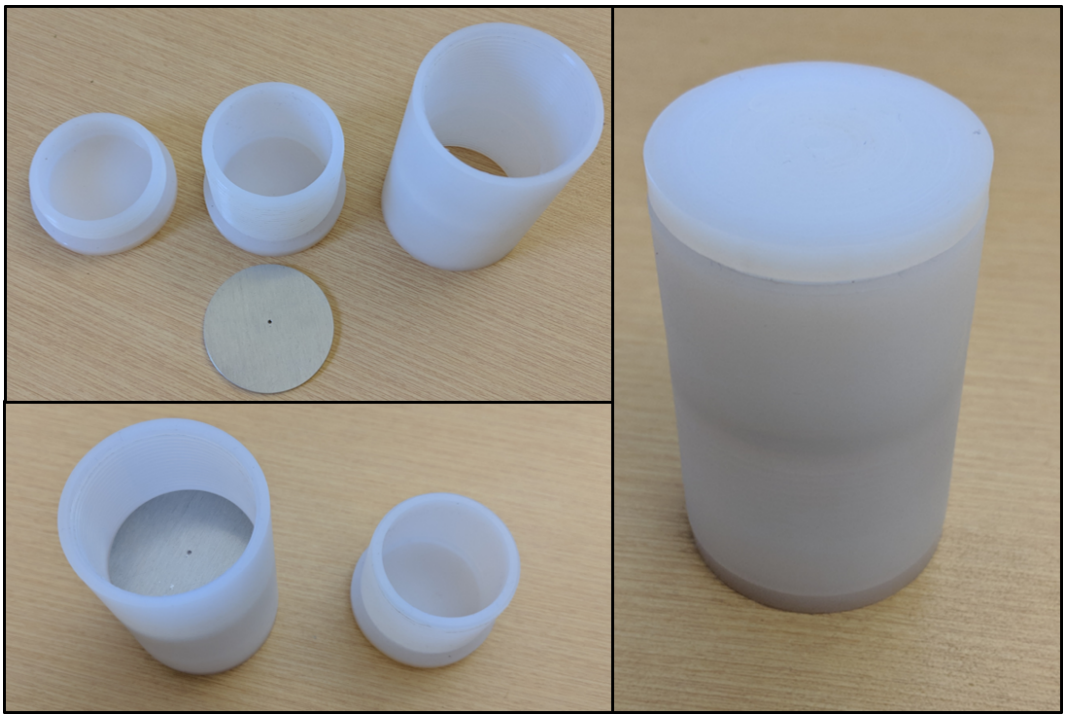

Figure 2.1: The PFA package used to house the XeF_2 and MEMS components.

be etched by the XeF_2, leaving no trace.

In developing the package, the important considerations were choice of material and mechanical design. The package must be composed of materials that are all highly resistive to XeF_2 etching. XeF_2 is a relatively unstable molecule. The fluorine atoms are highly reactive and will happily leave their xenon atom to fluorinate anything they come into contact with. For this reason the material used for the bulk of the package is perfluoroalkoxy alkane (PFA). PFA is a semi-transparent white material often used in tubing that carries highly corrosive materials. PFA is a fluoropolymer, similar to teflon, so it is chemically quite inert. This makes it a prime candidate for our package because the fluorines from the XeF_2 will not be able to attack the already fluorinated material. The bulk PFA is machined into three pieces. To assemble the package, the smaller of the two caps is screwed into the bottom of the tube. From here, the aluminum disk is placed on the inside lip of the tube. Aluminum was chosen for its near zero etch rate in XeF_2. Finally the remaining cap is screwed into the top of the tube, press sealing the aluminum disk between the cap and the lip. The package, in various stages of assembly, is shown in Figure 2.1.

After the package is assembled, there are two chambers that are separated by the aluminum disk. This disk has a 1 mm hole drilled in its center which will allow the XeF_2 to flow from one chamber to the other. During normal operation of the IC, before it destroys itself, this hole is covered by a microfabricted membrane. The membrane is impermeable to XeF_2 and keeps the two chambers completely separate. The final component of this system

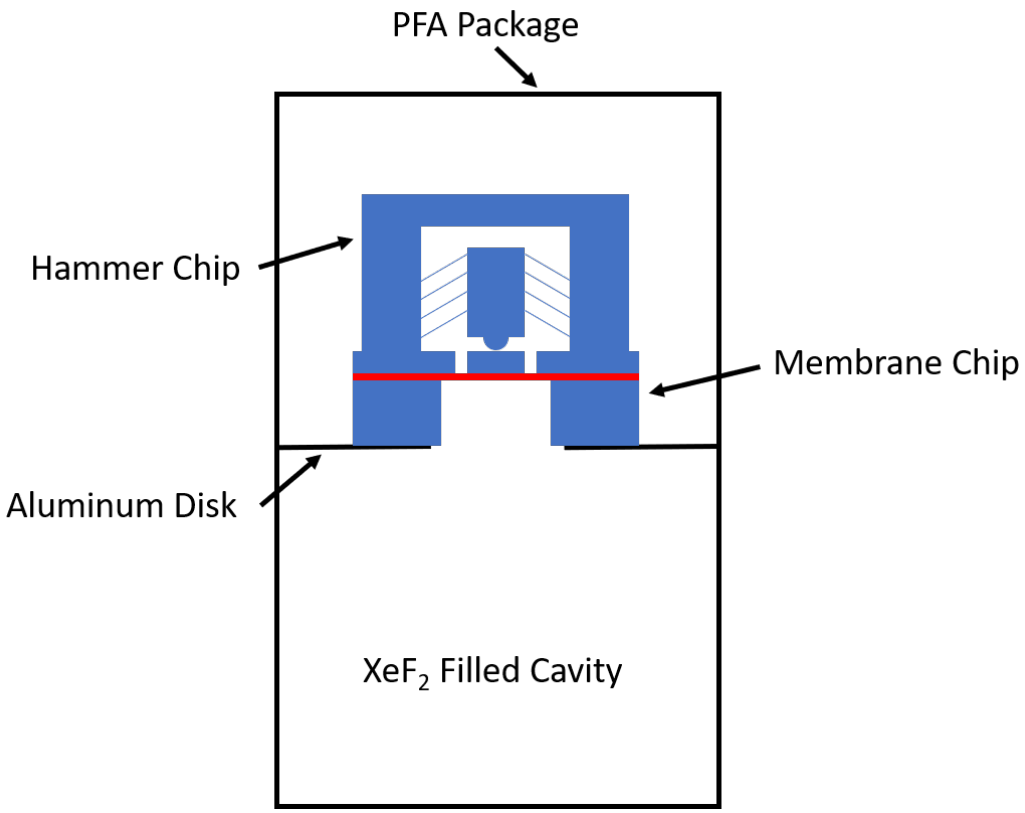

Figure 2.2: A cartoon cross section of the final package for the self-destructing silicon project. The two chambers are separated by an aluminum disk. One chamber contains the XeF_2, the other chamber contains the membrane chip with the hammer chip assembled into it.

is a MEMS device capable of storing and rapidly releasing a large amount of mechanical energy. This device, called a MEMS hammer, is capable of fracturing the membrane when a signal from SCμM is sent. This removes the barrier between the two chambers and allows the self-destruction process to begin.

2.2 Membranes and Two-Mask SOI Process

The microfabricated membranes must satisfy two opposing system constraints. They must be robust enough to survive all processing and assembly steps, yet weak enough to be reliably and repeatedly broken by the MEMS hammer. Additionally these membranes will be microfabricated from silicon and must be inert to XeF_2, a silicon etchant. So the membranes will need to be protected in some way to prevent this etching.

Membrane Design

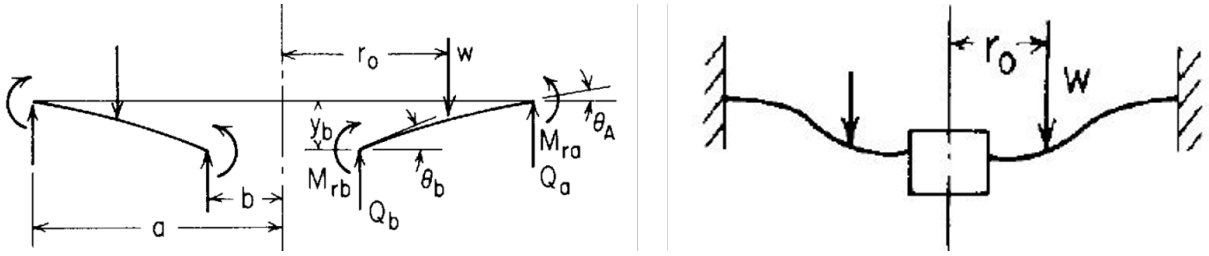

Figure 2.3: Variable definitions for a reinforced plate from Roark [13].

A membrane is one of the simplest structures that can be made from a two-mask SOI process. The membrane itself will be made of silicon dioxide with a reinforcing plate made of silicon. The analysis of this type of structure is relatively straightforward. From Roark [13], we can see that an annnular membrane has the following force-deflection and force-stress profiles, where y is the vertical deflection, a is the total membrane radius, b is the radius of the reinforcing plate, w is the applied force per unit of circumferential length, r_0 is the radius at which w is applied, ν is the Poisson's ratio of the membrane material, and t is the thickness of the membrane:

$$y = \frac{-wa^3}{D}\left(L_9 - \frac{C_2 L_6}{C_5}\right) \tag{2.1}$$

$$\sigma_{rb} = \frac{6wa}{t^2 C_5} L_6 \tag{2.2}$$

$$\sigma_{ra} = \frac{-6wa}{t^2}\left(L_9 - \frac{C_8 L_6}{C_5}\right) \tag{2.3}$$

$$\left.\begin{aligned}
C_2 &= \frac{1}{4}\left[1 - \left(\frac{b}{a}\right)^2\left(1 + 2ln\frac{a}{b}\right)\right] & L_3 &= \frac{r_0}{4a}\left\{\left[\left(\frac{r_0}{a}\right)^2 + 1\right] ln\frac{a}{r_0} + \left(\frac{r_0}{a}\right)^2 - 1\right\} \\
C_5 &= \frac{1}{2}\left[1 - \left(\frac{b}{a}\right)^2\right] & L_6 &= \frac{r_0}{4a}\left[\left(\frac{r_0}{a}\right)^2 - 1 + 2ln\frac{a}{r_0}\right] \\
C_8 &= \frac{1}{2}\left[1 + \nu + (1-\nu)\left(\frac{b}{a}\right)^2\right] & L_9 &= \frac{r_0}{a}\left\{\frac{1+\nu}{2} ln\frac{a}{r_0} + \frac{1-\nu}{4}\left[1 - \left(\frac{r_0}{a}\right)^2\right]\right\} \\
& & D &= \frac{Et^3}{12(1-\nu^2)}
\end{aligned}\right\} \tag{2.4}$$

Equation 2.1 shows the theoretical force deflection characteristic for this kind of membrane with the constants defined in 2.4. Equation 2.3 shows the theoretical stress that is

developed at the anchor points of the membrane as a function of applied load, w. These are the key equations for determining the fracture characteristics of the membranes. Equation 2.3 can be used to determine how much the membrane must deflect before it will fracture, by comparing the internal stress given by the equation with the known fracture stress of the oxide. Then, Equation 2.1 can be used to determine how much force must be applied to the membrane to produce this critical deflection and critical stress. This will be dictate how much force and deflection the actuator described in a subsequent section, the MEMS hammer, must be capable of applying.

Two-Mask SOI Process

To fabricate these membranes, a two-mask SOI process is used. The process begins with a 6-inch SOI wafer that has a 40 μm device layer, a 2 μm buried oxide layer, and a 550 μm handle wafer. Both the handle and the device silicon are doped with boron to a level of roughly 1e15. The wafer is first marked with a diamond scribe to indicate the run name and wafer number. The wafer is then cleaned in piranha for 10 minutes to remove any particulates generated by the scribing process. A 1.2 μm thick layer of Fujifilm OiR 906-12 i-line photoresist is spun on the device side of the wafer using an automatic coater system. This wafer is then exposed using a GCA 8500 wafer stepper. The wafer is developed with Dow Electronic Materials MF-26A developer for one minute. Following this step, the wafer is UV hardbaked for two minutes. Using the Bosch deep reactive ion etching (DRIE) technique, the pattern is etched into the devise layer of the SOI wafer. The photoresist is stripped using an oxygen plasma for two minutes.

Now that the device side of the wafer is etched, it must be protected so the back side of the wafer can be processed. There are many types of oxides that can deposited here, but a plasma enhanced chemical vapor deposition (PECVD) silicon dioxide is chosen for its fast deposition rate. A one micron film can be deposited in about an hour whereas that same thickness would take almost four hours in a high temperature oxide (HTO) furnace. Once the PECVD oxide is deposited, a 10 μm layer of SPR-220 i-line photoresist is spun on the backside of the wafer using a manual load photoresist spinner. This photoresist is exposed using a Karl Suss MA6 Mask Aligner and developed again using Dow Electronic Materials MF-26A developer. This resist is then hardbaked for anywhere from one hour to twelve hours at 120 C. Next, the protected device side must be bound to a handle wafer. Cool Grease is a thermal paste used to bond these two wafers. A syringe, without the needle attached, is filled with the Cool Grease and used to dispense pea sized drops of the paste. These drops are applied around the edge of the device side of the the SOI wafer as well as in the areas between dice. A handle silicon wafer is laid upon the device side, and pressure is applied on a hot plate at 50 C for 30 minutes to bond the two wafers. The same DRIE process is used to etch through the handle wafer. This etch step serves two purposes, it can pattern useful features into the back side as well as singulate individual chips from the wafer. The DRIE is effectively used to dice the chiplets out of the wafer, without needing to use a dicing tool.

Once this back side etch is complete, the photoresist is stripped in an oxygen plasma for 30 minutes, and the chiplets are plucked from the wafer manually using tweezers.

At this point the membrane chips are done being fabricated. For these devices to operate as membranes, the oxide must remain to act as the membrane. However, for other devices in the process, this buried oxide must be selectively undercut and removed. One of the benefits of using the backside etch to dice the chiplets is that some chiplets can continue processing while others are taken out of the processing line. For the devices that need the oxide undercut and recessed, the chiplets are placed on a carrier wafer and run through a vapor HF tool. This tool uses vapor HF to selectively etch a certain amount oxide underneath the silicon. While liquid HF could be used here, because there is no metalization, the vapor HF process is preferred to mitigate stiction of SOI features to the substrate. There are critical point drying tools in the lab that could be used in conjunction with a wet HF release, however minimizing the number of tools in the process flow is always preferred. The amount of undercut can vary slightly from run to run, but is typically around 6 μm. This allows some silicon features to be free from the substrate and mobile, while other features remain mechanically bound to the substrate. Now the wafers have been fully fabricated using the basic two-mask SOI process. However, the yield on most of the membranes was exceedingly low, so they had to be redesigned and processed differently, as discussed below.

Membrane Redesign and Process Modifications

When the membrane chiplets came out of the last step of the process, they would often be fractured as seen in Figure 2.4. This makes the membranes completely unusable as barriers to XeF_2. The membranes also sometimes fractured while being plucked from the SOI wafer after the back side etch. Either way, the membranes were too weak to survive basic handling, and modifications needed to be made.

Material	Coefficient of Thermal Expansion $[\frac{10^{-6}}{K}]$
Silicon	2.6
Silicon Dioxide	0.5
Silicon Nitride	3.3

Table 2.1: Comparison of the coefficients of thermal expansion for common MEMS materials.

One explanation for the premature fracturing of the oxide is the thermal stress inherent to the buried oxide layer. There are many methods to choose from when fabricating an SOI wafer. The wafers used here are fabricated from two silicon wafers that are oxidized and bonded together at the oxide interface. Finally the silicon from one of the wafers is ground down to the desired thickness. During the oxide growth step, the ambient temperature is over 1000 C and the oxide grows stress free. However, when the wafer is removed from the furnace and cools back down to room temperature, the oxide and silicon contract at different

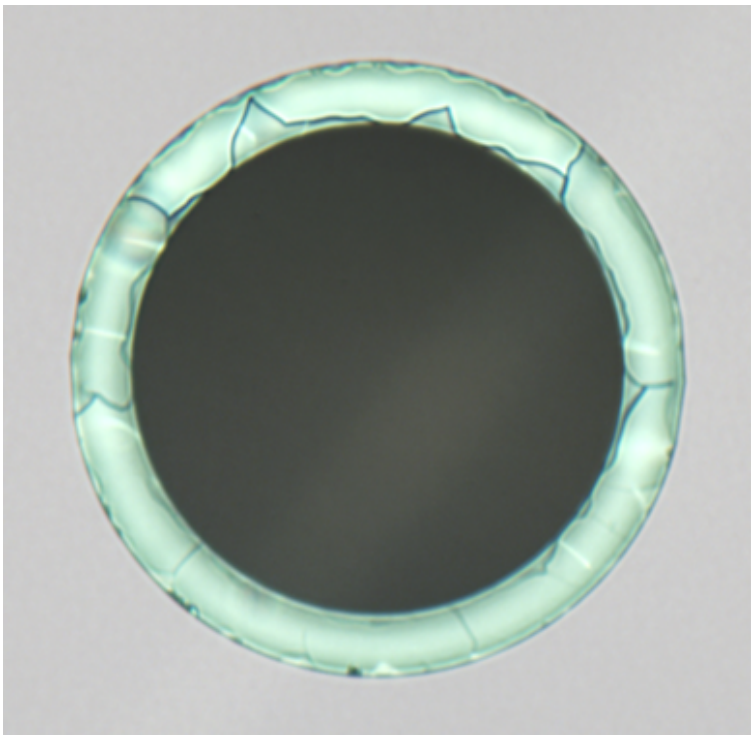

Figure 2.4: Image of a membrane with no process modifications directly after the last DRIE step. This membrane is 1 mm in diameter.

rates. This rate is dictated by the coefficient of thermal expansion of each material, and can be seen in the top two rows of Table 2.1. The silicon has a much larger coefficient than the oxide. This means that when the two materials cool down, the silicon will contract more than the oxide. The silicon is much thicker than the oxide so the oxide is forced to contract as much as the silicon. This leaves the oxide with a large degree of internal compressive stress. As long as the silicon remains bonded to the oxide, the oxide will remain stressed. However, once the device silicon above the oxide and the handle silicon below the oxide are etched away, there are no longer any external forces maintaining the stress in the oxide. When this happens, the oxide relaxes and due to the nature of the membrane structure that means the membrane will buckle. It is during this buckling that the oxide typically fractures. In Figure 2.4, the silicon disk is a far darker color than the surrounding silicon. This is because the membrane has buckled and this disk is no longer in a the same plane as the rest of the wafer. It is common to see features that have come out of plane showing up dark or black in microscope images.

To remedy this problem a new material was added to the process. This material was chosen for its thermal and internal stress properties, such that it would could offset the buckling nature of the oxide. If a material is added to one side of the membrane and this material has a higher coefficient of thermal expansion than the oxide, the oxide will be

restricted from buckling due to the tensile stress in this new material. Stoichiometric silicon nitride, Si_3N_4, has a higher coefficient of thermal expansion than either silicon or oxide, as is shown in Table 2.1. Additionally, this film is deposited with high levels of intrinsic tensile strain. This strain comes from the film shrinkage due to dissociation of Si-H and N-H bonds and the rearrangement of the dangling bonds to more stable Si-N bonds [14]. This material is also easy to integrate into the process. After the front side DRIE is performed, the wafer is cleaned and placed into a low pressure chemical vapor deposition (LPCVD) furnace capable of depositing nitride. 500 nm of nitride is deposited on both sides of the wafer, and selectively removed from the back side of the wafer using a nitride reactive ion etching (RIE) tool. From here the process continues normally. This additional layer kept the membranes from fracturing during processing and brought the yield to greater than 95%, up from less than 10%.

Now that the yield is sufficiently high, one additional processing step was performed to protect the backside of the membranes from being attacked by the XeF_2. As stated previously, as long as the membrane is intact, it must be a barrier to XeF_2, however the backside of the membrane chip, the handle wafer, is made of silicon which is readily etched by XeF_2. There is a thin fluoropolymer left behind by the DRIE process, however it is not sufficient to protect the silicon from the etchant. Alumina, Al_2O_3, forms an excellent barrier to the etchant and can easily be deposited onto a chip either by sputtering or atomic layer deposition. A 100 μm thick layer of alumina is deposited on the back side of the membrane chips using atomic layer deposition. Now the membranes form a solid physical barrier and are chemically inert to XeF_2 etching.

Membrane Testing

After the addition of the nitride and alumina layers into the process, the membranes could be tested for their force deflection profiles. The testing setup consisted of a load cell, a micron resolution optical stage, and a tungsten probe tip. During testing, the probe tip was fixed to the optical stage and used to push on the membrane which rested on the load cell. The stage was moved in increments of 2 μm, and the value on the load cell was recorded. The stiffness of the load cell itself was also measured and calibrated out of the membrane measurements.

Figure 2.5 shows the measured membrane data that was acquired using the above-mentioned testing setup with the labels explained in Table 2.2. In this table, the 'Support' column refers to the existence of a 10 μm wide beam in SOI that spans the full width of the annulus and connects the silicon reinforcing plate to the surrounding silicon area. The error bars in the figure represent one standard deviation above and below the mean force and deflection fracture values. Theoretically, the larger membranes had a larger spread of force and deflection required to break them. For this reason, the smaller membranes were used in the final prototype self-destruction system. This plot gives a starting point for designing the energy storage and rapid release device that will be tasked with breaking this membrane.

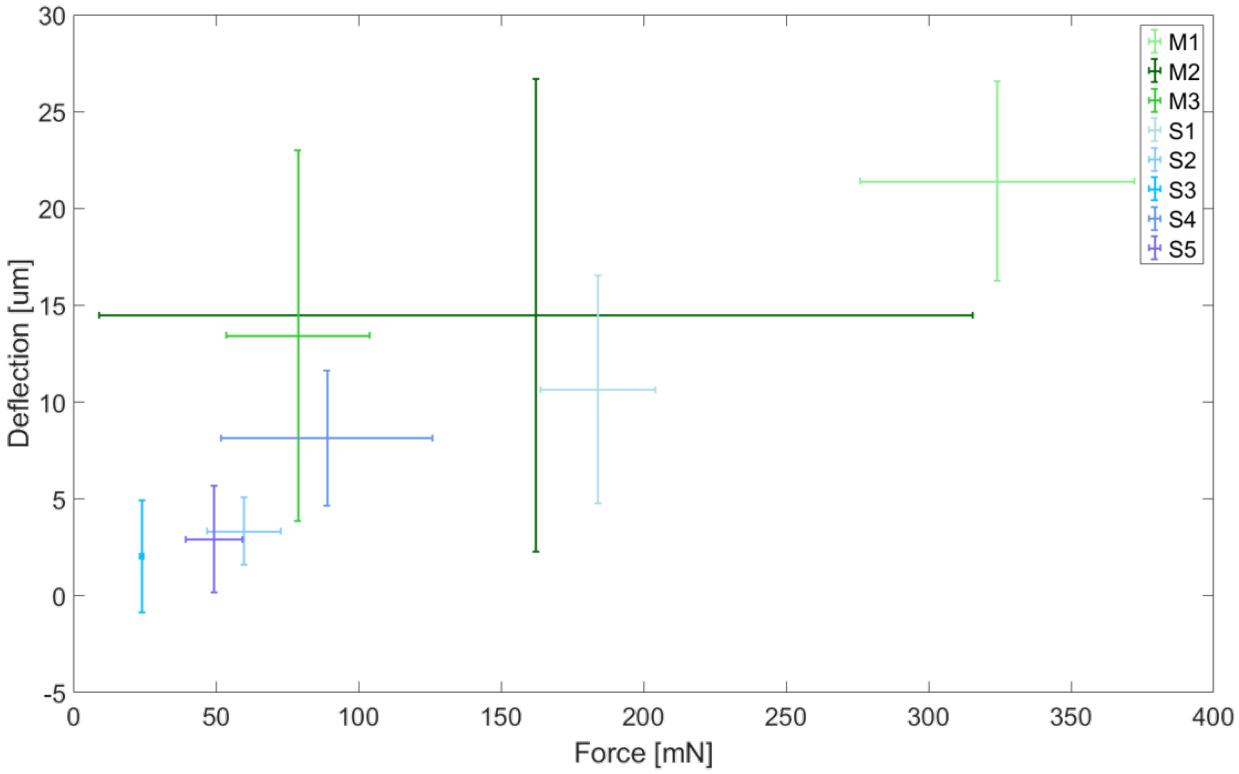

Figure 2.5: Plot showing the critical force and deflection values that cause membranes of various size to fracture. The error bars show one standard deviation above and below the mean.

So long as this device can apply the forces and deflections shown in this plot, it should be able to fracture the membrane and begin the self destruction process.

2.3 MEMS Hammer

Now that the membranes have been designed and fabricated, a device must be engineered that can fracture them. Because the device will be used to smash and break these membranes, it was named the MEMS hammer. There are a handful of constraints that this device must satisfy. Firstly, it must be able to deliver the force and displacement required to fracture the membrane it is paired with. The MEMS hammer must also have some form of electrical release if the self-destruction process is to be triggered by a sensor node. On that same note, the signal required to release the MEMS hammer should be sufficiently small in magnitude such that a standard microcontroller could source it. Additionally, an important consideration is that this device only needs to work one time given the intended total self-destruction application. This will allow for a much simpler design that can be preloaded at

Membrane Label	Diameter [μm]	Annulus Width [μm]	Support
S1	1000	10	No
S2	1000	25	No
S3	1000	50	No
S4	1000	25	Yes
S5	1000	50	Yes
M1	2000	10	No
M2	2000	50	No
M3	2000	100	No

Table 2.2: Table showing geometric parameters for various membrane designs.

the time of fabrication to simply store energy rather than requiring the integration of motors to actively store this energy. And finally, it would be beneficial to the development time line if this device used the same fabrication process developed for the membranes.

Mechanical Energy Storage

The primary goal of the MEMS hammer is to store a certain amount of mechanical energy and subsequently release that energy at the command of a microcontroller. Mechanical energy can be stored in a material by bending the material, stretching it, or some combination of the two. Given a geometry of a particular material, different amounts of energy can be stored depending on energy storage mode chosen.

Imagine a cantilever fixed at one end and free at the other with a length L, a width w, a thickness t, and a Young's modulus E. A force, F, is applied in the direction of the thickness of the beam, at the free end, and the the cantilever bends with the tip deflecting a distance Δx. The following equations hold true:

$$F = \frac{Ewt^3}{4L^3}\Delta x \tag{2.5}$$

$$U = \frac{Ewt^3}{8L^3}\Delta x^2 \tag{2.6}$$

$$\epsilon_{max} = \frac{3t}{2L^2}\Delta x \tag{2.7}$$

By combining the above equations, the maximum energy that can be stored in this cantilever before it fractures is given by the following equation:

$$U_{max} = \frac{ELwt}{18}\epsilon_{max}^2 \tag{2.8}$$

Now imagine the same cantilever where the force is applied axially in the direction of the length of the beam, again at the free end. Instead of bending the cantilever will stretch along its axis. The following equations describe this cantilever:

$$F = \frac{Ewt}{L}\Delta x \tag{2.9}$$

$$U = \frac{Ewt}{2L}\Delta x^2 \tag{2.10}$$

$$\epsilon = \frac{\Delta x}{L} \tag{2.11}$$

Again, combining these equations results in the maximum energy that can be stored axially in this cantilever before it fractures:

$$U_{max} = \frac{ELwt}{2}\epsilon_{max}^2 \tag{2.12}$$

By comparing equations 2.8 and 2.12 it is clear that stretching a material leads to the potential for storing 9 times more energy in the same volume of material. This intuitively makes sense because when bending a beam, the strain goes to zero at the neutral axis and therefore this volume of material does not store any energy. Whereas when axially stretching a beam, the entire volume of material is strained uniformly, and therefore the entire volume of material contributes to the stored energy.

Material	E [Pa]	Max Strain [%]	Energy Density [$\frac{mJ}{mm^3}$]
Silicon	$1.69\text{x}10^{11}$	0.5 - 1.0	2.1 - 8.5
PDMS	$7.5\text{x}10^{5}$	293	3.4
Polyurethane	$7.6\text{x}10^{6}$	500	95
Resilin	$2.0\text{x}10^{6}$	190	4

Table 2.3: Mechanical properties of energy storage materials [15], [16], [17].

The two material properties in the above equations can be used to help determine which materials will be best suited for this energy storage. Both Young's Modulus (E) and the maximum strain (ϵ_{max}) can vary over many orders of magnitude in different materials. Table 2.3 shows the material properties for some energy storage material candidates as well as resilin, the energy storage material used by insects, for comparison. Ideally, the material would have a high maximum strain, so it could be operated a safe distance away from its fracture point. Additionally, the material should be able to store large amounts of energy per unit volume. This will minimize the total amount of material that needs to be strained to achieve the required stored energy. One more important consideration is the ease of incorporating

this energy storage material into a fabrication process. While it is possible to assemble energy storage components, it is easier and faster if the energy storage elements are fabricated in place without the need for assembly. For this reason, polyurethane will not be used for the energy storage material in the MEMS hammer because it would be exceedingly difficult to incorporate into the process. Researches at the University of Maryland have successfully incorporated PDMS into a standard SOI process [7]. The process, while relatively straightforward, does increase the complexity from the two-mask SOI process previously discussed and there are some issues with the PDMS adhering to the silicon. While PDMS may be a more desirable material in some instances, the added processing steps combined with the potential for silicon to meet or exceed its energy density, were the deciding factors in using silicon as the energy storage material for the MEMS hammer.

Now the the same two-mask SOI process will be used for the MEMS hammer, a more thorough analysis can be done on the beams that will store the energy required to break the membranes. Table 2.3 shows that there is a range in the maximum strain, and subsequently energy storage capability, or silicon. In fact, some places in the literature claim silicon can be strained up to 6% before fracturing [17]! This variance in fracture strain shows that this limit is sensitive to the processing and preparation of the silicon device. Processing steps to increase this limit will be discussed in Section 2.4. From the previous section, the displacement and force requirements of the MEMS hammer are known to be on the order of 10s of μm and 100s of mN. In order to make the most energy dense structures, axial stretching should be used. So, if this device must deflect 40 μm, that means that the beams that compose the MEMS hammer would need to be 4 to 8 mm long, and oriented with their long axes perpendicular to the plane of the membrane. While this is not a non-starter, it is certainly not ideal considering there will be other overhead on the MEMS hammer chip to support other functionality. To remedy this, a different approach is taken that uses both bending and stretching to store energy into these silicon beams.

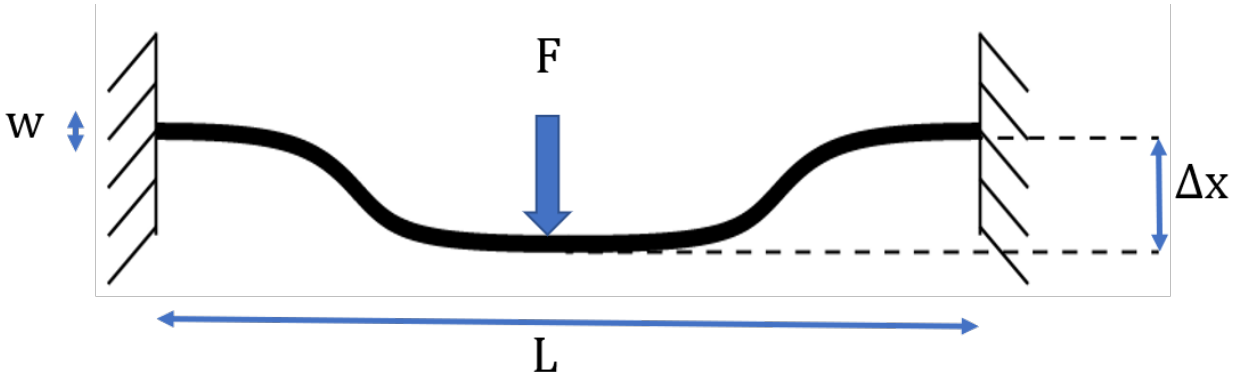

Figure 2.6: Centrally loaded fixed-fixed beam.

To achieve this, a model was first developed to predict the force deflection characteristics of these beams. The diagram in Figure 2.6 shows the layout of a beam with an applied

force. These beams are centrally loaded fixed-fixed beams, which have a force-deflection profile given by the following equation, when the force is applied in the direction of the beam width:

$$F = 16\frac{Ew^3t}{L^3}\Delta x \tag{2.13}$$

This equation accounts only for the bending strain energy stored in the beam. However, as this beam is deflected further, and the anchors keep the ends of the beam fixed, axial strain is introduced. This adds a third order term to the equation. A simple derivation of this non-linear term is done by assuming the spring stretches linearly, like a rubber band. By ignoring the angle boundary conditions at the fixed and guided edges, the derivation becomes a simple geometry problem. The non-linear term turns out to be:

$$F = 8\frac{Ewt}{L^3}\Delta x^3 \tag{2.14}$$

Combining Equations 2.14 and 2.13, the following equation is obtained which approximates the force-displacement relation of a large deflection fixed-fixed beam:

$$F = N\left[16\frac{Ew^3t}{L^3}\Delta x + 8\frac{Ewt}{L^3}\Delta x^3\right] \tag{2.15}$$

Taking these geometric simplifications into account the resulting maximum strain that develops in the beam can be given by the following:

$$\epsilon_{max} = \frac{12w}{L^2}\Delta x + \frac{2}{L^2}\Delta x^2 \tag{2.16}$$

Equations 2.15 and 2.16 show the force-deflection and strain-deflection profiles for N beams with Young's Modulus E, width w, thickness t, and length L.

To test the accuracy of this equation, these springs were designed and fabricated in the two-mask SOI process. The force-deflection profile was measured with a Dage 4000 bond tester. This tool is capable of sampling and recording the force-deflection profile of various MEMS and macro scale structures. Using this tool the profiles of these beams were measured and compared with the model. Figure 2.7 shows that there is excellent agreement between the theory and the measured force-deflection relationship. At any deflection, the measured force deviates from the theoretical force by less than 5%. The energy storage is now fully characterized and can be integrated into the final MEMS hammer design. The remainder of the design focuses on delivering the stored energy to the membrane as well as developing a way to latch the hammer in place.

Figure 2.8 shows the initial design of the MEMS hammer. The energy storing beams are the same centrally loaded fixed-fixed beams we have been analyzing up to this point. They are connected to the hammer itself and anchored on either side. The cylindrical impactor is the component that will eventually make contact with the membrane to fracture it. The rest of the structure helps support normal operation of the hammer. In the following subsection the latching design will be explained and analyzed.

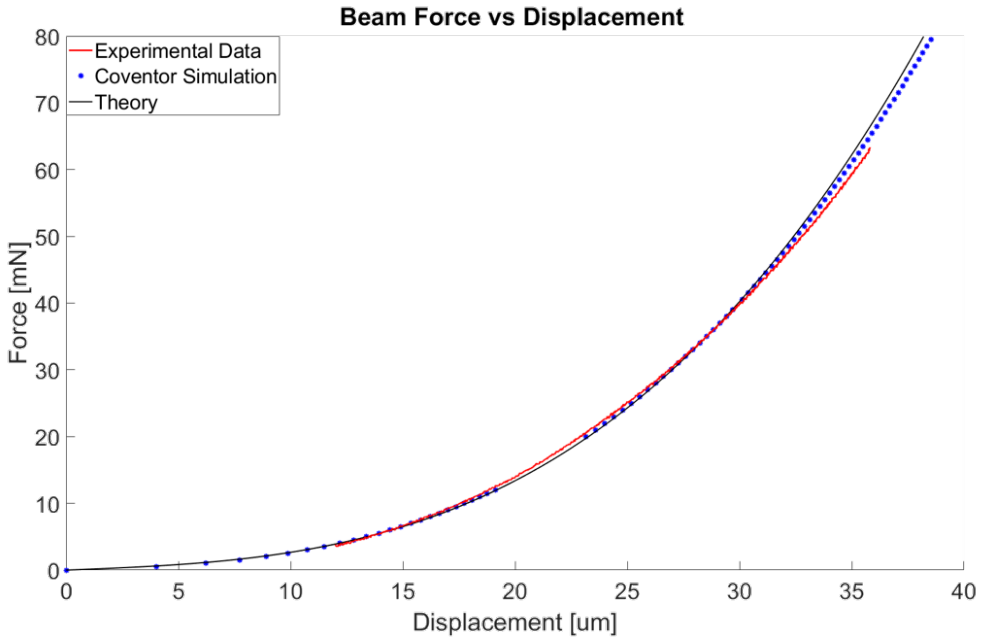

Figure 2.7: Plot showing theory, simulation, and data all agreeing to within 5%.

Mechanical Latching

Initially, the hammers were latched into place mechanically. This was the simplest and most straightforward means of keeping the energy storing beams in their high energy state. Figure 2.9 shows the MEMS hammer in its initial unloaded position. From here, the left probe tip is used to push the lever arm towards the right. As this happens the lever arm rotates about the pin and catches onto the hammer itself. As the lever arm continues to be pushed towards the right, the hammer is pulled back and the energy storing beams are loaded. Once the lever arm is next to the mechanical latch, the right probe tip is used to pull the latch about 5 μm down to allow the lever arm to pass by. After this happens, the lever arm is pushed all the way to the right, and the latch is allowed to return to its initial position, locking the lever arm, and thus the hammer, in place. Once it is time for the hammer to release its stored energy, the mechanical latch is pulled back, again with a probe tip, and both the latch and the hammer are accelerated forwards. This latching technique, while easy to implement, has many drawbacks. The biggest one being that the release requires a mechanical input that must occur at a probe station. To remedy this, an electrostatic latch was developed and implemented.

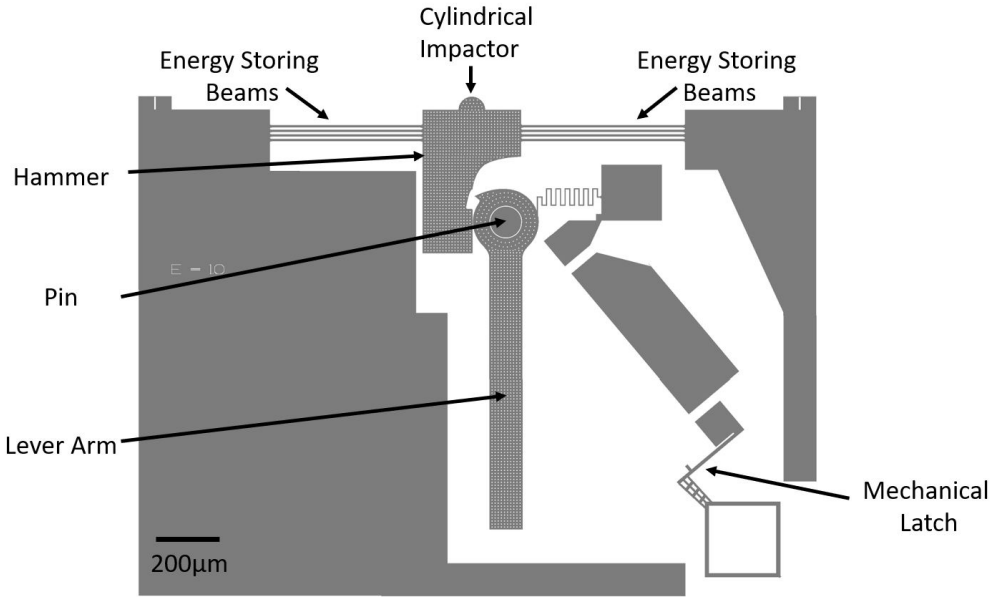

Figure 2.8: Initial design of MEMS hammer with salient parts labeled.

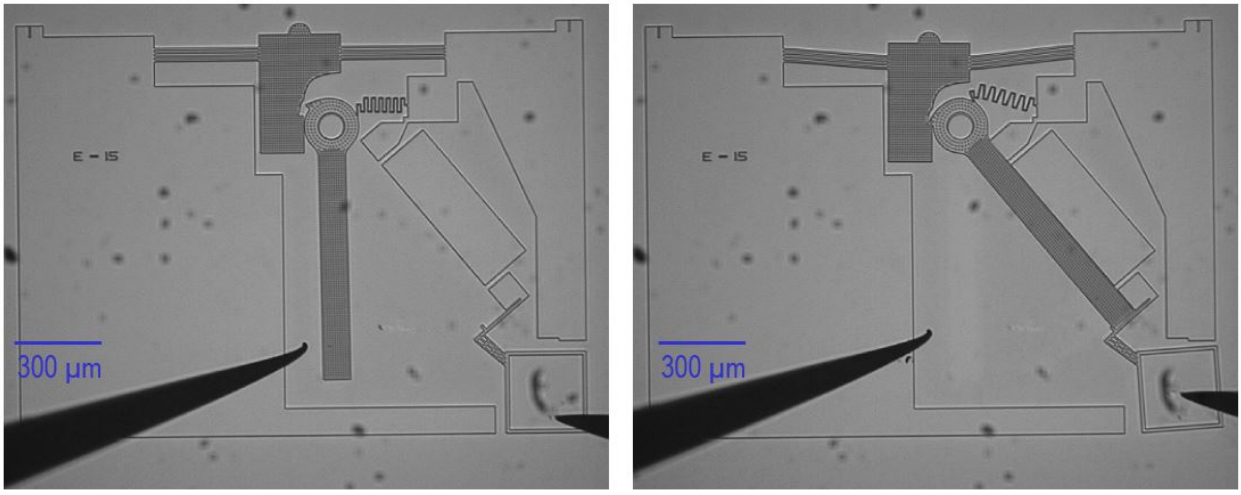

Figure 2.9: MEMS hammer in its unlatched (left) and latched (right) states.

2.4 Electrostatic Latches

The electrostatic force is the workhorse of this two-mask SOI process. One of the most compelling reasons for using this process is that the integration of sensors, actuators, and mechanisms can be trivial. In this case, the electrostatic latch can easily be patterned and fabricated right next to the lever arm. The goal of this component is to maintain the hammer in its high energy state while some voltage is applied, and for the hammer to release its energy

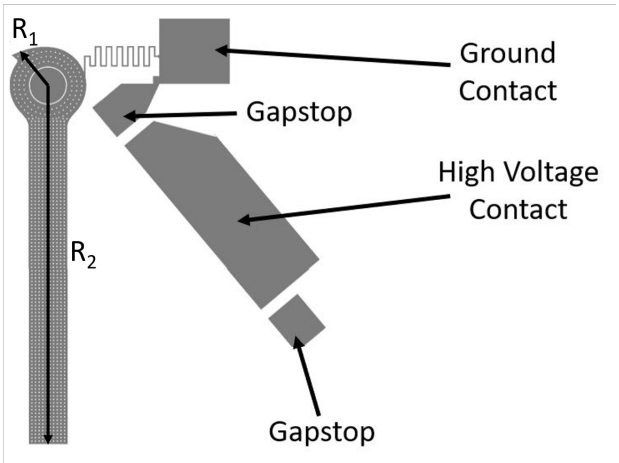

Figure 2.10: Layout of an electrostatic latch.

when this voltage is removed. To achieve this, a voltage must be applied across the lever arm and some structure anchored to the substrate. There also must be some built in gap stop so that the two structures do not pull in and touch each other. The basic layout is shown in Figure 2.10. With no mechanical latch, the lever arm is again pushed towards the right. As it moves, it will eventually make contact with both of the gapstops. These gapstops protrude from the high voltage contact such that after the lever arm is in contact with the gap stops, there is a narrow electrostatic gap between the lever arm and this high voltage contact. When a voltage is applied across the ground contact and the high voltage contact, the serpentine spring grounds the lever arm and there will be an electrostatic force pulling the lever arm towards the high voltage contact given by the following equation, where ϵ is the permittivity of the material in the gap, V is the voltage applied, A is the overlap area, and d is the gap between the surfaces:

$$F_{ES} = \frac{1}{2}\epsilon V^2 \frac{A}{d^2} \tag{2.17}$$

Using this equation, an expression can be derived to determines the minimum requirements for the latch. As described previously, the lever arm rotates around a central pin. This pin acts as a fulcrum allowing forces applied on one side to counteract forces applied to the other side. When the lever arm contacts the hammer, the distance from the contact point to the center of the pin (R1) is much smaller than the distance from the end of the lever arm to the center of the pin (R2). Therefore a smaller electrostatic force can theoretically be used to balance a much larger force from the hammer. By balance the moments induced by these forces, the following equation must hold true for this system to be in static equilibrium:

$$F_{Hammer} = F_{ES}\frac{1}{2}\frac{R_2}{R_1} \tag{2.18}$$

The factor of $\frac{1}{2}$ comes from the fact that there is a distributed force along the entire length of the lever arm. This distributed force can be modeled as a point force at the geometric center of the distributed load, $\frac{R_2}{2}$. Equation 2.18 shows that the weak electrostatic force is effectively amplified by the ratio of R_2 to R_1. However, this is not quite the full picture due to the serpentine spring that attaches the lever arm to the ground contact. This spring adds an additional moment in the same direction as the hammer. This gives the following equation which fully describes the static behavior of this system:

$$F_{Hammer}R_1 + F_{SS}R_{SS} = \frac{1}{2}F_{ES}R_2 \tag{2.19}$$

The radii R_1 and R_2 must be chosen appropriately to ensure that this latch will be operational. While mathematically R_1 should be as small as possible to amplify the electrostatic force maximally, there are other factors that limit how small it can be. The pin shown in Figure 2.8 is the anchor around which the lever arm rotates, and therefore must be strong enough to supply the reaction forces necessary for that rotation. Typically in a two-mask SOI process, if an anchored feature survives the processing steps, it will remain anchored during standard operation as well. However, with these MEMS hammers, hundreds of mN are being applied so special care must be taken to ensure that the anchors are strong enough to remain fixed to the handle wafer. The fracture stress of silicon dioxide varies in the literature from 0.77 MPa [18] to 364 MPa [19]. Due to this large variation in fracture stress, experiments were performed with the Dage bond tester to determine the fracture stress of this particular oxide. Anchors of various sizes were fabricated and tested tested to failure with the Dage bond tester. It was determined that a pin of radius 46 μm can withstand forces up to 300 mN in shear, putting the fracture stress of this oxide at 54 MPa. This value sets the minimum radius of the pin and thus the minimum for length of R_1, 120 μm. This pin can withstand forces up to 250 mN.

Now that R_1 has been successfully minimized, an appropriate length for R_2 is chosen such that the system will remain in static equilibrium. Unfortunately, the electrostatic force is so small that for this force balance to work, R_2 would need to be exceedingly long. To roughly estimate the electrostatic force, limits can be places on the free variables in Equation 2.17. The minimum gap that can be reliably achieved in this process is defined as the offset between the gapstops and the high voltage contact. While this offset is technically defined in layout, the fabrication process will have an effect on the repeatability of this offset. Taking all of that into consideration the smallest reliable gap that could be patterned and etched was 0.5 μm. This sets the minimum on the parameter d in Equation 2.17. The voltage used on the latch is technically only limited by the electrical breakdown field in air. This value is roughly 300 V, as this size scale. However, if the system is going to use an off the shelf microcontroller to supply a voltage, this value will need to be on the order of 5 V. Lastly, A, the overlap area, will be the product of the thickness of the SOI layer, 40 μm, and the length of the high voltage contact. Combining all of these constraints, if the hammer applies 200 mN of force at a distance of 120 μm, for an electrostatic latch with a driving voltage of 5 V to balance it, the lever arm would be over 5 cm long. This far exceeds the size of a

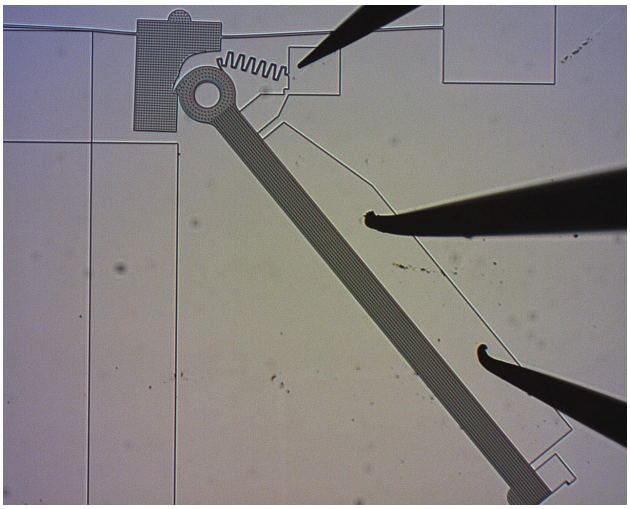

Figure 2.11: A proof of concept electrostatic latch. This 2 mm long lever arm required 160 V to restrain the attached MEMS hammer.

single die and furthermore it would be effectively impossible to make a lever arm stiff enough that it would not bend by more than 0.5μm over those 5 cm. However, to prove the concept could work, a weaker hammer was fabricated with a long lever arm and the limitation on the voltage was removed. The device, shown in Figure 2.11, successfully latched while the 160 V signal was applied. When the voltage was removed the lever arm swung open and the hammer accelerated forwards.

From here, the goal was to decrease the operating voltage while increasing the maximum output force of the hammer. As previously stated, simply increasing the length of the lever arm was not a viable option for achieving this, so a new topology was designed. This design contains one or more purely mechanical lever arms that are arrayed before an electrostatically latched lever arm, like we saw previously. Each of these mechanical stages amplifies the force from the electrostatic latch until it is large enough to counteract the large force from the hammer. One of these devices can be seen in Figure 2.12. This system can be described by the following equation where N is the total number of stages:

$$F_{Hammer}R_1 = -F_{SS}R_{SS}\sum_{i=0}^{N-1}\left[\frac{R_2}{R_1}\right]^i + F_{ES}\frac{R2}{2}\left[\frac{R2}{R1}\right]^{N-1} \tag{2.20}$$

This equation shows the exponential relationship between the number of stages and the amplification factor of the electrostatic force. The impact of the serpentine spring, however is also amplified by a similar factor. It is important to decrease the spring constant of this spring, and thus F_{SS}, by as much as possible to create the most voltage efficient latch.

Devices were fabricated that had between 2 and 8 latch stages. Each of these devices was tested to determine the minimum latching voltage. To test these devices, the lever arms of the mechanical stages were sequentially engaged with a probe at a probe station. The

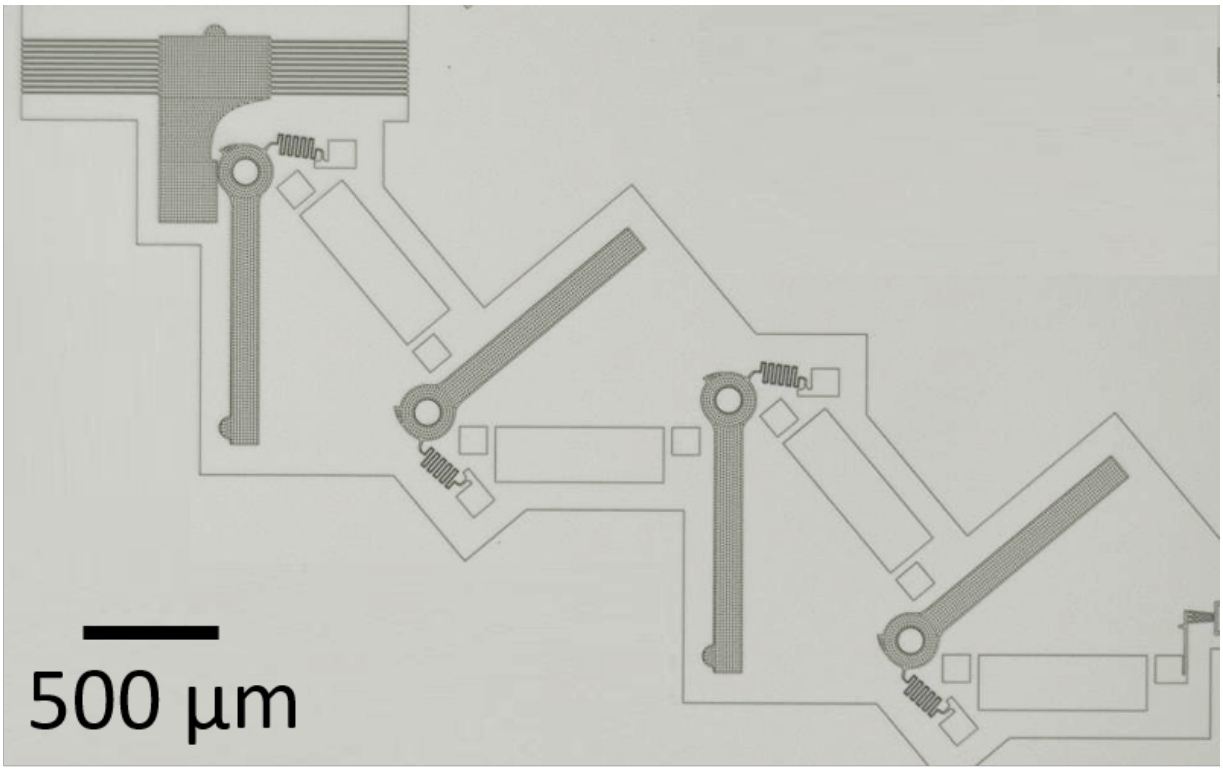

Figure 2.12: A 4 stage electrostatic latch.

final electrically active latch is then engaged and a sufficiently large voltage is applied across the electrostatic latch contacts such that the latch remains closed. This voltage is slowly turned down until the latch no longer stays shut. The lowest voltage value at which the latch remains closed is recorded as the latching voltage. Results for this test are shown in Figure 2.15 with the label 'Air Latch'. This data, while generally agreeing with the theory, poses a few problems. First, the latching voltage, even for an 8-stage device, is too high to be compatible with a standard microcontroller. Additionally, the variability from device to device is extremely high. The error bars on the plot, representing one standard deviation from the mean are roughly 50 V. Lastly, the flattening of the voltage with increasing number of stages is a function of the serpentine spring geometry. This line approaches a value that balances the electrostatic force with force from the serpentine spring. Regardless of how many stages are added, the electrostatic force must match this force from the serpentine spring. Currently, it is possible to trigger the release of a MEMS hammer electrically, however the operating voltage required to do so is higher than desired. To fix this, modifications to the two-mask SOI process were introduced.

Low-Voltage Electrostatic Latches

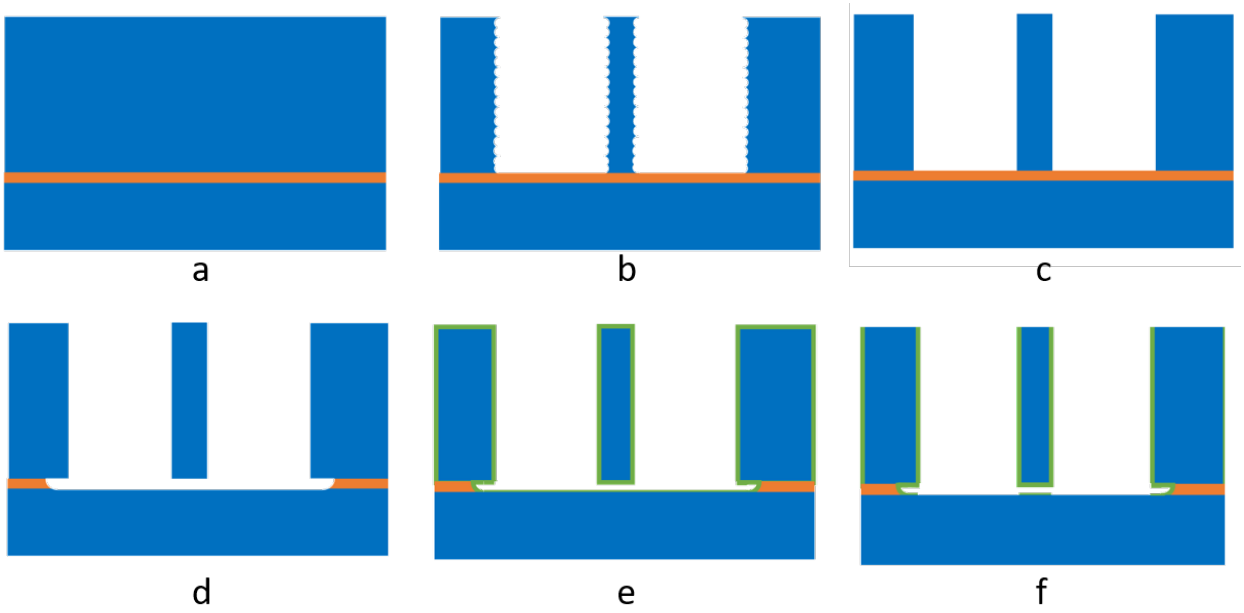

Figure 2.13: Process steps for low-voltage latches. a) Start with SOI wafer b) Device side DRIE c) H_2 anneal at 1100 C d) Vapor HF release e) Al_2O_3 ALD f) Al_2O_3 RIE

To decrease the operating voltage of these latches, the electrostatic force needs to be increased. To do this we can look to Equation 2.17 for guidance. There are a few options to generate the same or higher force at a lower operating voltage. Firstly, the the gap, d, could be decreased. This is an attractive option due to the square relationship of this parameter and the electrostatic force. Decreasing this gap will have a large effect on the performance. Additionally, the parameter ϵ can be changed. This is linearly related to the electrostatic force, and typically is set at the vacuum permittivity, ϵ_0. This is because typically electrostatic gaps are used as actuators or sensors in which the gap needs to change size. The only way for this to happen is if there is a fluid or a vacuum between the two plates that define this gap. Typically this fluid is air, but water has also been used [20]. In this case however, the gap does not need to change sizes during operation, so it could be filled with any material, including a solid. This realization is the key to creating latches that operate at far lower voltages than previously demonstrated.

If the electrostatic gap is filled with a solid material, both variables d and ϵ in Equation 2.17 can be moved in a favorable direction. Previously the electrostatic gap distance, d, was limited by the photomask registration as well as the DRIE process resulting in a minimum gap of 0.5 μm. However, if a conformal deposition process is used to coat the sidewalls with a material, this gap is now defined by the thickness of material deposited. This thickness can easily be controlled down to 10s of nm, even in an academic cleanroom setting. Decreasing

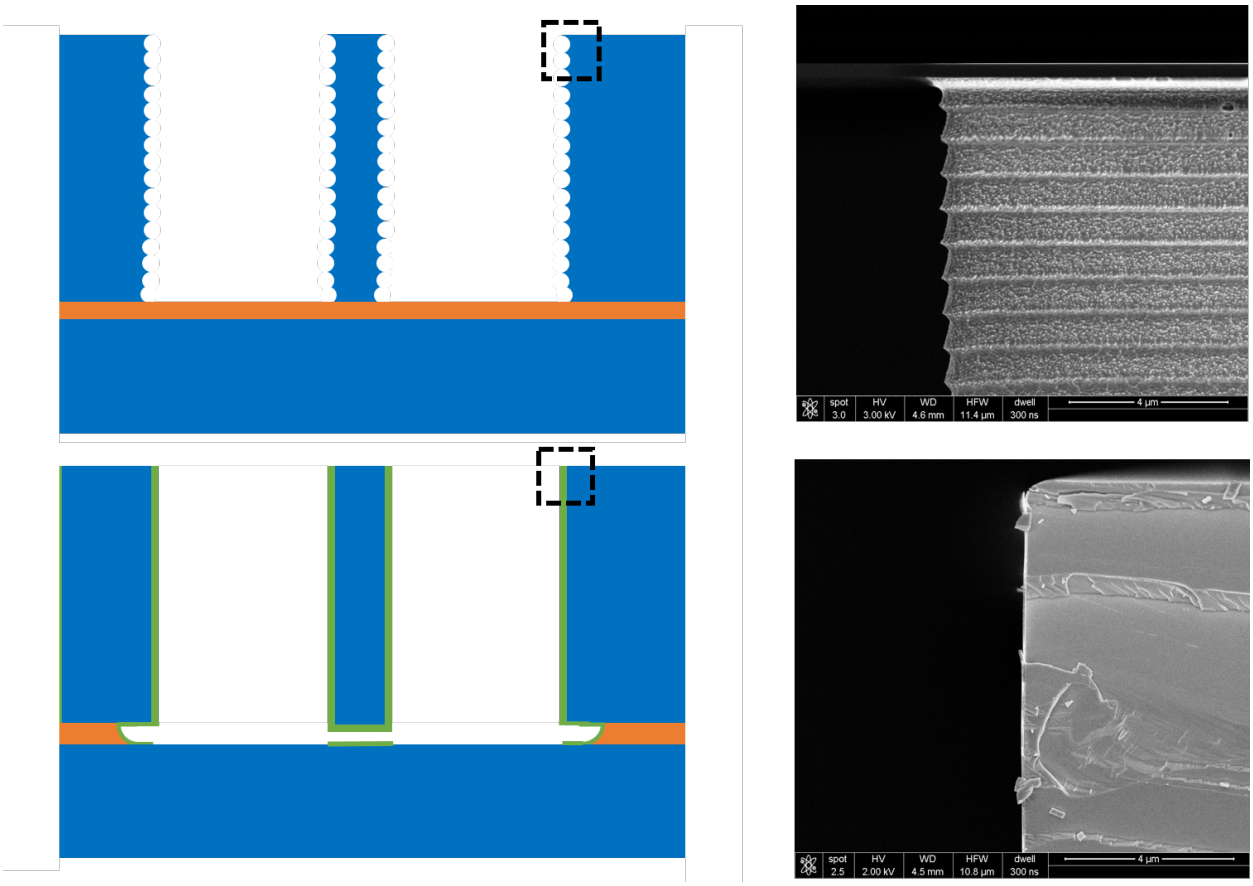

Figure 2.14: Cartoon and accompanying SEM cross sections of a wafer after the device side DRIE (top) and after the final step of the low-voltage latch process (bottom).The SEM images in the right column depict the areas contained in the dashed boxes of the left column.

the electrostatic gap from 0.5 μm to 50 nm would theoretically increase the electrostatic force by a factor of 100. The added benefit of this process is that it is capable of depositing relatively high-κ materials. The relative permittivity of materials such as these can easily be over 5, which in turn generates an electrostatic force that is 5 times greater than it would be with an air gap.

There is one major issue with this approach, however. If the sidewalls are brought into direct contact with each other, they must be exceedingly smooth. Equation 2.17 assumes that the two sides of this gap are perfectly smooth parallel plates. The DRIE process does not generate smooth sidewalls, but rather sidewalls with scallops as shown in the cross section in figure 2.13b. These scallops can be tuned by changing the process parameters of the DRIE, however this will also change the characteristics of the etch. Once a stable and high quality etch process is developed it is highly undesirable to tweak it, so an additional processing step is added to smooth out the sidewalls after the DRIE step is complete. Lee and Wu have

shown that silicon sidewall roughness from DRIE can be smoothed dramatically in a low pressure high temperature hydrogen anneal process [21]. Typically performed in an epitaxial growth furnace, when silicon is heated to 1100 C the hydrogen enables the silicon atoms to diffuse along the silicon surface. The atoms will settle into a configuration that minimizes the surface energy, and thus the surface roughness, of the entire structure. By choosing the process pressure and anneal time carefully, one can effective smooth out all features under a certain radius of curvature. In this process that minimum radius of curvature is set to be roughly 1 μm. After the hydrogen smoothing is performed, the dielectric must be deposited. There are many different means of depositing dielectrics, however the one that suited our needs best was atomic layer deposition (ALD). This technique generates a high quality film with reliable and repeatable control of the thickness. This technique is also desirable because the process can be performed at the chip and wafer scales, whereas processing chips in a furnace would be exceedingly difficult. While many materials can be deposited via ALD, the dielectric chosen here is aluminum oxide (Al_2O_3). This is a very well understood well characterized ALD process. In fact it was one of the first materials for which ALD was developed [22]. Al_2O_3 has a relative permittivity of about 8, and a breakdown field of 500 $\frac{V}{\mu m}$ [23]. These properties make Al_2O_3 a very attractive material for the gap dielectric.

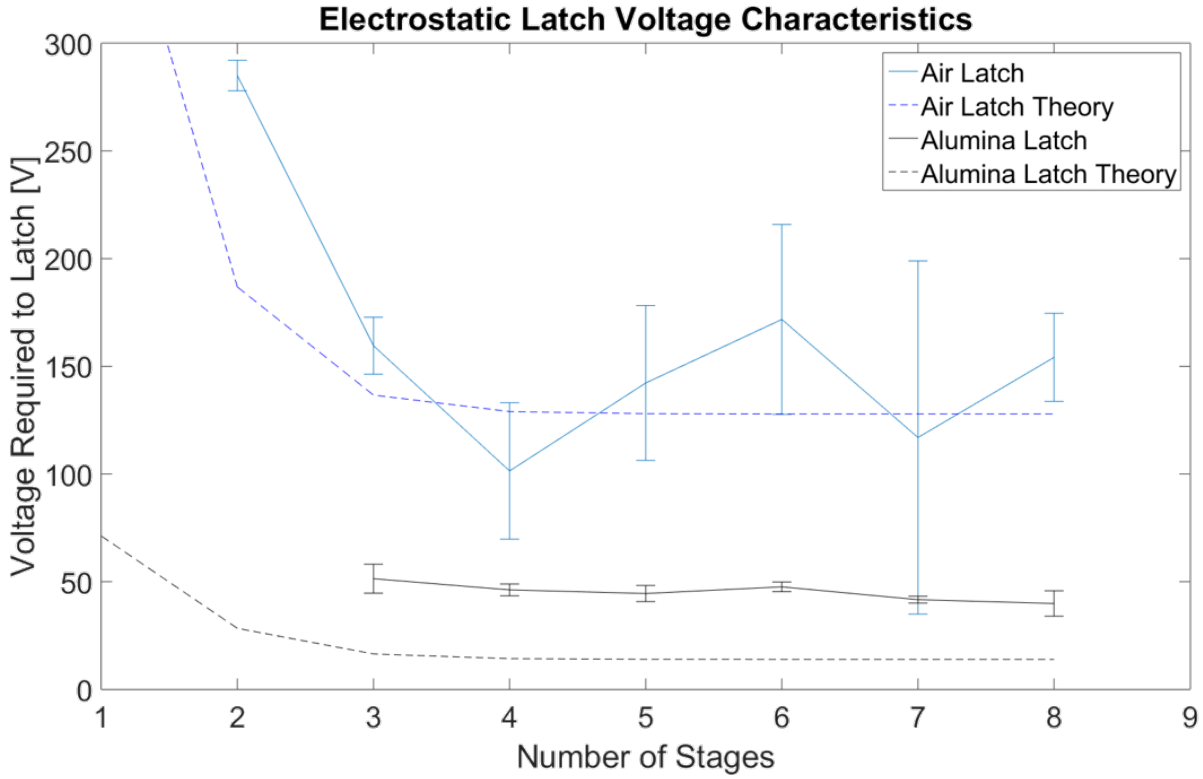

Figure 2.15: Plot showing the data of latches with alumina gaps vs air gaps. These latches are restraining a hammer that applies 173 mN.

The modifications to the standard two-mask SOI process are shown in Figure 2.13. The process starts with a the same single crystal silicon SOI wafer with a 40 μm device layer, a 2 μm buried oxide layer, and a 550 μm handle wafer. From here the front side DRIE is performed just as it was previously. Next the above-mentioned hydrogen anneal is performed. The process pressure is 10 T, the processing temperature is 1100 C, and the total anneal time is 10 minutes. Generally, this should smooth out all features with a radius of curvature less than 1 μm. From here, depending on the exact device being made, the backside DRIE process can be performed the same as is in the standard two-mask SOI process. After this optional step, the devices (either on the entire wafer or on individual chips) are released in a vapor HF tool to undercut the oxide. Next the ALD is performed. 50 nm of Al_2O_3 is deposited everywhere on the device as shown in Figure 2.13e. This would be the last step of the process, however all of the surfaces have a layer of Al_2O_3 on them. This insulator must be stripped from the top surfaces of the silicon structures so that a voltage can effectively be applied to the silicon. To do this an Al_2O_3 RIE is used to anisotropically remove the dielectric from the flat horizontal surfaces while leaving the dielectric on the vertical sidewalls mostly untouched. The cross section after this step is shown in Figure 2.13f. It is important to note the shadow of Al_2O_3 that doesn't get removed by the RIE under the released structures. The released section of device layer silicon acts as a mask over the Al_2O_3 on the substrate and protects it from being etched. Scanning electron microscopy (SEM) cross sections of a device after the DRIE step and after the final step can be seen in Figure 2.14.

An array of these multiple-stage low-voltage latches, one of which is shown in Figure 2.16, was fabricated and tested to compare the air gap latch performance with that of the alumina gap latches. One of the main takeaways is that the latching voltage is much lower for the alumina latches than the air latches. This general trend is in line with the theory; the alumina latches require less voltage than the air latches. Additionally, the deviation in latch performance is much less with the alumina latches. They are much more repeatable and consistent than the air latches. However, the alumina latches require more voltage than the theory predicts. The most likely cause of this is the assumption inherent to Equation 2.17 that the plates of this capacitor are perfectly parallel. Any surface roughness or particles on the sidewall will have a large effect on the operation of this device.

Figure 2.17 shows an optical profilometry measurement of the sidewall of a lever arm. The vertical height of the structure is mapped to brightness, and the roughness on the lever arm itself is greater than 50 nm. This roughness could lead to the deviation from theory in the presented data. While the hydrogen anneal process nominally smooths out all features less than 1 μm in radius of curvature, there is a secondary process occurring. Lee shows that etching occurs at the interface between the device silicon and the buried oxide [24]. Both the silicon and the oxide are consumed during this process and one of the byproducts is water. This water can easily oxidize other areas of the silicon which will in turn be etched by the same process. This is undesirable for two reasons. First, this oxidation can occur on the sidewalls themselves and lead to less smooth surfaces, as we see in our devices. Secondly, it releases the SOI from the buried oxide, and thus the substrate, creating a potentially long slit at this interface as shown in Figure 2.18.

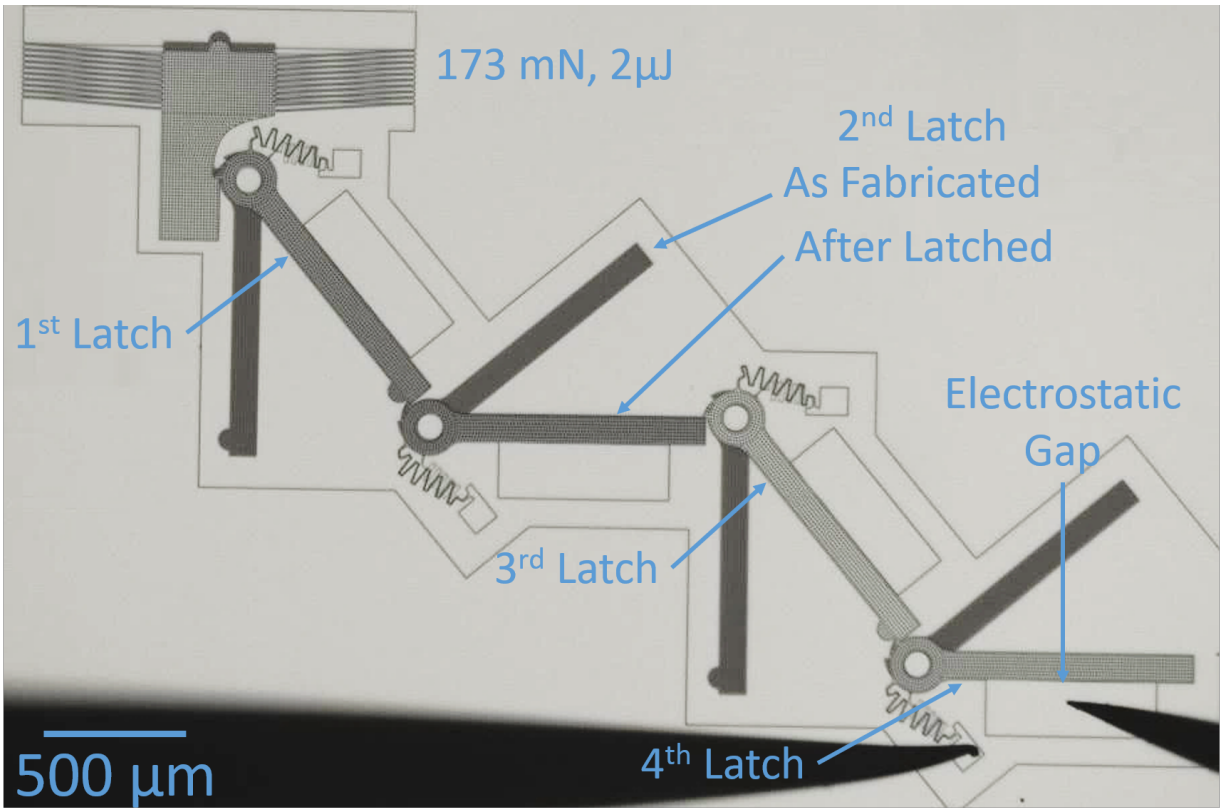

Figure 2.16: Image showing a multiple stage low-voltage latch. The labels show the current position of the lever arms with the alumina shadow of the 2^{nd} stage also labeled.

This effect hinders the ability to create good mechanical anchors, which could render the entire wafer unusable. It is possible that the parameters of the hydrogen anneal could be modified and tuned to combat these effects and further reduce the surface roughness, thus decreasing the operating voltage of the latches. However, the latch voltage was sufficiently low to perform a proof of concept self-destructing silicon demonstration, so no further process development was performed.

The hydrogen anneal process has an additional beneficial effect on the energy storage elements used for the MEMS hammers. Footing, also called notching, is an undesirable second order effect of the Bosch DRIE process that can be at least partially ameliorated with this anneal step. Footing occurs when the DRIE process is used on a material with a buried dielectric, such as an SOI wafer. This etch uses positively charged ions that are accelerated from the plasma toward the wafer. When the etch front reaches the dielectric interface, the ions begin to charge the top surface of the dielectric. This charge builds up and starts to affect the trajectory of incoming positive ions. These ions are deflected laterally and hit the bottoms of the sidewalls. This leads to a nasty lateral etch, which vaguely resembles

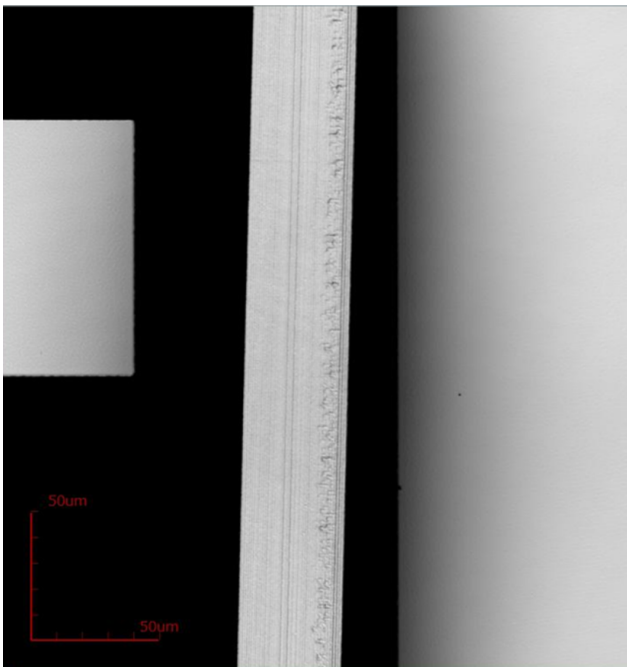

Figure 2.17: Confocal microscopy image of the sidewall of a lever arm after sidewall smoothing processing step.

a foot, at the bottoms of DRIE features. Figure 2.19 depicts a cross section SEM showing how jagged and rough the footing leaves the silicon.

Footing can lead to critical failures when attempting to maximize the energy stored in a silicon beam. Table 2.3 shows a range of values for the fracture strain in silicon. The given range, 0.5% - 1.0%, is generally what has been seen in this research, however as stated previously, fracture strains of up to 6% have been reported [17]. This wide range in fracture strain is at least partially due to how the sample is prepared. If the sample is etched, the sidewall profile and roughness will affect the sample's fracture strain. Small sharp features, such as those seen in Figure 2.19, act as stress concentrations that concentrate the stress in the material and effectively reduce the fracture strain of the feature. To make matters worse, footing occurs at the interface of the beam and its anchor where the strain is already at a maximum. Without the hydrogen anneal process, the beams have these jagged stress concentrations and the maximum strain is limited to 0.5%. When the anneal is added to the process, these jagged features are smoothed away and the maximum fracture strain doubles to 1.0%. This leads to a 4 times increase in the total energy that can be stored in these beams.

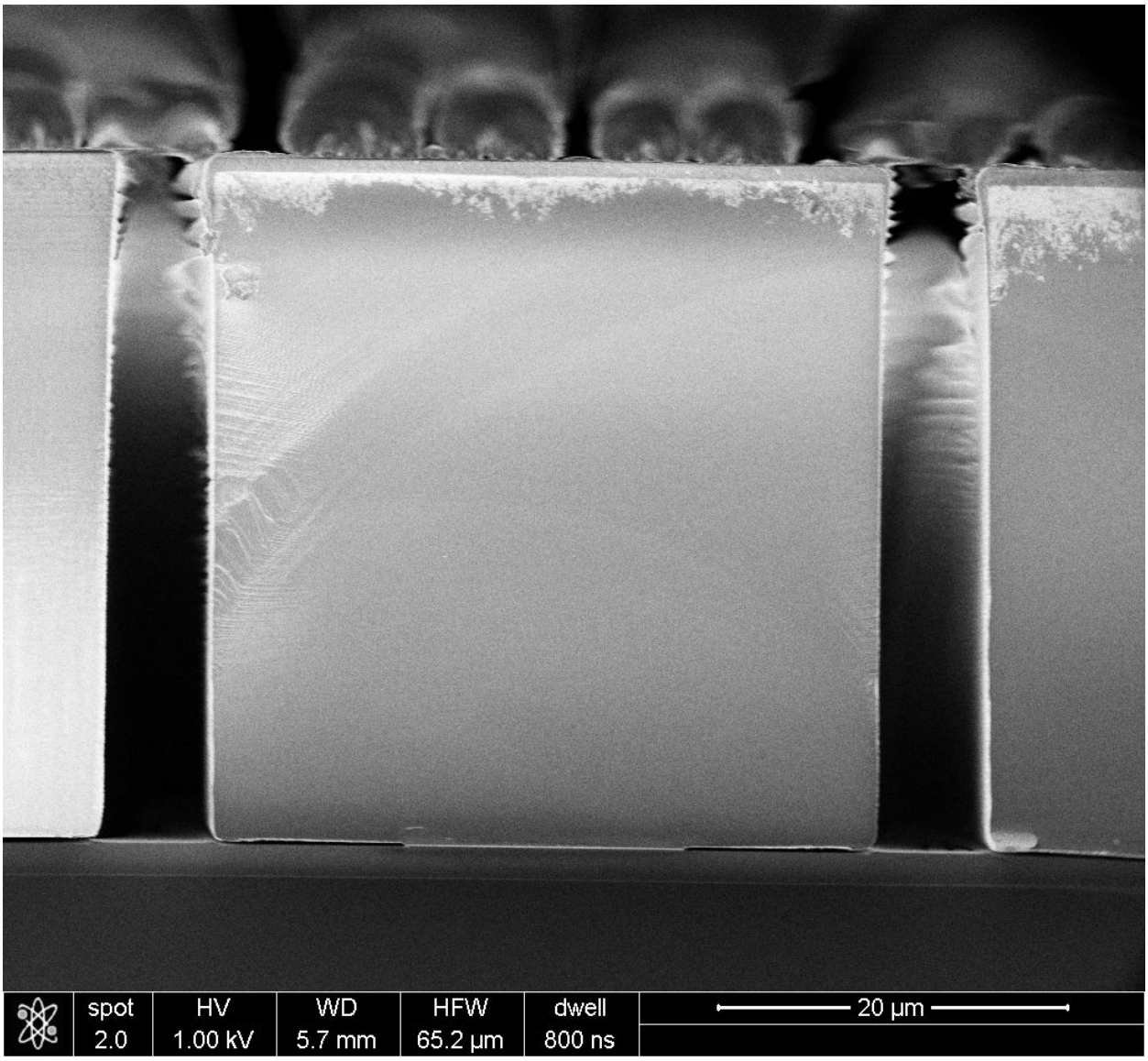

Figure 2.18: Cross section SEM image of 12 μm oxide undercut from the hydrogen anneal process.

2.5 Assembly and Integration

Now that each component of the self-destructing silicon project has been designed and fabricated, the entire system can be assembled and tested. The system comprises the MEMS hammer chip, the membrane chip, the XeF_2, and the package. The two MEMS chips are shown in Figure 2.20. The hammer chip looks slightly different from the multiple stage designs show up to this point. First the total area of the electrostatic latch has been optimized

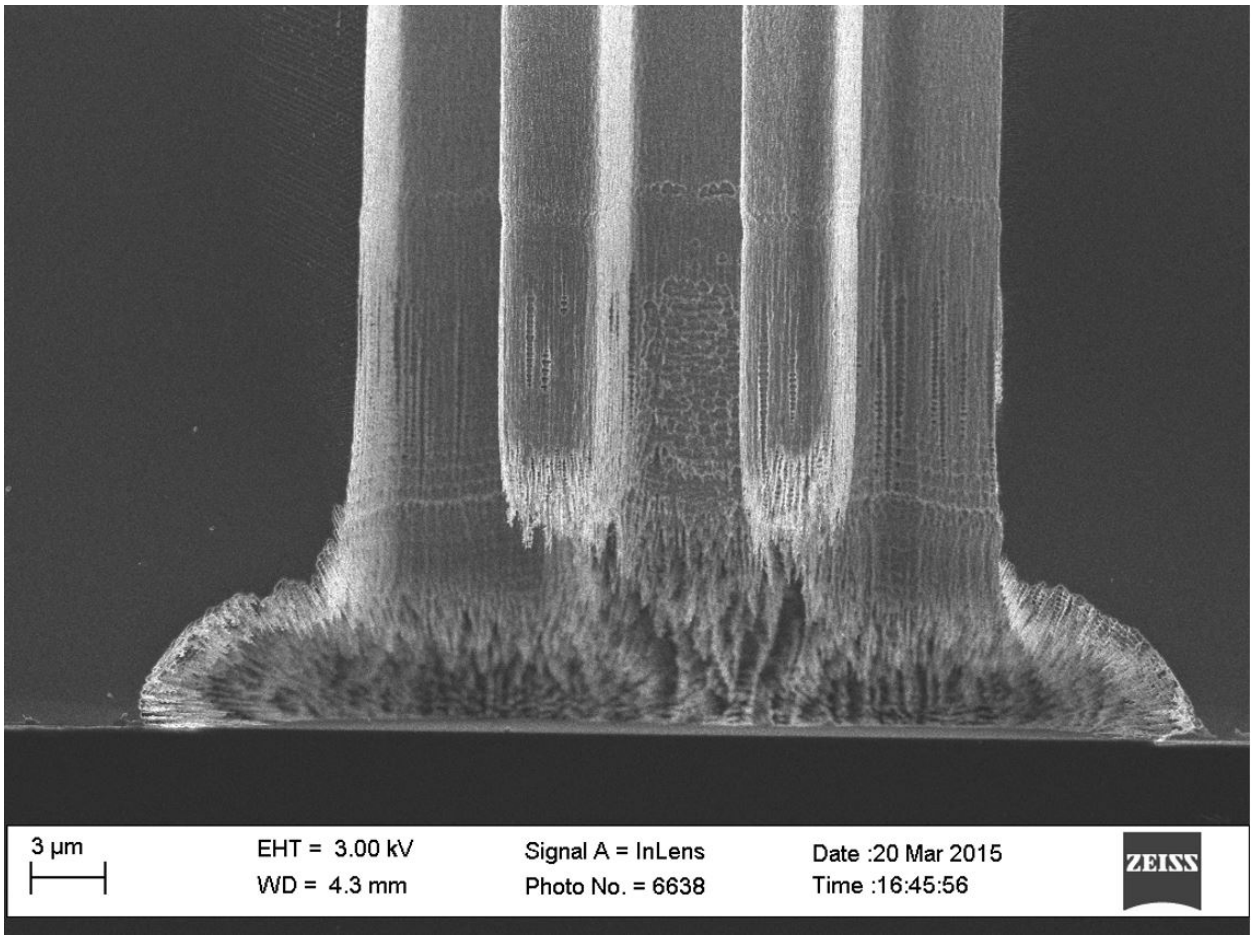

Figure 2.19: A cross sectional SEM of the footing of three etch holes that have terminated on the buried oxide.

to fit on a chip. To do this, the high voltage contact for each stage except the last was removed. This allowed the lever arms to be rotated and translated in a way that allowed a 4 stage latch to fit cleanly in a confined area. This compact design did not affect the performance of the electrostatic latch. Additionally, the hammer chip has two pillars that extend from the top of the chip. This chip will be rotated 90 degrees such that these pillars point straight down. From here, the pillars align with the rectangular holes in the membrane chip, shown in Figure 2.20. The hammer chip is inserted into the membrane chip and the two chips have been successfully assembled. To begin, the membrane chip must first be affixed to the aluminum disk. An off the shelf epoxy glue is used to bond the two together. The modularity of the package design is critical here so that the epoxy can cure not in the presence of XeF_2. After this the Hammer chip must be prepared for insertion into the membrane chip.

Before the hammer chip can be assembled into the membrane chip, the MEMS hammer

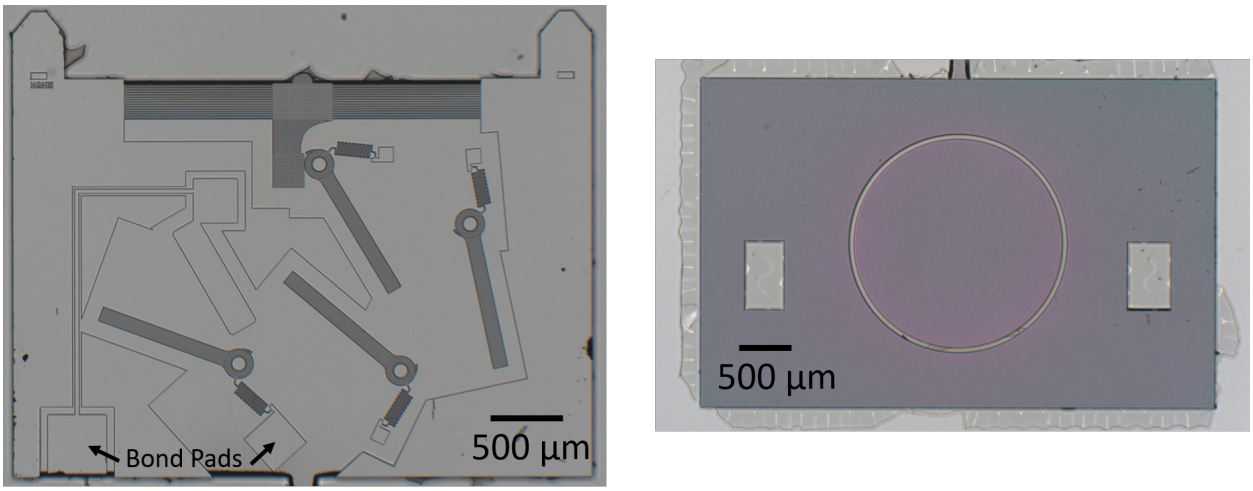

Figure 2.20: The two MEMS chips for the self-destructing silicon project. (left) The MEMS hammer chip with bond pads for the electrostatic latch labeled and (right) the membrane chip.

must be prepared electrically and mechanically. First, wires are bonded to the hammer chip. These wires will eventually send the signals to the low-voltage electrostatic latch. Wire bonds cannot be used because the chip must be handled and rotated afterwards; the wire bonds would not have enough slack to allow for this. Instead of wire bonds, thin insulated copper wires are used with a conductive epoxy. These wires are 60 μm in diameter and have a thin enamel for electrical insulation. Two micromanipulators were used to bond the two wires required for the low-voltage latches under a dissection microscope. The epoxy used must be cured on a hot plate, so all the assembly must be done on the hotplate itself while it is turned off. The micromanipulators are placed into vices on the lab bench next to the hotplate. From here, a copper wire was wrapped around the probe tip on each micromanipulator with an exposed end of the wire hanging straight down. Now the wire can be maneuvered with the micromanipulator. This setup can be seen in Figure 2.21. A pneumatic pump dispenser is used to apply a small drop of silver epoxy to the bond pads where the wires will be attached. The micromanipulators are used to bring each wire into contact with its respective silver epoxy covered bond pad. Once the wires have touched down on the bond pads, the hot plate is turned to 150 C and the epoxy is cured for at least 30 minutes. After the curing step, the coiled wire is pushed off the probe tips using a pair of tweezers and the micromanipulators are removed from the setup.

After the hammer chip has been wired successfully, it must be electrostatically latched and inserted into the membrane chip. To do this, the hammer chip is placed under the probe station and held in place using a probe. Two separate probes are used to sequentially engage the lever arms of each stage until the final electrostatic stage is engaged. Once this happens the appropriate voltage is applied through the copper wires and the latch remains

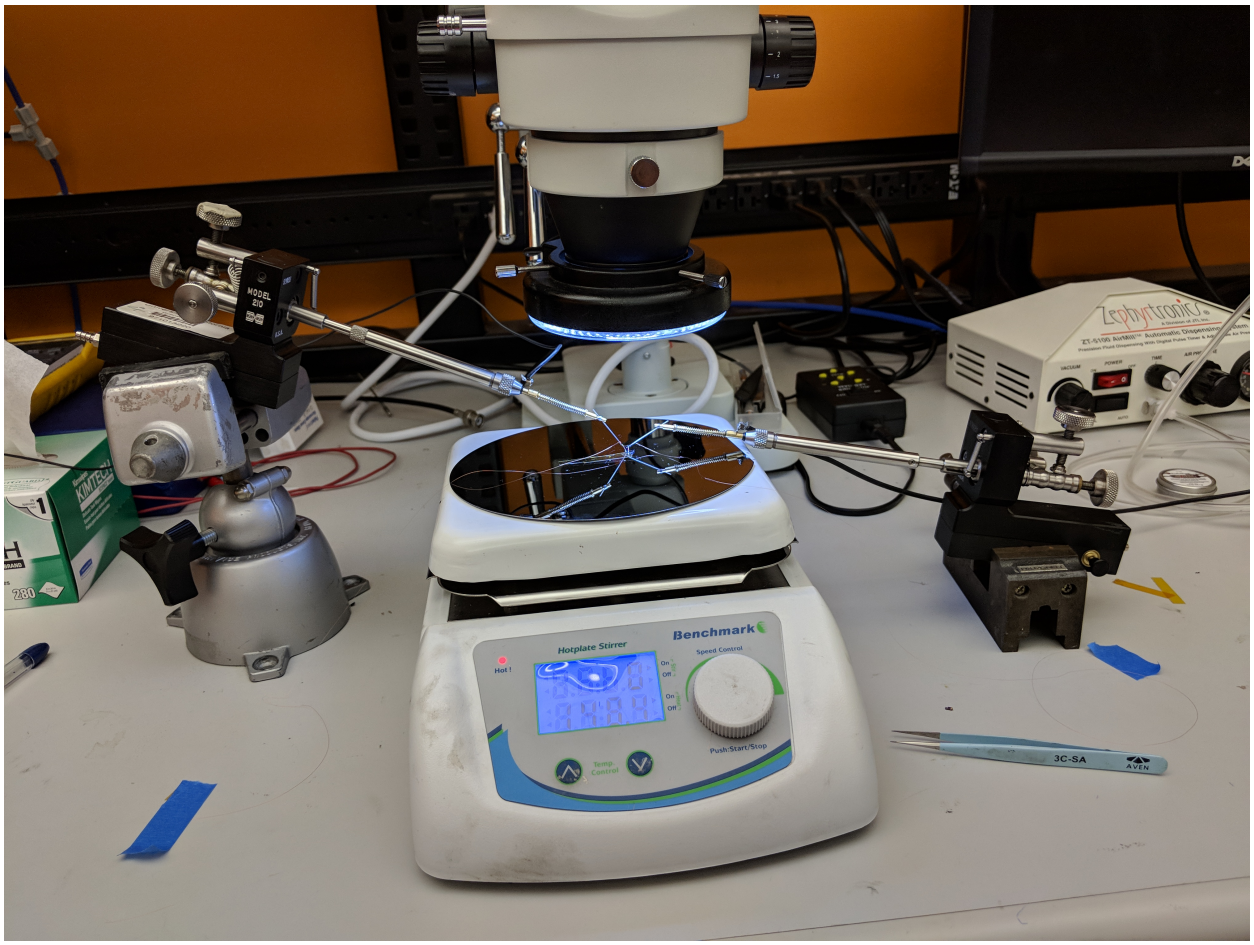

Figure 2.21: Setup used to precisely bond insulated copper wires to electrostatic latch bond pads.

shut. The hammer chip can now be handled with tweezers and inserted into the membrane chip. To do so, a 45 degree mirror is used to aid in the assembly. This mirror allows easy viewing of features and chips that are at right angles to the chuck of the probe station. Therefore, when the hammer chip is perfectly inserted into the membrane chip, the features on the hammer chip will be visible in the mirror. The final assembled chips can be seen in Figure 2.22. It is clear in this figure that the hammer chip is latched electrostatically by looking at the hammer chip through the 45 degree mirror, shown in the left image of Figure 2.22. The alumina shadows are visible in the locations where the lever arms were originally fabricated. Additionally the copper wires can be seen bonded to the pads of the electrostatic latched. The assembled chips were then tested under the probe station to ensure that all the components worked together. The voltage on the latch, 22 V in this case, was turned down to 0 V at the power supply attached to the copper wires. This caused the latch to release which allowed the hammer to fracture the membrane. Figure 2.23 shows the before and

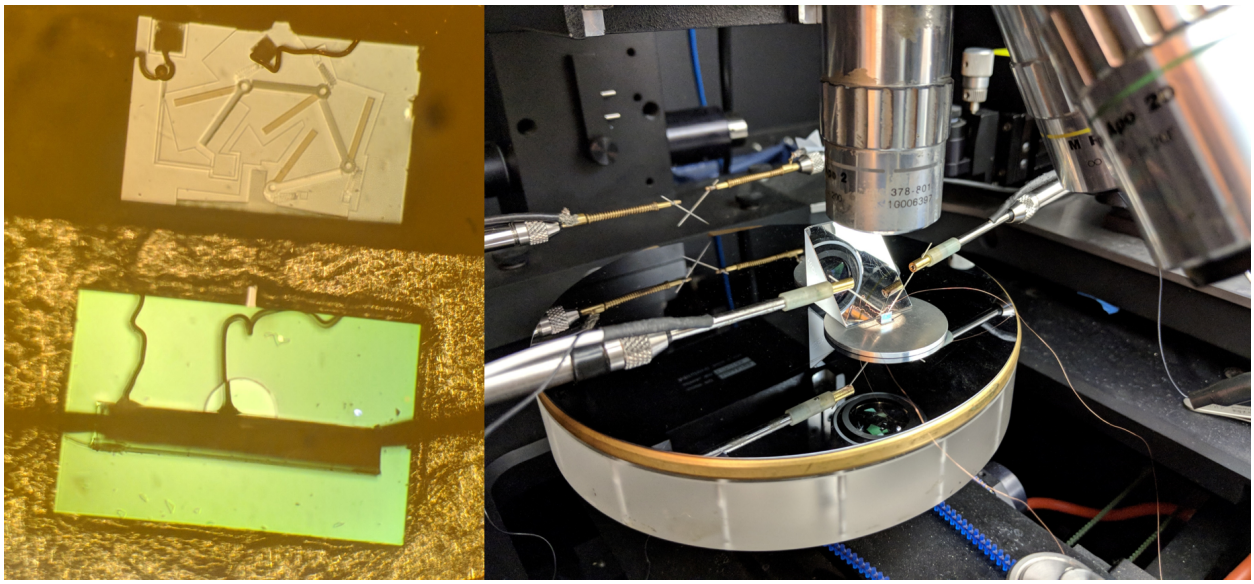

Figure 2.22: A 45 degree mirror is used at a probe station to aid in the assembly of the hammer chip into the membrane chip. (left) The image seen through the objective with the membrane chip shown in the bottom of the image, and the hammer chip, seen through the 45 degree mirror, shown in the top of the image. (right) A picture of the probe station setup. The two chips are seen on the aluminum disk with the 45 degree mirror sitting next to them, all under the objective of the microscope.

after images taken at the probe station. In the top image, the membrane is seen intact and the hammer is latched in its high energy position. After release, the bottom image shows the membrane fractured and the hammer head is no longer visible because it is below the surface of the membrane chip SOI.

Now that all the components have been proven to work together, the final demonstration can be completed. The same procedure was used to latch and assemble the chips on the aluminum plate. An excess of XeF_2, roughly 300 mg, was added to the bottom half of the PFA package. Separately, the same procedure was used to glue the membrane chip onto the aluminum disk, and assemble the loaded hammer chip into the membrane chip. Next the aluminum disk, with the two chips bound to it, was inserted into the package and rested on the PFA lip. The top of the package was threaded onto the rest of the package taking special care not to break the copper wires coming out of the top of the package. The whole package was put into an oven set to 70 C, and the voltage on the electrostatic latches was reduced to zero. This causes the hammer to fracture the membrane and the XeF_2 to attack both the MEMS hammer chip and the membrane chip. The setup was left for 10 hours at this elevated temperature. After the test was complete, the package was taken apart and the chips examined. The before and after pictures can be seen in Figure 2.24. While both chips are certainly etched, the etch did not fully vaporize the chips. This is due to non-

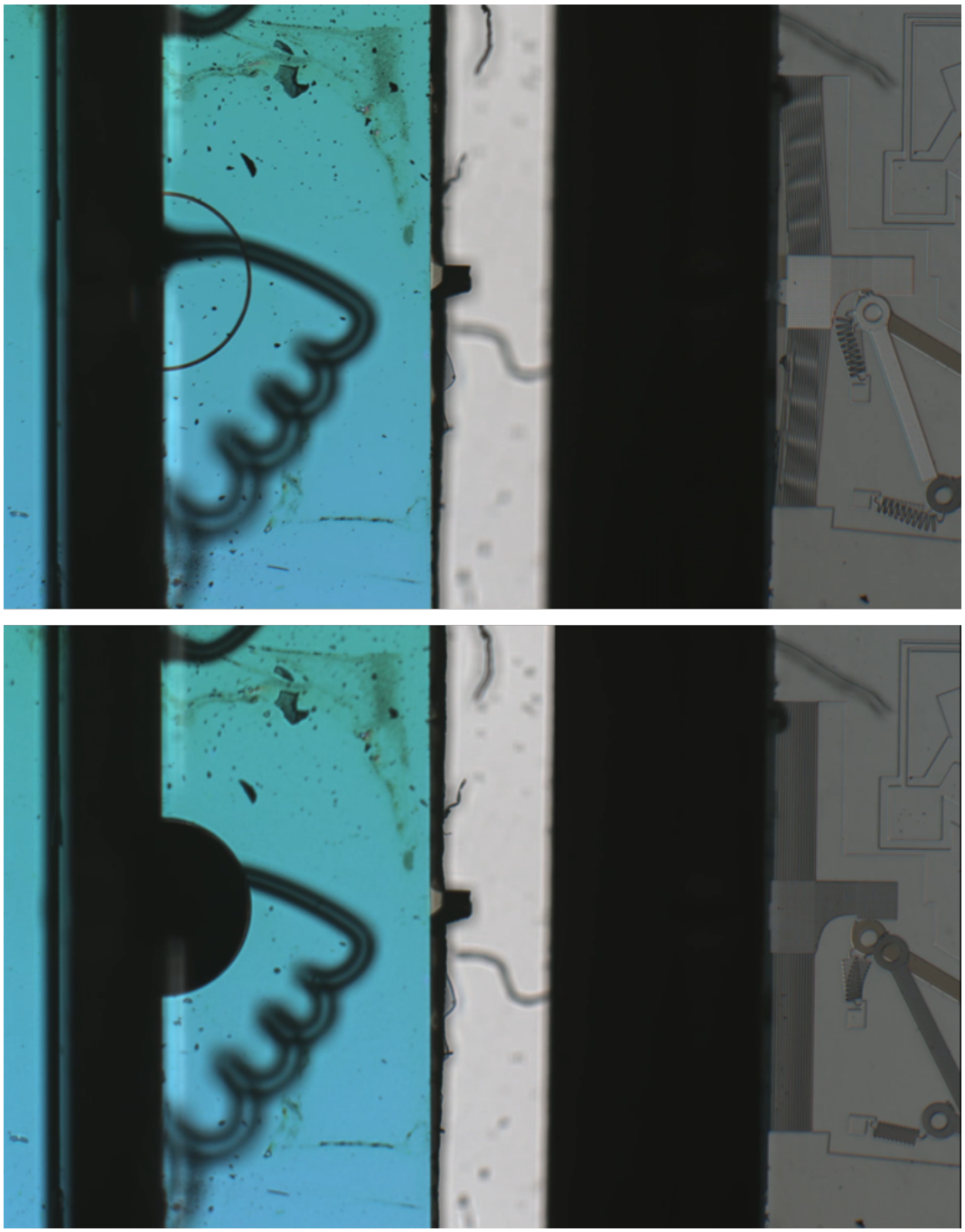

Figure 2.23: Before (top) and after (bottom) images of the electrically initiated fracture of a membrane. In both images, the left side of the image shows the physical chips and right right side shows the hammer chip as seen through the 45 degree mirror.

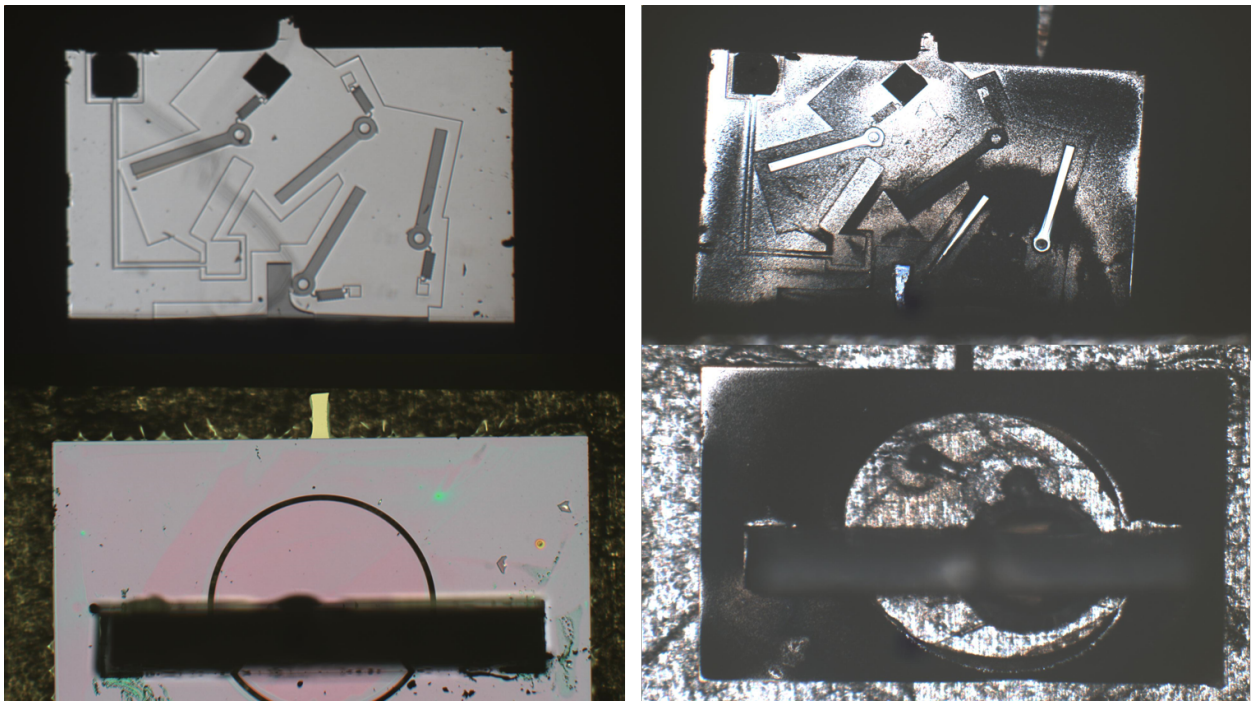

Figure 2.24: Before (left) and after (right) images of the hammer and membrane chips exposed to XeF_2 in package.

standard etch conditions inside the package. Typically in a XeF_2 etch process, the system is pulsed. Initially, gaseous XeF_2 is introduced into the etch chamber. Every few minutes the remaining XeF_2 and any etch byproducts are pumped out of the chamber, and new XeF_2 is pumped into the etch chamber. This leads to a steady and relatively constant etch rate. In this setup however, the etch products and the entire volume of XeF_2 sit in the same chamber until the test is complete. This leads to the result we see here, incomplete etching of the silicon chips. Although the chips weren't fully etched, the difference in mass between these MEMS chips and an actual IC will help fix this problem. The total volume of silicon in these two chips is roughly 15 mm^3, whereas the volume of a silicon based IC could easily be thinned for a total volume of 0.4 mm^3. This reduced mass would be much more easily etched to completion with the system we developed.

While this integration effort successfully broke a membrane and partially etched these MEMS chips, the signal required to do so was 22 V in the best case, and a standard microcontroller cannot generate such a voltage. To remedy this, additional devices were designed and fabricated to create a low voltage relay that could be triggered with a microcontroller. This device looks similar to the air gap latches with the addition of a resonator next to the high voltage contact. Figure 2.25 shows the device after it has been latched (top) and after the trigger signal has been applied and the latch has swung open (bottom). The four probe tips visible in the left side images send the electrical signals to the device. Probe 1 applies a

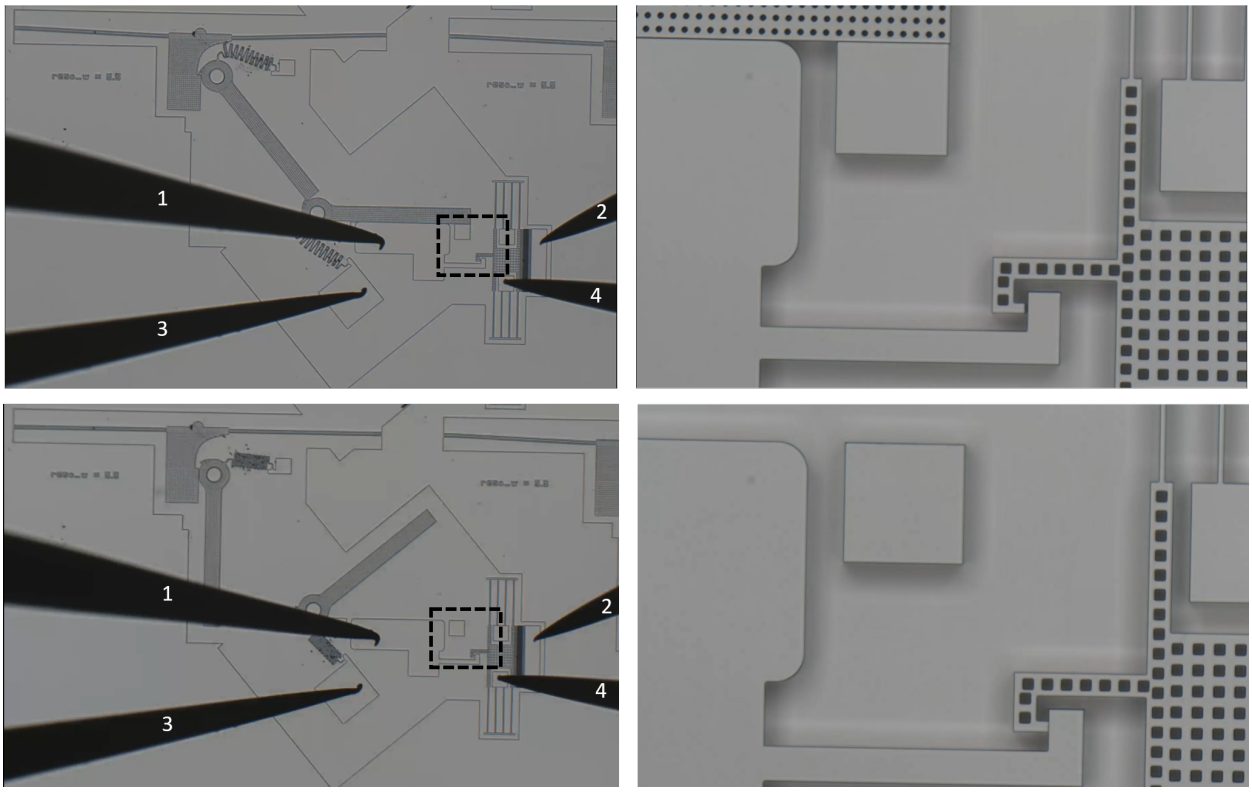

Figure 2.25: MEMS hammer and electrostatic latch before (top) and after (bottom) the 3.3 V square wave is applied. The dotted boxes in the left column correspond to the zoomed in images in the right column.

62 V signal through a 1 $M\Omega$ pull-up resistor. This resistor could be integrated and fabricated in the SOI layer in future iterations. Probe 2 applies the same 62 V signal to one side of the comb finger array, with no pull-up resistor. Probe 3 is grounded. Lastly, probe 4 supplies the 3.3 V square wave that triggers the release of the latch. During latched operation, the high voltage signal from probe 1 keeps the latch closed. Then, when a microcontroller provides the 3.3 V square wave at the resonant frequency of the resonator on the bottom right, it starts to resonate and shortly thereafter, it makes contact with an extension of the high voltage contact. Upon contact, the resonator pulls the voltage on this pad down. This in turn causes the lever arms to open and the hammer to accelerate forward. While this triggering method does still require a large bias voltage of 62 V, the only voltage that needs to be switched to trigger the hammer release is the low 3.3 V signal. This signal could be sourced natively from SCμM, the system on chip developed by the circuit designers in the Pister group.

Chapter 3

Jumping Microrobots

One of the many unsolved problems in microrobotics is the development of an efficient and reliable means of locomotion. This ability to move from place to place is a defining characteristic for a microrobot. While many research groups have worked on various means of microrobot mobility, it remains a challenging and exciting area of research. Inherent to microrobot mobility is the ability to operate autonomously and free from tethers or external power sources. Arguably the best example of a fully autonomous microrobot is from 2003 when Hollar et al. developed a 10 mg solar powered silicon robot [10]. It was an incredible integration effort, combining three chips from separate processes that each served a different purpose. One chip contained the MEMS motors and legs that allowed the microrobot to move, another contained the solar cells and high voltage buffers that gave the microrobot power, and lastly a CMOS sequencer chip that sent control signals to the motors through the high voltage buffers. While every component of this system was meticulously designed and incredible fabrication processes were developed, when all was said and done the microrobot only managed to spin clockwise about 20°. This is not to belittle the research effort by Hollar, but rather to build upon what they learned. For autonomous tether-free operation, the microrobot will either need to store power on-board or scavenge from its surroundings. Either way, the motors, logic, and buffers must be designed for low power operation. The Hollar microrobot performed fantastically here. It generated 100 μW of power from its solar cells and consumed 100 nW in the motors and legs, 2.5 μW in the buffers, and 22 nW in the sequencer. This microrobot was capable of generating an excess of 10 times the power its components required to run. Where the microrobot fell short however was in is leg and mobility design. The idea was for two single degree of freedom legs to drag the robot body forward, similar to an army crawl. While the mechanisms themselves moved as expected, the microrobot could not propel itself forward even on a smooth surface of machined aluminum. Needless to say, if this microrobot were to attempt to traverse a real world environment, even the smallest of obstacles would prove an impassible barrier.

The microrobot proposed in this work aims to build from the successes of the Hollar crawling microrobot and improve upon the mobility issues by changing the core locomotion method. Walking and crawling microrobots are inherently limited in the obstacles they can

overcome by the stroke of their legs. Generally, if the microrobot cannot lift its leg over an obstacle, it cannot pass over that obstacle. In the real world there will certainly be obstacles many times the size of these microrobots, so if they want to move around effectively in these environments, walking and crawling should be avoided. For this reason, a jumping microrobot was developed. The main advantage jumping offers is the ability to overcome these obstacles that would impede the walkers. However, this comes at a cost. While walkers can generally move forward at a constant speed, jumpers generally need to spend time storing energy before they can jump. For insects this involves loading an elastomeric protein, resilin. For the microrobots presented here, the energy will be stored in silicon beams.

3.1 Jumping Mechanics

An additional benefit of using jumping for locomotion is that the simulation of its performance is relatively straightforward. The microrobot operates in two regimes: one where it stores energy, and another where it releases that stored energy to jump. When the system is in the first stage, the MEMS components can be designed such that they largely do not need to interact with the surface on which the microrobot is resting. In the second stage, the microrobot can be thought of as a mass that has some initial stored mechanical energy that gets released and turned into kinetic energy. This separation of the MEMS operation from the jumping mechanics allows for the same separation in the design and simulation space. First the required stored energy can be determined using simple physics models. Following this, the MEMS structures can be designed and tested such that they store that energy.

Determining the energy required to accelerate a mass a certain distance vertically and horizontally is not a difficult problem. Imagine a point mass that has some initial kinetic energy, $U_{kinetic}$. This energy can be written as a function of the mass, m, and velocity, v_i, of the object as follows:

$$U_{kinetic} = \frac{1}{2} m v_i^2 \tag{3.1}$$

This equation can be rearranged to solve for the initial velocity of the mass, and furthermore this velocity can be split up into horizontal, v_{ix}, and vertical, v_{iy}, components based on the initial angle, θ, of the mass:

$$v_i = \sqrt{\frac{2U_{kinetic}}{m}} \tag{3.2}$$

$$v_{ix} = \cos(\theta)\sqrt{\frac{2U_{kinetic}}{m}} \tag{3.3}$$

$$v_{iy} = \sin(\theta)\sqrt{\frac{2U_{kinetic}}{m}} \tag{3.4}$$

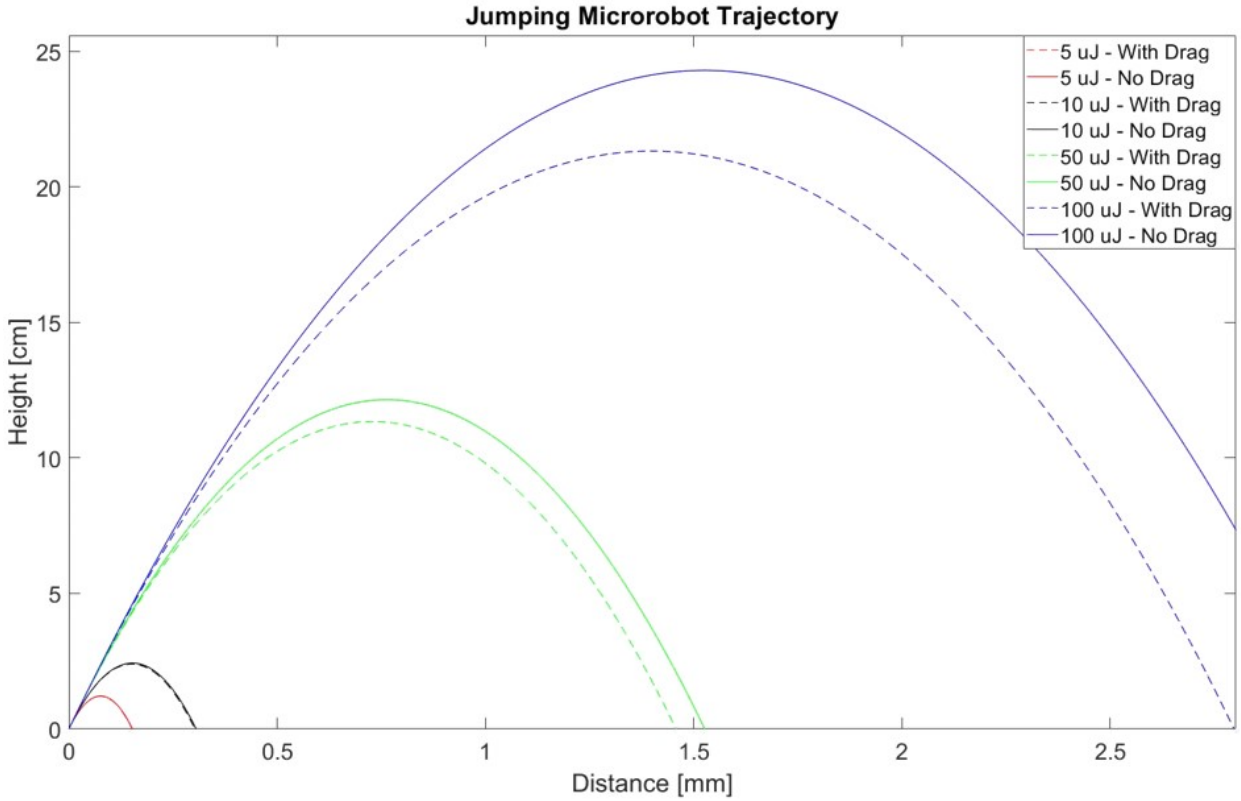

Figure 3.1: Comparison of microrobot trajectories with varying initial kinetic energy. m = 43 mg, A = 4 x 7 mm^2, $C_D = 1.5$, and $\rho = 1.2\ \frac{kg}{m^3}$.

From here the trajectory of this mass can be described as follows, where g is the gravitational acceleration and t is time:

$$x(t) = v_{ix}t \tag{3.5}$$

$$y(t) = -\frac{1}{2}gt^2 + v_{iy}t \tag{3.6}$$

Equations 3.5 and 3.6 fully describe the simplified motion of this point mass. However, this analysis ignores the effect of drag on the microrobot. While this might be ok with a larger scale jumper, the small masses here require a model that incorporates drag. The force due to drag on an object can be written as follows, where C_D is the drag coefficient, ρ is the density of the material through with the object travels, A is the cross-sectional area of the object, and v is the velocity:

$$F_{drag} = \frac{C_D \rho A v^2}{2} \tag{3.7}$$

In this design, drag is a parasitic aerodynamic force that reduces the overall performance of the jumping microrobot. However, the microrobot also generates some useful aerodynamic lift force. This lift force is identical in form to Equation 3.7, with the coefficient of lift, C_L, replacing the coefficient of drag, C_D. In all jumping designs presented in this work, the coefficient of lift is so small that the lift force can be ignored. However, by designing the microrobot to maximize the ratio of C_L to C_D, the microrobot becomes a more efficient glider and the range of each jump could be extended significantly.

Using Equation 3.7, it can be shown [25] that the trajectories for a microrobot with drag are given by the following:

$$x(t) = \frac{2m}{C_D A \rho} ln \left(1 + \frac{C_D A \rho}{2m} v_{ix} t\right) \tag{3.8}$$

$$y(t) = \frac{2m}{C_D A \rho} ln \left[\cos\left(\sqrt{\frac{C_D A g \rho}{2m}} t\right) + \sqrt{\frac{C_D A \rho}{2mg}} v_{iy} \sin\left(\sqrt{\frac{C_D A g \rho}{2m}} t\right)\right] \tag{3.9}$$

Equations 3.8 and 3.9 along with Equations 3.5 and 3.6 are plotted in Figure 3.1 with the following constants, $m = 43$ mg, $A = 4$ x 7 mm^2, $C_D = 1.5$, and $\rho = 1.2$ $\frac{kg}{m^3}$. From studying jumping and flying insects on the same scale as the jumper proposed here, Bennet-Clark et al. determined that the drag coefficients for these animals was in the range of 1 - 1.5 [26]. These trajectories will obviously be a function of the initial takeoff angle, so the plotted curves were chosen to highlight the maximum height achievable by these microrobots. In other words, the takeoff angle is close to 90°.

Figure 3.1 and Equations 3.8 and 3.9 provide useful benchmarks for designing a jumping microrobot. The eventual goal of the prototype proposed in this work is for the microrobot to have the ability to jump 10s of cm high at a rate of multiple times per minute. It is clear from this figure that to reach a jump height of 10 cm, the microrobot will need to store 50 μJ of mechanical energy. Furthermore, this assumes that all of that stored mechanical energy gets converted into kinetic energy of the microrobot body. While there will be additional loses, such as the added weight and drag from any electrical tethers, this provides a good starting point for designing a microrobot capable of jumping 10s of cm.

3.2 Passive Jumping Microrobot

Now that the relationship between initial kinetic energy and jump height has been established, a prototype jumping microrobot can be developed. In the previous chapter, the MEMS hammer focused mainly on applying a force across a distance and Equation 2.15 was derived to describe that profile. Integrating this equation over the displacement gives the energy stored in N beams as a function of deflection:

$$U = N \left[8 \frac{E w^3 t}{L^3} \Delta x^2 + 2 \frac{E w t}{L^3} \Delta x^4\right] \tag{3.10}$$

Using Equation 3.10, the same beams used for the MEMS hammer can be used to create a jumping microrobot. Furthermore, the changes required to the layout are minimal, the hammer head must be replaced by a foot from which the chip can jump. Additionally, the same two-mask SOI fabrication process can be used so there will be no process development required. The initial prototype can be seen in Figure 3.2. This prototype is theoretically capable of storing 5.3 μJ of mechanical energy and jumping 4.5 cm high, due its low mass.

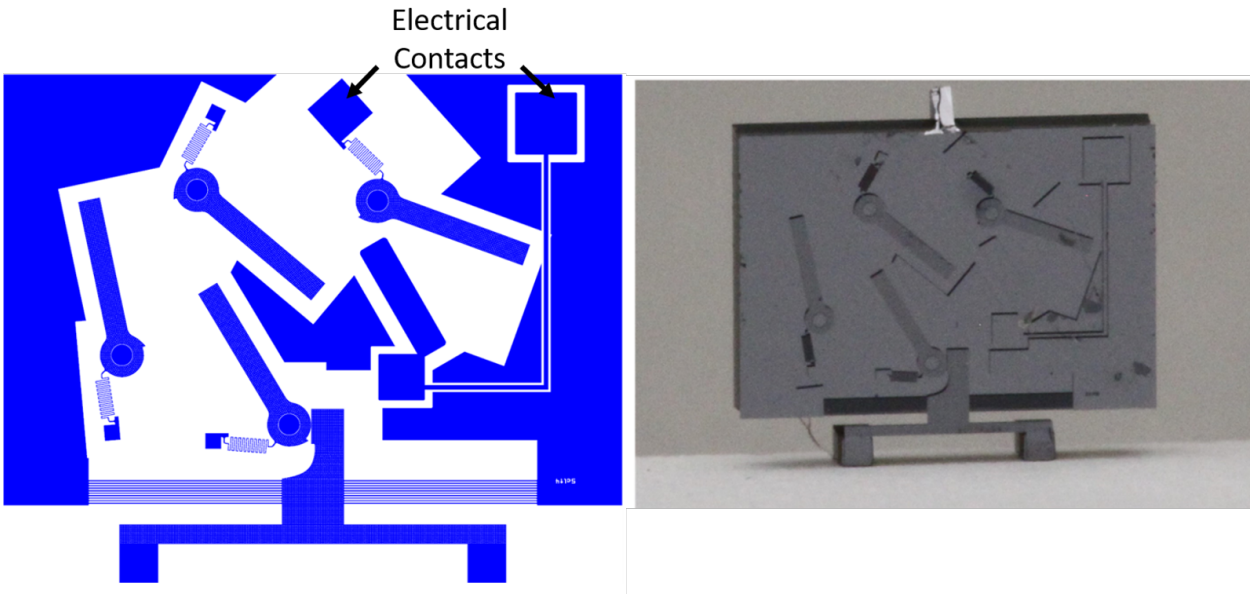

Figure 3.2: A prototype jumping microrobot in layout (left) and as fabricated (right). The chip is 4x3 mm^2.

This jumper prototype was tested in a similar way to the MEMS hammer. The same copper wire and silver epoxy bonding process was used to make electrical connection to the electrical contacts on the device. After this, the energy storage beams were pulled back at a probe station and the lever arms were armed sequentially. Finally, a sufficient voltage is put across the two copper wires and the microrobot is stood up on its two feet as shown in the right of Figure 3.2. Unfortunately, the mass of the device was so small, roughly 14 mg, and the stiffness in the copper wires so high that it was difficult to keep the chip upright and on its feet. Additionally, the full deflection of the beams was only 100 μm, so if the feet were not making perfect contact with the surface, the leg could release and extend fully before even touching the surface of the testing area. Due to these issues, the passive jumping microrobot never made it into the air, but it did provide some useful design insight for the next iteration of the jumper.

While it was appropriate for the MEMS hammer to be mechanically energized at a probe station, a jumping microrobot should have the ability to store mechanical energy independently. Eventually this microrobot will have power on-board that can be used to

run some motor which in turn does work to store mechanical energy. This would allow the next iteration of the jumping microrobot to jump multiple times, whereas in the best case, the passive jumper would only be able to jump once before needing to be put back under a microscope to be reloaded. Additionally, the stroke of the leg should be increased in the next iteration of the design. Here, the leg stroke of 100 μm was small enough that the foot would often extend fully and not make contact with the ground. If the same fixed-fixed beams are to be used with this longer stroke leg, the length of the beams will need to increase with the increasing stored energy, such that the strain remains below the fracture limit of the silicon. While this is possible, it would be better if the width of the chip and the energy stored could be decoupled. A more pressing issue with the fixed-fixed beams is the efficiency with which they can be loaded by an external force. The motors used in this design will be more thoroughly discussed in a subsequent section, but at their core they can apply a constant force over a distance of roughly 1 mm.

Imagine a device that has a set of 10 fixed-fixed beams similar to the jumper in Figure 3.2. If these beams are 2500 μm long (it should be noted that the length, L, of these beams is the sum of the length of silicon on the left side of the foot and the right side of the foot, making the beams centrally loaded and fixed-fixed), 5.7 μm wide, 40 μm thick, and deflect 100 μm they should store 5 μJ of mechanical energy and require a force of 200 mN to do so. If a motor, only capable of applying a single constant force, were to load these beams it would need to have an output force of 200 mN and a stroke of at least 100 μm. The issue here is that a motor with that performance could easily store twice the energy reported above using a different type of spring. A spring with a linear force-deflection profile would follow Hooke's Law, with a spring constant k, and the following energy equations:

$$F = k\Delta x \tag{3.11}$$

$$U = \frac{1}{2}F\Delta x \tag{3.12}$$

$$U = \frac{1}{2}k\Delta x^2 \tag{3.13}$$

It follows from Equation 3.12 that a linear spring would be able to store 10 μJ of energy with an applied force of 200 mN and a total deflection of 100 μm. So, while the centrally loaded fixed-fixed beams are great from an energy storage per layout area perspective, these beams underperform a simple linear spring when loaded with a constant force.

3.3 Linear Springs For Energy Storage

The springs used as the energy storage units in this design should be linear and able to deflect hundreds of microns to millimeters without fracturing. This large deflection is important for two reasons. First, as stated previously, a small stroke length for the leg of a jumping microrobot makes it less likely that the foot will make good contact with the surface

from which it jumps. Secondly, Equation 3.12 shows that for a given force, F, to maximize the energy stored in a spring, the deflection in that spring should also be maximized. The motor characteristics will set the limit on the maximum force, so the energy stored will be linearly related to how far the springs can deflect before fracturing.

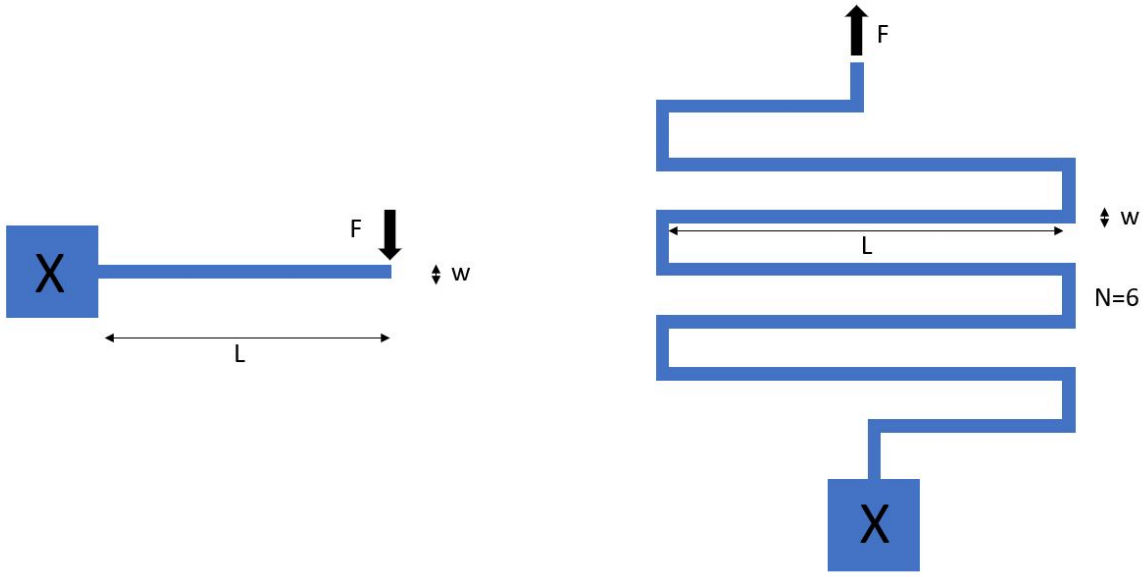

Figure 3.3: Layout view of a simple cantilever (left) and a serpentine spring (right) with the thickness dimension into the plane of the page.

A cantilever, shown in Figure 3.3, is a simple example of a linear beam. A cantilever with a width, w, a length, L, a thickness, t, and a Young's modulus, E, will have a force-deflection profile given by the following when the force, F, is applied at the tip in the direction of the width:

$$F = \frac{Etw^3}{4L^3}\Delta x \tag{3.14}$$

The maximum strain generated at the base of the cantilever will be:

$$\epsilon_{max} = \frac{3w}{2L^2}\Delta x \tag{3.15}$$

A simple cantilever has some notable drawbacks when being used as the energy storage element for a jumping microrobot. First the total deflection is limited significantly by Equation 3.15. For a cantilever to store the necessary energy while surviving the required deflections for the robot, it would have to be prohibitively long. Additionally, when a force is applied to the end of the cantilever the tip does not deflect in a single direction. As the

cantilever bends, the length of the cantilever effectively shortens. While it is certainly possible to design around this, ideally the energy storage element elongates in a single constant direction.

To fix both of these problems, multiple cantilevers can be combined together. If two cantilevers are put in series with their free ends touching, a fixed-guided beam is created. To find the spring constant of a single fixed-guided beam, the spring constants of both cantilevers are combined (using the parallel resistor combination formula), yielding a spring constant that is half that of the original cantilever. However, the length of this new beam is double that of the original cantilever. This gives the following equation for a fixed-guided beam:

$$F = \frac{Etw^3}{L^3}\Delta x \tag{3.16}$$

The maximum strain in this beam can be calculated as follows using equation 3.16:

$$\epsilon_{max} = \frac{w}{2}\frac{M}{EI} \tag{3.17}$$

$$\epsilon_{max} = \frac{w}{2}\frac{FL}{EI} \tag{3.18}$$

$$\epsilon_{max} = \frac{6w}{L^2}\Delta x \tag{3.19}$$

Both equations 3.16 and 3.19 are for a single fixed-guided beam. This unit can be repeated multiple times placing rigid trusses between each repeating unit. This creates a serpentine spring shown in Figure 3.3. In doing so, both the spring constant and strain are reduced by a factor equal to the number of beams, N. This gives the final design equations for the force-deflection profile and the maximum strain of this serpentine spring:

$$F = \frac{Etw^3}{NL^3}\Delta x \tag{3.20}$$

$$\epsilon_{max} = \frac{6w}{NL^2}\Delta x \tag{3.21}$$

Using Equations 3.20 and 3.21, serpentine springs were designed and fabricated using the two-mask SOI process. The serpentine spring was made up of 50 beams each having a width of 10 μm, a thickness of 40 μm, and a length of 369 μm. Using equation 3.20, the theoretical spring constant is 2.68 N/m. The Dage bond tester was used to verify this spring constant. Due to tool limitations, many springs were fabricated in parallel to increase the overall stiffness tested by the Dage, which has a minimum detectable signal of around 5 mN. The measured stiffness of an individual serpentine spring came to 1.62 ± 0.22 N/m. One possible explanation for this deviation from the theory is the non-infinite rotational stiffness at the intersection of the beams and trusses. This model assumes that the trusses do not

rotate or bend, but rather remain perfectly vertical. The following analysis allows bending of these trusses.

Using the standard Euler-Bernoulli beam equation where M_t is the applied moment on the truss, E is the Young's modulus, I is the area moment of inertia, and ρ_t is the radius of curvature of the truss, the analysis will be performed on the serpentine spring given in Figure 3.3:

$$\frac{1}{\rho_t} = \frac{M_t}{EI} \tag{3.22}$$

The applied moment bends the truss into an arc. The length of the arc is the initial length of the truss, L_t, the radius of curvature of the arc is ρ_t, and the angle swept by the arc is θ_t. These three variables are related by the following equation:

$$L_t = \rho_t \theta_t \tag{3.23}$$

By combining Equations 3.22 and 3.23, the following is obtained:

$$\theta_t = \frac{M_t L_t}{EI} \tag{3.24}$$

The angle swept by the entire truss, θ_t, is twice the angle of the tip of the truss, θ_{tip}. This tip angle is what will determine how much additional deflection the bending in the truss causes. Substituting in for the tip angle gives the following:

$$\theta_{tip} = \frac{M_t L_t}{2EI} \tag{3.25}$$

The moment applied to the truss, M_t , can be written as follows where F is the applied force, and L_b is the length of long horizontal beams of the serpentine spring:

$$M_t = \frac{F L_b}{2} \tag{3.26}$$

Combining Equations 3.26 and 3.25 gives the following:

$$\theta_{tip} = \frac{F L_b L_t}{4EI} \tag{3.27}$$

The moment induced by this force F will be identical in magnitude on each of the trusses in the serpentine spring, however the sign is opposite for the left-side trusses and the right-side trusses. Therefore, the left-side trusses bend to the left by θ_{tip} and the right-side trusses bend to the right by θ_{tip}. This leads to an additional extension in the serpentine spring, Δx_θ, which can be written as follows:

$$\Delta x_\theta = N L_b Sin(\theta_{tip}) \tag{3.28}$$

Using the small angle approximation and substituting in Equation 3.27, this additional serpentine spring deflection becomes:

$$\Delta x_\theta = \frac{FL_b^2 L_t N}{4EI} \tag{3.29}$$

$$\Delta x_\theta = \frac{3FL_b^2 L_t N}{tw^3} \tag{3.30}$$

Rearranging Equation 3.20 to solve for the deflection in the rigid truss serpentine spring gives the following:

$$\Delta x_{rigid} = \frac{FL_b^3 N}{tw^3} \tag{3.31}$$

Combining Equations 3.30 and 3.31 gives the total deflection of the serpentine spring:

$$\Delta x_{tot} = \Delta x_\theta + \Delta x_{rigid} \tag{3.32}$$

$$\Delta x_{tot} = \frac{3FL_b^2 L_t N}{Etw^3} + \frac{FL_b^3 N}{Etw^3} \tag{3.33}$$

The truss length, L_t, for this design is roughly one tenth the length of a horizontal beam, L_b. Plugging this value into Equation 3.33 gives the following serpentine spring constant:

$$F = \frac{10Etw^3}{13NL^3}\Delta x \tag{3.34}$$

Equation 3.34 with the above spring parameters gives a theoretical spring constant of 2.06 N/m. For the serpentine spring that was fabricated and tested, the measured spring constant falls below both of these theoretical values. For each serpentine spring designed in this work, Equations 3.20 and 3.34 were used to approximate the spring constant and test structures were fabricated to measure the exact spring constant. Equation 3.21 sets the maximum operating deflection of the serpentine spring. The fracture strain in these serpentine springs, when the hydrogen anneal step is performed, generally is 1.0%. At peak deflection the beams are designed to have 0.9% strain. With this design limit in place, the beams have never fractured during operation. Now that an appropriate energy storage element has been designed for the jumping microrobot, motors and mechanisms must be developed to actively load these structures.

3.4 Electrostatic Inchworm Motors, Mechanical Advantage and an Inchworm of Inchworms

In the previous chapter, a series of lever arms was developed to amplify the electrostatic force such that it could balance out a much large force from a MEMS hammer. That exact method cannot be implemented here, because it required a manual input at a probe station

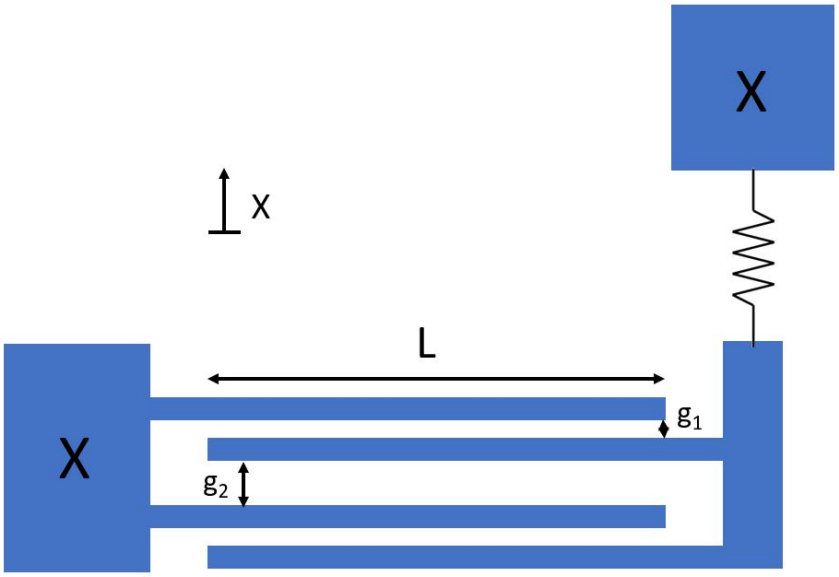

Figure 3.4: The unit cell for a gap closing array. Many sets of these comb fingers are arrayed to generate large electrostatic forces.

to achieve that mechanical advantage. However, that system can be used as a starting point for designing motors and mechanisms capable of storing 10's of microjoules into some energy storage element. For a jumping microrobot at this size scale to jump 10 cm high, Figure 3.1 shows that about 50 μJ of mechanical energy must be stored and converted into kinetic energy. To determine the force required to store this energy into an appropriately sized serpentine spring, Equation 3.12 can be used. While there is no hard limit on the total deflection of this spring, 2 mm is a good maximum value when considering process limitations and area restrictions. With this deflection, to store 50 μJ of energy, a force of 50 mN would be required. While this force is considerably lower than the forces generated by the MEMS hammers, an on-chip motor will be responsible for generating it as opposed to a human at a probe station.

The actuator tasked with generating this force is an electrostatic inchworm motor. This is the same low power motor that Hollar used for his crawling microrobot. The first iteration of this motor was developed in 2001 by Yeh, Hollar and Pister [27]. The device uses gap closing arrays (GCAs) to generate an electrostatic forces. The unit cell of a GCA is shown in Figure 3.4. When a voltage, V, is applied across the two sets of fingers, the following force works to pull the movable fingers up in the positive x direction:

$$F_{ES} = \frac{1}{2}\epsilon_0 N V^2 L t \left(\frac{1}{(g_1 - x)^2} - \frac{1}{(g_2 + x)^2} \right) \tag{3.35}$$

The low power and high force nature of this actuator makes it an ideal building block

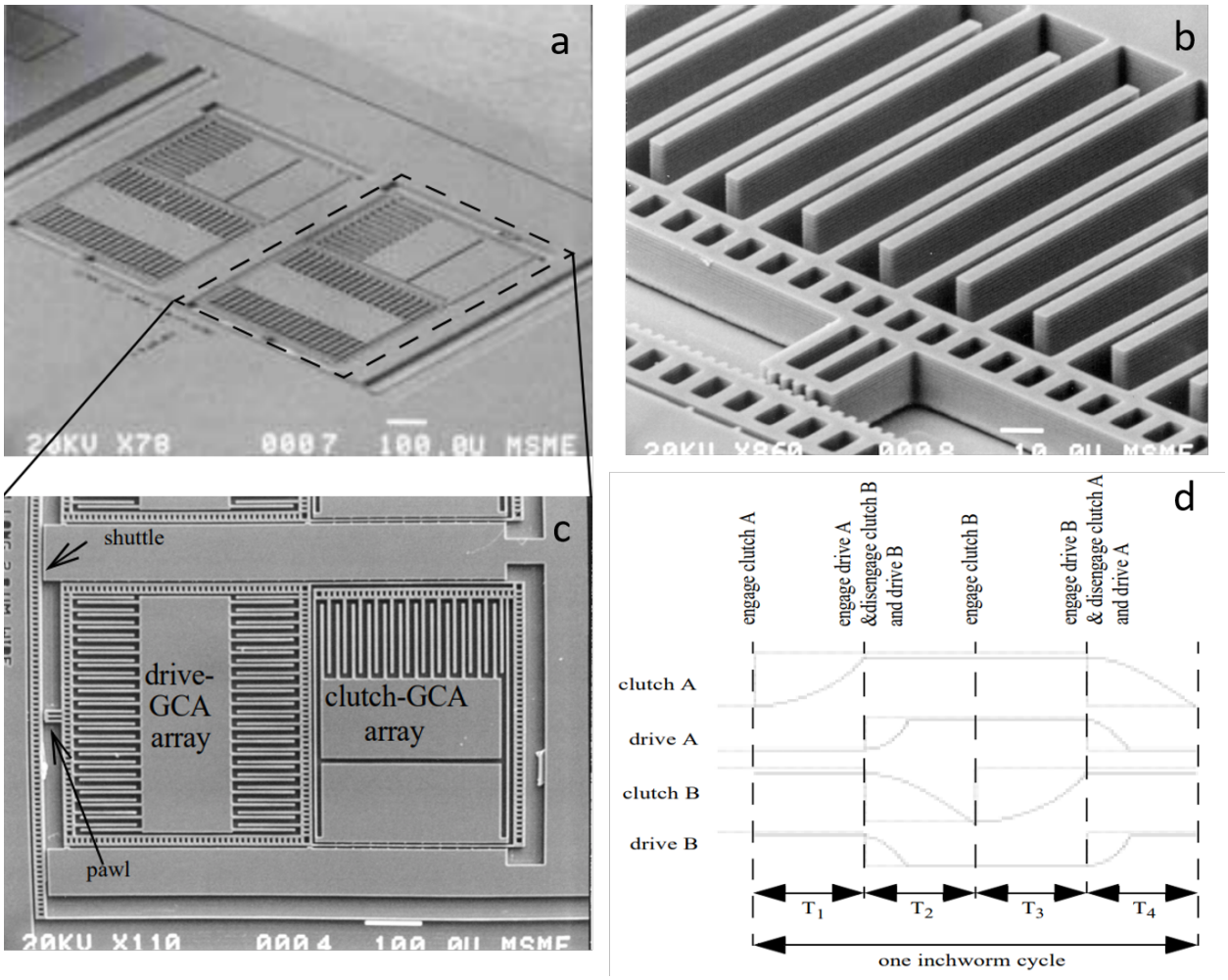

Figure 3.5: The first iteration of an electrostatic inchworm motor from 2001 designed by Yeh et al. [27]. a) An angled view of the gap closing actuators b) Zoomed in image of the shuttle, fingers and teeth c) Top down view with labeled components of one half of the motor d) The phased voltage signals that drive the motor through one full cycle of operation

for the inchworm motor. The main drawback to the GCA is its limited output stroke. It can generate high forces, but only over small distances. In Equation 3.35, x corresponds to the travel of the movable fingers. This travel is inherently limited to a maximum value of g_1 based on the geometry of the device. In practice, x is limited to $g_1 - 1\mu$m to keep the two sets of fingers from pulling in and touching each other. To increase the usable travel of this device, g_1 could be increased, but Equation 3.35 shows that the total force drops off significantly if g_1 is large. This limitation means that a GCA can apply a large force over a distance of about 2-3 μm. This is a long way off from the 2 mm requirement for the 10 cm jumping microrobot. Yeh et al. engineered the electrostatic inchworm motor that

comprised 4 individually addressable GCAs. When run in the proper sequence these GCAs could take many small steps that culminated in a large overall displacement, similar to how an inchworm moves from place to place. Figure 3.5a shows the entire inchworm motor with the four separate GCAs visible. The inchworm motor is made up of two identical halves, one of which is shown in Figure 3.5c. These two GCAs work together to engage the pawl, shown more clearly in Figure 3.5b, with the shuttle and move the shuttle down a few microns. The voltage profiles for the drive and clutch GCAs for both halves of the motor are shown in Figure 3.5d.

To show how these four GCAs work together to advance the shuttle, a cartoon of the inchworm motor in operation is shown in Figure 3.6. Initially all GCAs are at rest, 3.6a. The offset of the two pawls with the shuttle is critically important. Notice that the teeth on Pawl 1 are aligned to interlock with the teeth on the shuttle whereas the teeth on Pawl 2 are offset a half period. To begin, Clutch 1 is actuated and the pawl makes contact with the shuttle, 3.6b. Drive 1 is then actuated which moves the shuttle down one half step, 3.6c. Now the teeth on the shuttle are lined up so the teeth on Pawl 2 can engage. Clutch 2 is actuated and the teeth engage, 3.6d. Now Clutch 1 can release followed by Drive 1 being released. All of the GCAs are reset passively by a spring when the voltage across the fingers is removed, 3.6e. Finally, Drive 2 can be actuated, which moves the shuttle down an additional half step, 3.6f. Now the shuttle teeth are again aligned with the teeth on Pawl 1 and the entire process can be repeated until the shuttle has been displaced the desired amount. This motor output a force of 50 μN, and had a force density of 0.017 mN/mm^2. For this motor topology to generate the 50 mN required for the 10 cm jumping microrobot, nearly 3,000 mm^2 of actuator area would be required.

This inchworm motor design was improved upon by Penskiy and Bergbreiter with their angled arm electrostatic inchworm motor [28]. This new design removed the clutch GCA entirely which was beneficial in many ways. First, this design reduces the number of control signals required to run the motor from four to two. This is a huge advantage when using these motors in a microrobot because it reduces the number of wires and connections that need to be made. Each of these connections has a finite probability of failure so fewer connections gives a higher likelihood of a working microrobot. Secondly, this design reduces the overall layout area, which in turn increases the areal force density. When trying to generate the high forces required to make this microrobot jump, the higher the force density the better. This motor design also increases the electrical to mechanical efficiency by reducing the parasitic capacitance. The more efficient the motor design is the farther the microrobot will be able to move on a single charge of its batteries or in an environment where it cannot scavenge a lot of energy.

A fabricated angled arm inchworm motor is shown in Figure 3.7. Here, a set of two GCAs driven with a single voltage signal is used as both the drive and the clutch. In Figure 3.7, the right side GCAs, surrounded by the green dashed boxes, are actuated first. This brings both the top and bottom pawls into contact with the shuttle and then pushes the shuttle forward a half step. Now the left side GCAs, surrounded by the blue dashed boxes, are actuated. This pushes the shuttle forwards an additional half step. During this step, the voltage on the

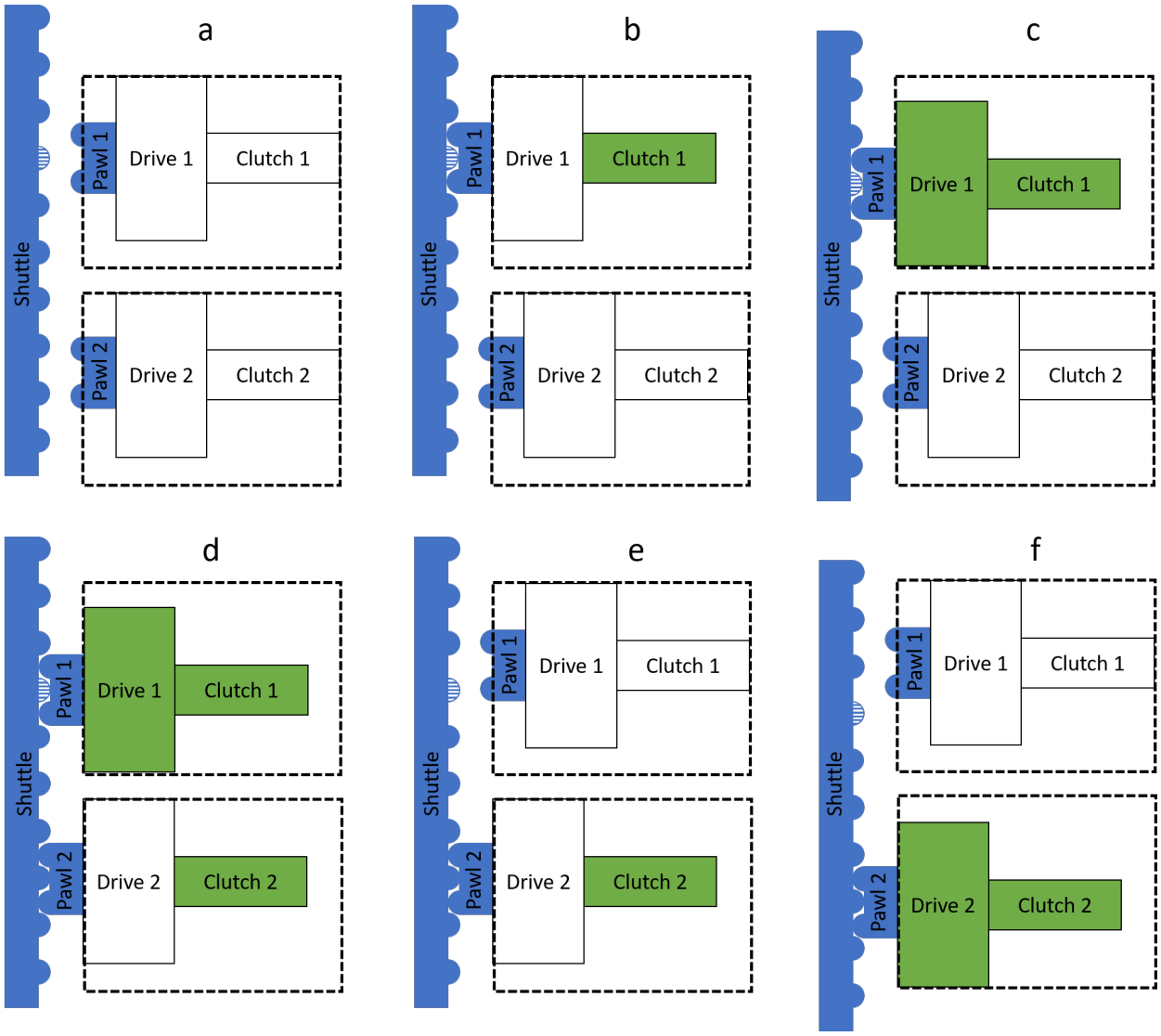

Figure 3.6: Cartoon of the original electrostatic inchworm motor operation. a) All GCAs are in their rest position b) Clutch 1 is actuated bringing Pawl 1 into contact with the shuttle c) Drive 1 actuates pulling the shuttle down a half step d) Clutch 2 is actuated bringing Pawl 2 into contact with the shuttle e) Clutch 1 is released, which releases Pawl 1 from the shuttle and Drive 1 is released f) Drive 2 is actuated, pushing the shuttle down an additional half step. Pawl 1 is now aligned with the shuttle teeth and can be engaged to start the next cycle, now one tooth above where it engaged earlier.

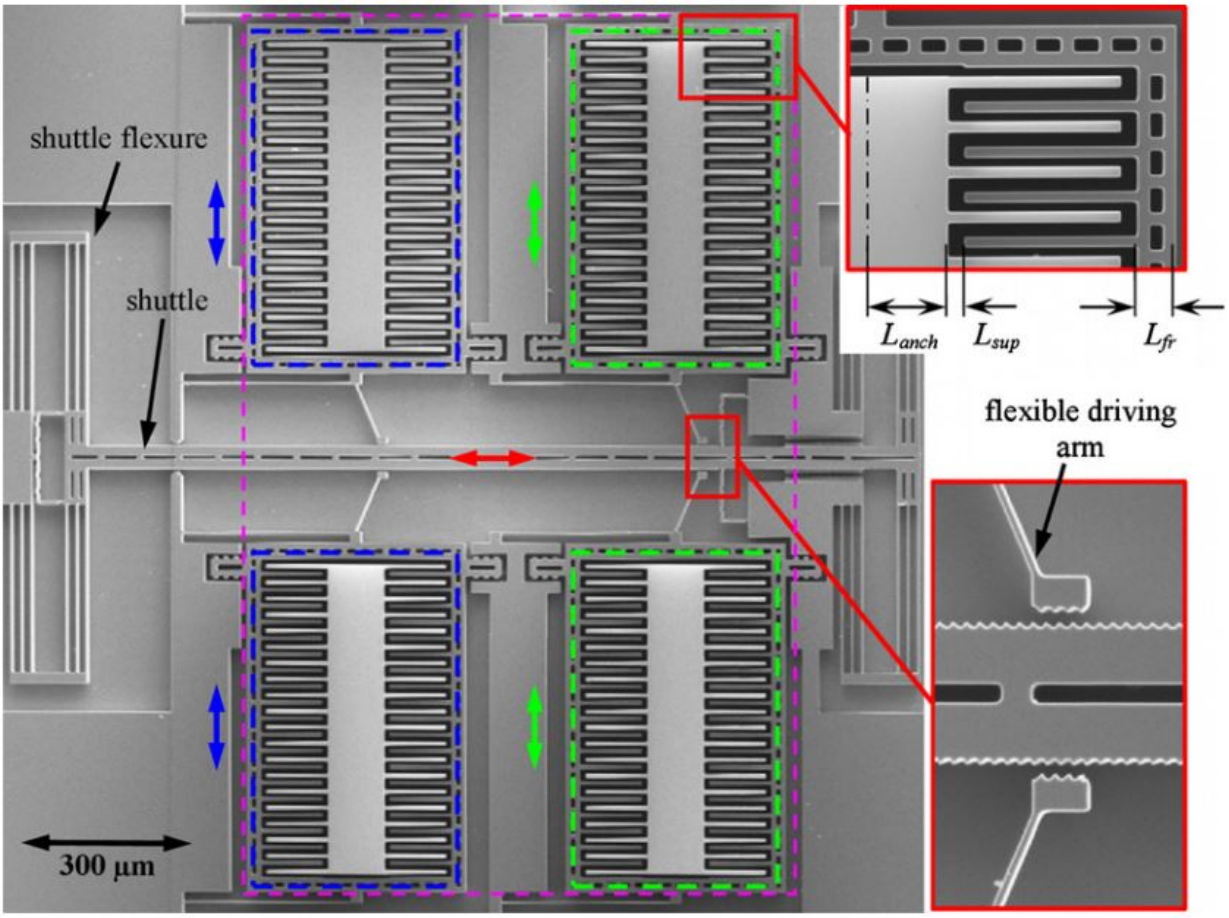

Figure 3.7: An angled arm electrostatic inchworm motor from [28]. This design only requires two control signals, one for the GCAs surrounded by the green box and one for the GCAs surrounded by the blue box.

right side GCAs is released and the pawls return to their initial position. These steps can be repeated as many times as necessary to reach the desired shuttle displacement. This motor has a force density of 1.38 mN/mm^2 at a voltage of 110. For this motor to generate the 50 mN required for the 10 cm jumping microrobot, about 36 mm^2 of actuator area would be required. While this is a vast improvement over the original electrostatic inchworm motor design, it still requires a great deal of layout area for the motors alone. Additionally, if a the design goal for the jumper were 20 cm or more, the actuator area would need to increase to a prohibitively large fraction of the total die size.

To address this issue, the force from these motors must be amplified in some way. The Penskiy angled arm inchworm motors were used as a base model from which the Pister group, specifically Daniel Contreras, made modifications to fit our fabrication process and wafer specifications. The initial idea for amplifying the force from these motors was to use

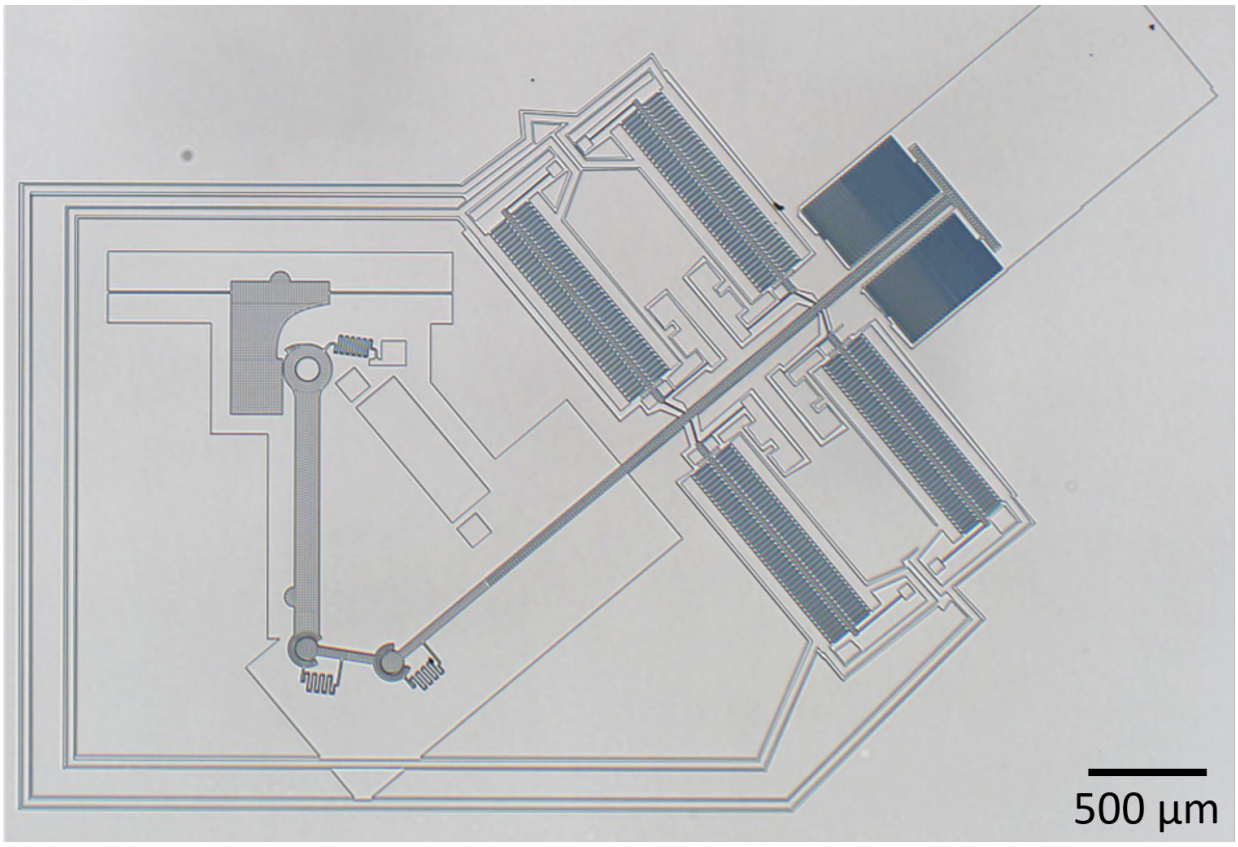

Figure 3.8: A fabricated inchworm motor attached to a lever arm driving a MEMS hammer.

the same mechanisms used for the MEMS hammer project with two pin joints to create a four-bar linkage. These pin joints, also designed by Daniel Contreras [29], are similar in function to the pin at the center of the lever arms discussed earlier, however the center of the joint can translate in plane as opposed to being fixed. A fabricated device is shown in Figure 3.8. The inchworm motor was supposed to push the shuttle which would in turn actuate the lever arm to the right and start loading mechanical energy into the beams of the MEMS hammer. The motor here could output 1 mN of force at 100 V, and with the mechanical advantage from the lever arm, this should have increased that force to 8 mN. This device was difficult to test and had a fatal flaw that would prohibit it from being used in a jumping microrobot. First, while testing the inchworm, the shuttle would often pop out of plane dislodging one or both pin joints. These joints did not have sufficient vertical stiffness to keep the joints in tact throughout the travel of the device. Additionally, this setup could only ever displace an energy storage element 30 - 50 μm due to the nature of the lever arm and pin system. To achieve the 2 mm necessary for the 10 cm jumping microrobot, some new force amplification system must be implemented.

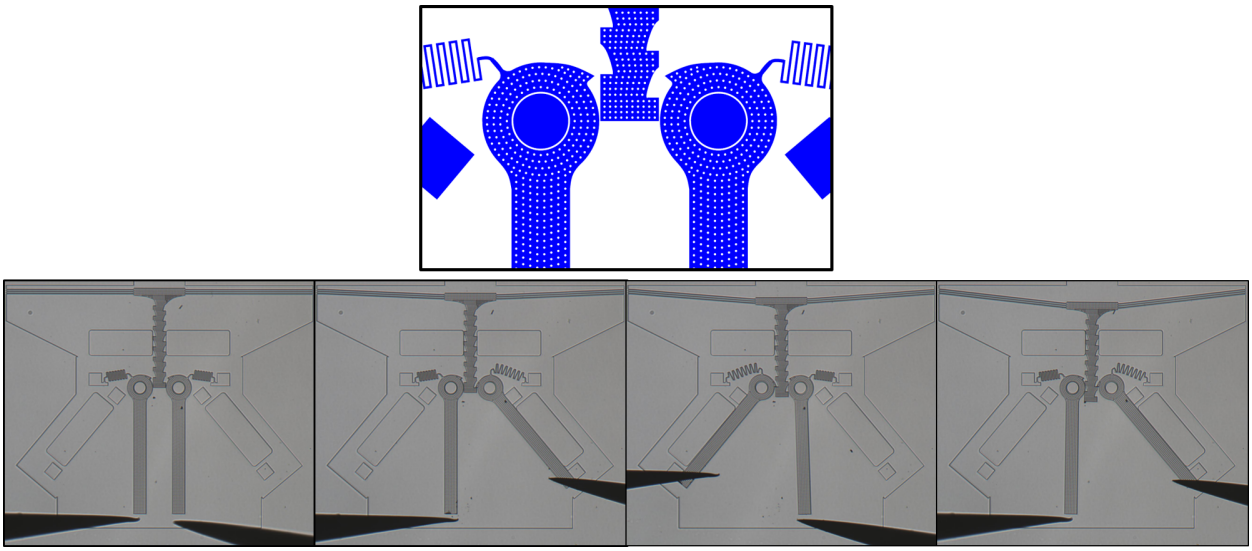

Figure 3.9: Two lever arms that are actuated manually to move a central shuttle. A zoomed in picture of the layout is shown above.

Inchworm of Inchworms

Just as the GCA is used as a building block for the inchworm motor, so too can the inchworm motor be used as a building block for an even higher force motor system. One of the issues with using a MEMS hammer lever arm to achieve this mechanical advantage is the small distance across which the amplified force can be applied. This problem can be addressed using the same fix Yeh et al. used back in 2001. Two sets of these devices can work together to inch along a central shuttle, with one actuator holding onto the shuttle while the other actuator resets. A manual proof of concept was designed and fabricated to test this idea. The device is shown in Figure 3.9. Two lever arms are adjacent to a central shuttle. The shuttle has cutouts on the left and right sides that are similar to those on the MEMS hammer. These cutouts are on a pitch of 80 μm and the left and right arrays are offset by 40 μm. This allows one side to engage and depress the shuttle to a point where the other side is aligned and ready to do the same. Starting at the bottom left most image in Figure 3.9, the two lever arms are at rest and the shuttle is not deflected. The right lever arm is moved to the right using a probe tip and the central shuttle is deflected 40 μm. Now the left lever arm can start to be moved to the left. Once it has made contact with the shuttle, the right lever arm is released and returns to its initial position. The left lever arm continues towards the left and eventually deflects the shuttle an additional 40 μm. Next the right lever arm is pushed towards the right and the whole process can be repeated. While this theoretically now allows for large displacements at these larger forces, there is still the issue of effectively applying the motor force to the end of the lever arm.

Using a four-bar linkage to attach the shuttle of the inchworm motor to the lever arm

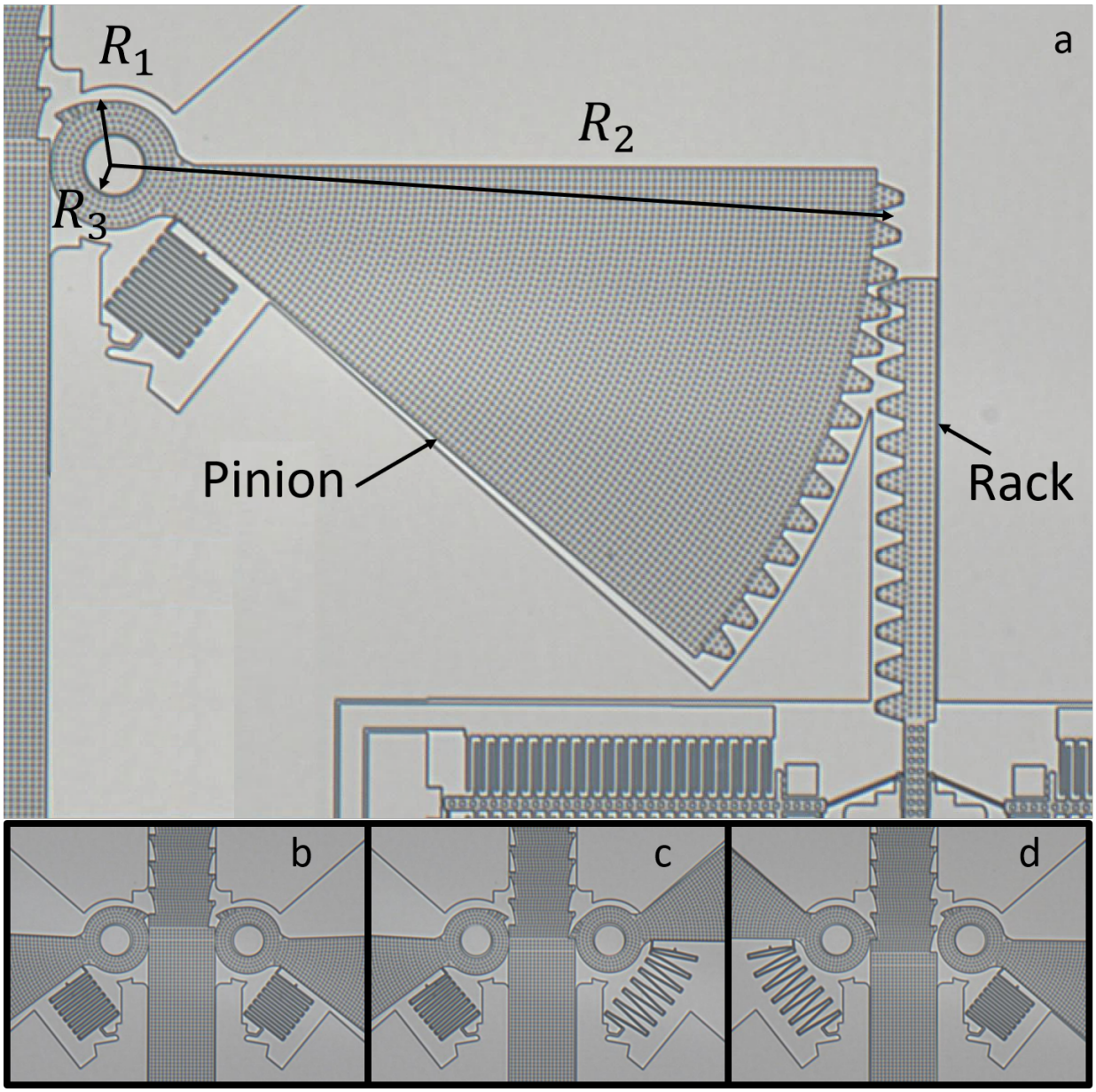

Figure 3.10: The rack and pinion system developed to amplify the force output of the electrostatic inchworm motor.

was problematic and prohibited the motor from successfully storing any mechanical energy. Additionally, even if the mechanics did work perfectly, as the motor pulled the lever arm, the angle between the lever arm and the shuttle changes. This means the mechanical advantage factor also changes. Ideally, this factor would remain constant throughout the entire travel of the lever arm. This can be achieved by creating a MEMS rack and pinion. This device,

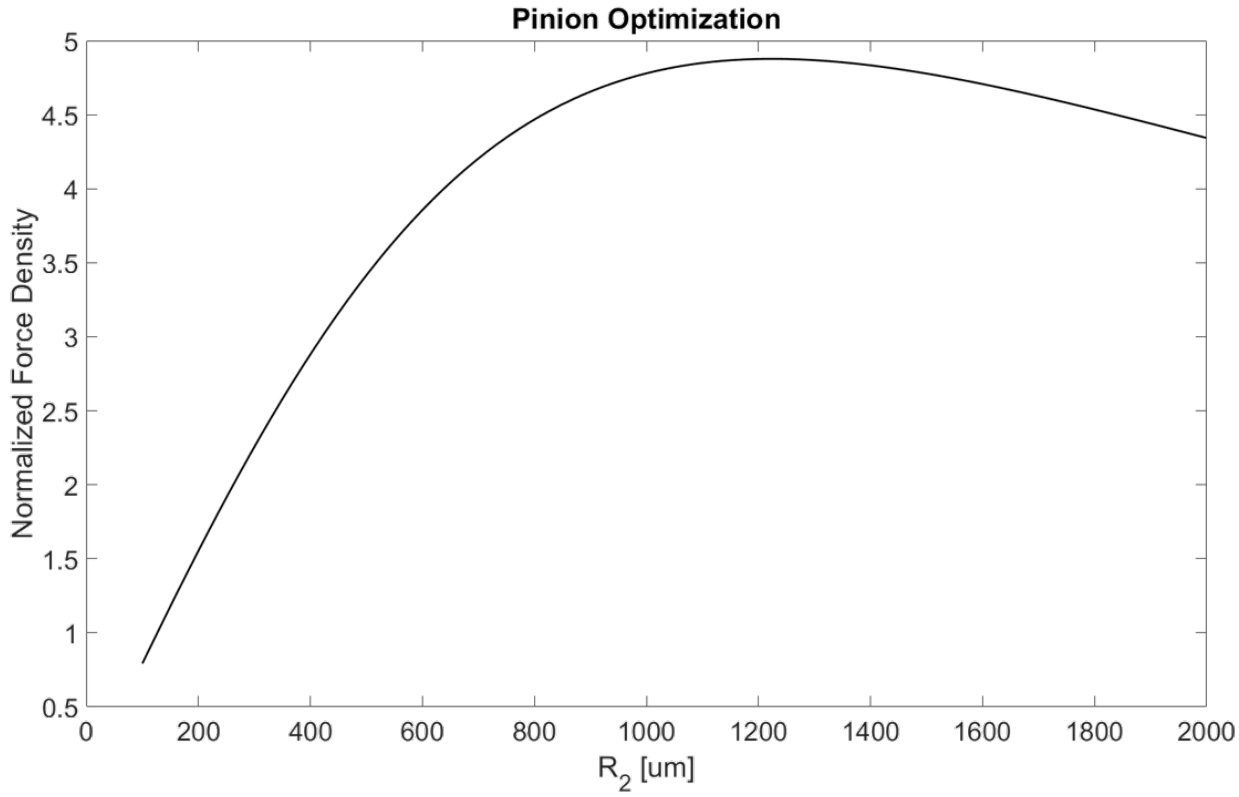

Figure 3.11: Plot of the force density of the inchworm of inchworms normalized to the force density of the single inchworm motor used to drive the pinions as a function of the pinion length, R_2. The maximum occurs at about 1200 μm where the force density of the system is over 4.5 times the density of the inchworm motor alone and has a mechanical advantage of 10.

shown in Figure 3.10a, uses a single stationary pin joint to translate and amplify the force from the inchworm motor on the right to the central shuttle on the left. The inchworm motor, partially visible in the bottom right of the figure, works to push the rack upwards. The involute teeth on the rack make contact with the pinion and rotate it about the central pin. This causes the latch to make contact with the central shuttle and deflect it. This setup allows for a constant mechanical advantage from the inchworm motor to the central shuttle.

This rack and pinion system is the core component of a new motor topology, the inchworm of inchworms (IoI). The IoI is composed of two identical sets of racks and pinions that are mirrored across the central shuttle. Figure 3.10 shows the first two steps of the IoI. Before the IoI begins operation, neither pinion is in contact with the central shuttle as shown in Figure 3.10b. Then the inchworm motor on the right side of the central shuttle takes enough micro steps, defined as one full cycle of the inchworm motor advancing the rack forward by about 2 μm, to push the pinion all the way to its backstop. This deflects the central shuttle

by 40 μm and aligns the cutout on the left side of the central shuttle with the latch on the pinion as shown in Figure 3.10c. The right side inchworm motor stops actuating, but the voltage is maintained on the GCAs keeping the right side pinion in place. The IoI has now taken one macro step. From here, the left side inchworm motor can begin taking micro steps until its pinon makes contact with the central shuttle. At this point the right side inchworm motor can release its GCAs and a serpentine spring returns both the rack and the pinion to their initial locations. The left side inchworm motor continues pushing the left side pinion up until the central shuttle has moved an additional 40 μm. Now the second macro step has been taken, shown in Figure 3.10d. This process can be repeated as many times as necessary to move the central shuttle the required distance.

As discussed in the previous chapter, R_1 has been minimized to ensure that the pin, defined by R_3, remains anchored to the substrate throughout device operation. The minimum value for R_1 is roughly 120 μm. To determine the appropriate value for R_2, the increase in mechanical advantage must be weighed against the increase in layout area used. Figure 3.11 shows the normalized force density of this motor as a function of the R_2 length of the pinion. The force density here was normalized to the force density of the single inchworm motor used to drive that rack and pinion. This plot shows that if a pinion length of 1200 μm is chosen then the IoI will have over 4.5 times the force density of the inchworm motor alone. Subsequently, with a pinion length of 1200 μm and a value for R_1 of 120 μm, the force from the inchworm motor is amplified by a factor of 10. The serpentine spring attached to the pinion does decrease this value, however it does so by less than 1% so the effects of this spring are ignored.

Other than having a high force density, the IoI is beneficial because it protects the most delicate parts of the electrostatic inchworm motors. One of the most common ways an inchworm motor fails is from a broken pawl or angled arm. These are narrow pieces of silicon that will easily fracture if a force is applied in an undesired direction which happens if the shuttle of the inchworm motor is bumped or jostled. The IoI will be used as the motor for the jumping microrobot in a following section and it will be shown that the central shuttle is attached to the foot from which the microrobot jumps. When the microrobot lands on this foot, it has a high likelihood of temporarily coming out of plane or at the very least being knocked off axis. If the inchworm motors were to directly push on this main shuttle, their pawls and angled arms would most certainly not survive more than a single jump before one or more of their angled arms broke off. By the nature of the actuator, these components must sit within a few microns of the shuttle they drive, and the impact of the foot with the ground could easily deflect the shuttle laterally more than a few microns. With the IoI, the pinions can completely disengage with the central shuttle during a jump. This removes the inchworm motors, and therefore the pawls and angled arms, from being in direct contact with the central shuttle as it comes back into contact with the ground after a jump. This leads to a jumping microrobot that can jump multiple times without destroying its on-board motor.

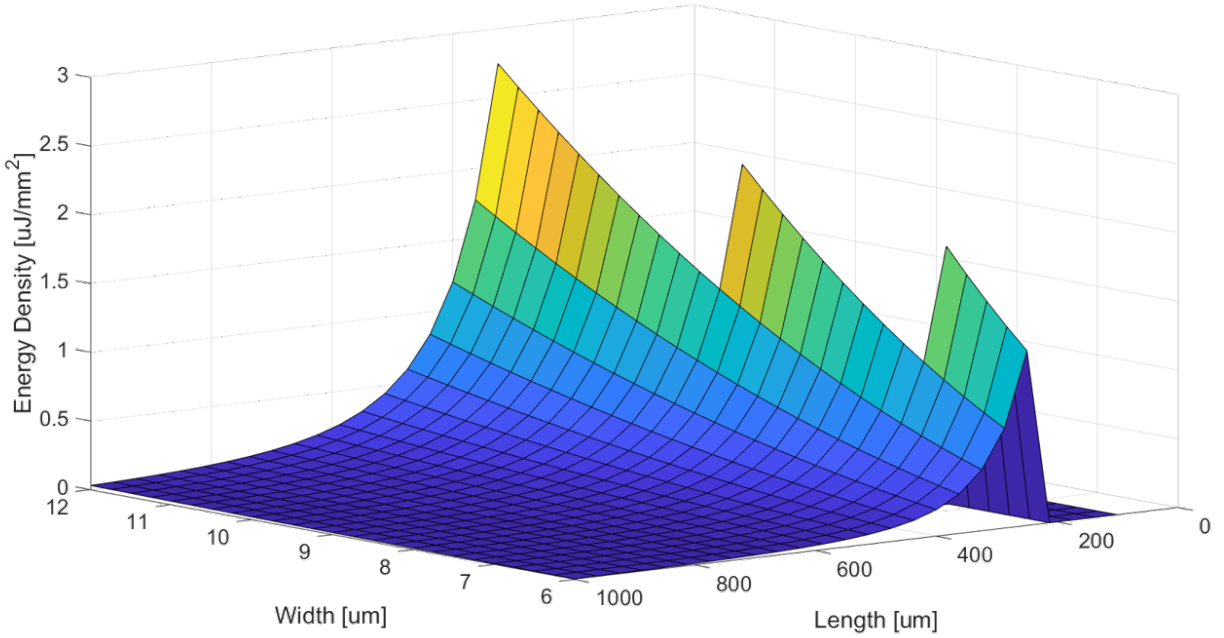

Figure 3.12: This plot shows the energy density of a serpentine spring across part of the fabrication space.

3.5 High Energy Density Springs

Just as it is important to maximize the force density of the motor, it is equally important to maximize the energy density of the springs that will be deflected by the motor. These springs must have an energy density sufficient to store the required energy to jump while not being too large to fit inside the layout area. The energy density of the serpentine springs discussed above is given by the following equation where E is the Young's modulus of silicon, w is the beam width, t is the silicon thickness, N is the number of meanders, Δx is the total deflection, and g is the gap between adjacent meanders:

$$U_{density} = \frac{1}{2}\frac{Ew^3t}{NL^4[N(w+g)+\Delta x]}\Delta x^2 \tag{3.36}$$

One important consideration for these energy density calculations is that the total area used must include both the spring layout area as well as the area into which the spring extends when fully deflected. The area into which the spring extends is unusable by other mechanisms and therefore cannot be ignored. Additionally, this optimization problem is underconstrained and difficult to optimize in closed form. However, it is possible to glean general design rules from Equation 3.36. Beams that are the most energy dense will be wide, have short lengths, and have large deflections. A MATLAB script was generated to sweep the variables across the design space taking into consideration processing limitations.

An example of the plotted output from this MATLAB script is shown in Figure 3.12. The jaggedness of this plot comes from the sharp nonlinearity of beam fracture. The MATLAB script zeros out all energy density values if the beam exceeds a maximum strain of 0.9%, because the beam will likely fracture at this value. The most energy dense spring that can be fabricated has a density of slightly more than 2.5 $\mu J/mm^2$, and as expected has a large width and a small length.

One major factor limiting the energy density of these serpentine springs is that the spring must expand into additional chip area as it stores energy. This can be fixed by again changing the type of spring used to store this mechanical energy. This time a coil spring is implemented. These springs have been used for hundreds of years by the Swiss watchmaking industry and can theoretically be far more energy dense than a serpentine spring. Muñoz-Guijosa et al. [30] developed a model for a coil spring that can be expressed as the following equation, where w is the spring width, t is the thickness, R is radius to the outer most ring of the spiral, L_{tot} is the total length of the spiral, and $\Delta\theta$ is the angular deflection:

$$F = \frac{Ew^3t}{12RL_{tot}}\Delta\theta \tag{3.37}$$

It follows that the maximum strain in this beam can be written as:

$$\epsilon_{max} = \frac{w}{2L_{tot}}\Delta\theta \tag{3.38}$$

Both Equations 3.38 and 3.37 can be written as a function of linear displacement Δx, if the substitution $\theta = \Delta x/R$ is made. This gives the following equations:

$$F = \frac{Ew^3t}{12R^2L_{tot}}\Delta x \tag{3.39}$$

$$\epsilon_{max} = \frac{w}{2RL_{tot}}\Delta x \tag{3.40}$$

Using Equations 3.39 and 3.40 a MATLAB script was generated to find the maximum theoretical energy density of these coil springs. Figure 3.13 shows that the maximum energy density here is over 40 $\mu J/mm^2$! This is a large theoretical improvement over the serpentine spring energy density, however the device must be fabricated and tested to determine the practical energy density of these coil springs.

A set of two coil springs was fabricated, shown in Figure 3.14, each has a width of 12 μm, a thickness of 40 μm, 10 μm of gap between each coil, and 8 spirals. These coils were designed with integration into the jumping microrobot in mind, and therefore additional features were added. Since the coils have a low translational stiffness in the plane of the wafer, an internal guide was added on each coil to keep its gear teeth from slipping with the central shuttle. This means that the radius of the outer most coil is no longer equal to the radius at which the force is being applied to the coil. That is to say, there is an additional ring of silicon extending past the last coil of the spring and it is at this radius that the force

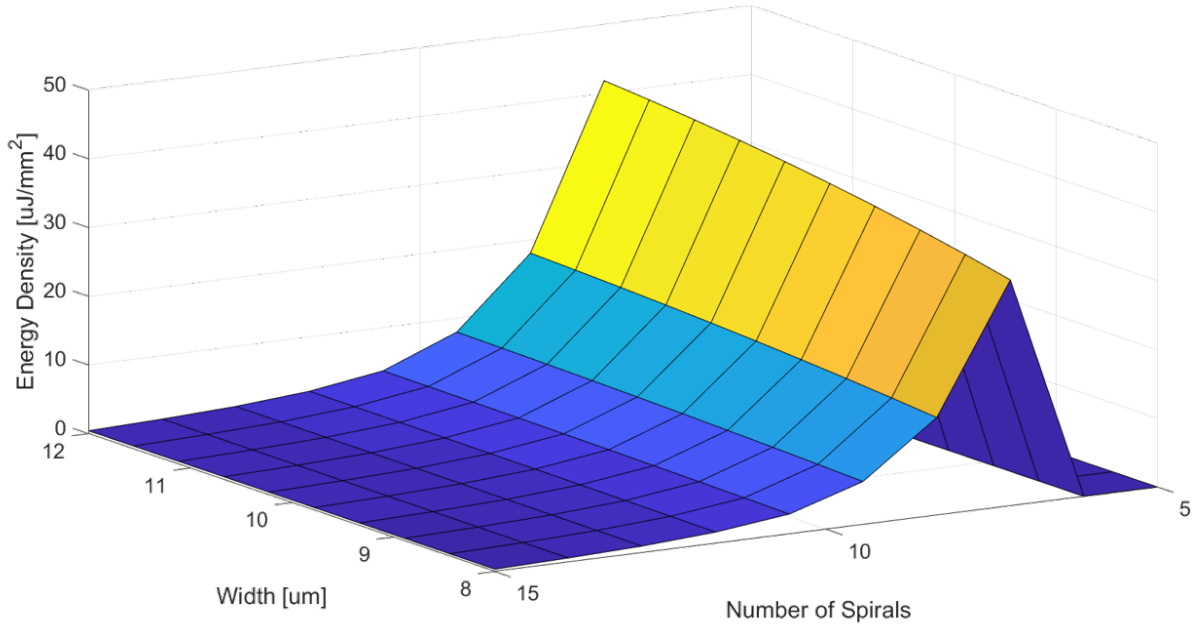

Figure 3.13: This plot shows the energy density of a coil spring across the fabrication space.

is applied to the coil. Equation 3.39 is updated accordingly to account for this addition to the R term. This change to the design decreases both the spring constant as well as the energy density of these devices. Additionally, the spring constant, and thus energy density, were kept intentionally low to maintain a maximum required force of less than 10 mN to fully actuate this shuttle. This should allow inchworm motors with a force output of 1 mN with a 10:1 mechanical advantage from the IoI structure to fully actuate these springs. Each of these coils theoretically has an energy density of 13.7 $\mu J/mm^2$, a spring constant of 1.7 N/m, and should store 1.68 μJ of mechanical energy. After testing with the Dage bond tester, it was determined that the energy density was 10.5 $\mu J/mm^2$, the spring constant 1.79 ± 0.10 N/m, and the energy stored 1.75 μJ. Although only one geometry of spiral spring was tested, it appears that the theory accurately predicts the performance of these springs. Although the spiral springs are considerably more energy dense than the serpentine springs there are issues with rapidly releasing that stored energy. The spiral springs are less reliable compared to the serpentine springs and often get stuck before they have fully unwound. With more work, the spiral springs certainly have the potential to overcome these barriers, but currently the highest density serpentine springs will be used for the jumping microrobot.

3.6 Inchworm Motor Driven Jumping Microrobot

Now that all the components of the jumping microrobot have been designed and tested individually, the entire microrobot can be fabricated and assembled. While the ultimate goal

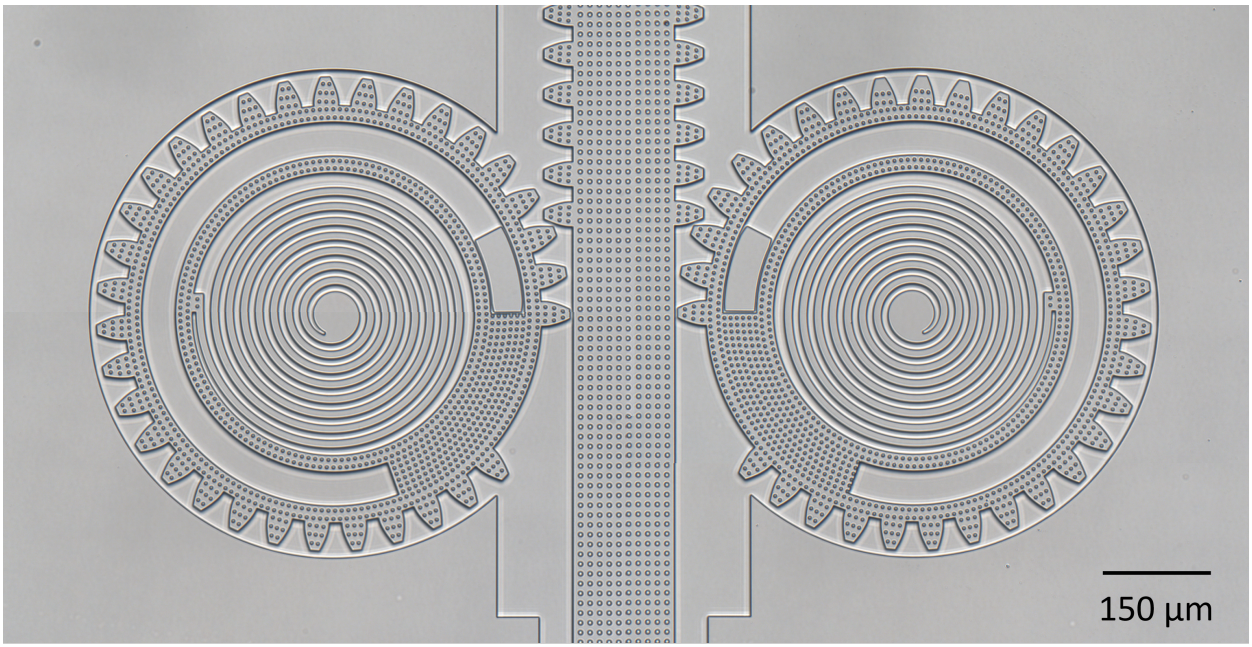

Figure 3.14: Two fabricated coil springs each with a 12 μm width, 40 μm thickness, 10 μm gap between successive coils, and 8 spirals in total.

of this project is to create a microrobot capable of jumping 10's of centimeters into the air multiple times per minute, here the goal was to integrate these components to determine if they will all work together. The prototype device, depicted in Figure 3.15, was fabricated in the two-mask SOI process with the addition of the hydrogen anneal step to increase the fracture strain of the silicon springs. Four serpentine springs were used to store the mechanical energy for this device. The springs were attached to the central shuttle which in turn was attached to a foot from which the microrobot will eventually jump. This foot is made of device layer silicon and there are two pillars of substrate silicon on the left and right side of the foot. In total, the springs could store 4.0 μJ of mechanical energy while requiring a force of 7.1 mN. The total deflection of these springs was 1120 μm. The springs were attached to the central shuttle of an IoI which could theoretically apply 10 mN of force at the central shuttle, amplified from 1 mN at the individual electrostatic inchworm motors. The routing was all done in the SOI layer to 5 pads at the top of the microrobot. With 4.0 μJ of energy stored, this microrobot should theoretically be able to jump 1.0 cm high when it jumps straight up, according to Equations 3.8 and 3.9.

It was noted previously that the maximum deflection of these serpentine springs is bounded to an upper limit of 2 mm. While this is not a hard limit, this maximum deflection directly corresponds to how far off the edge of the chip the foot must extend. Figure 3.15 shows the central shuttle and the foot as seen through the backside of the wafer directly after the back side DRIE step is completed. Initially, during this etch step the buried

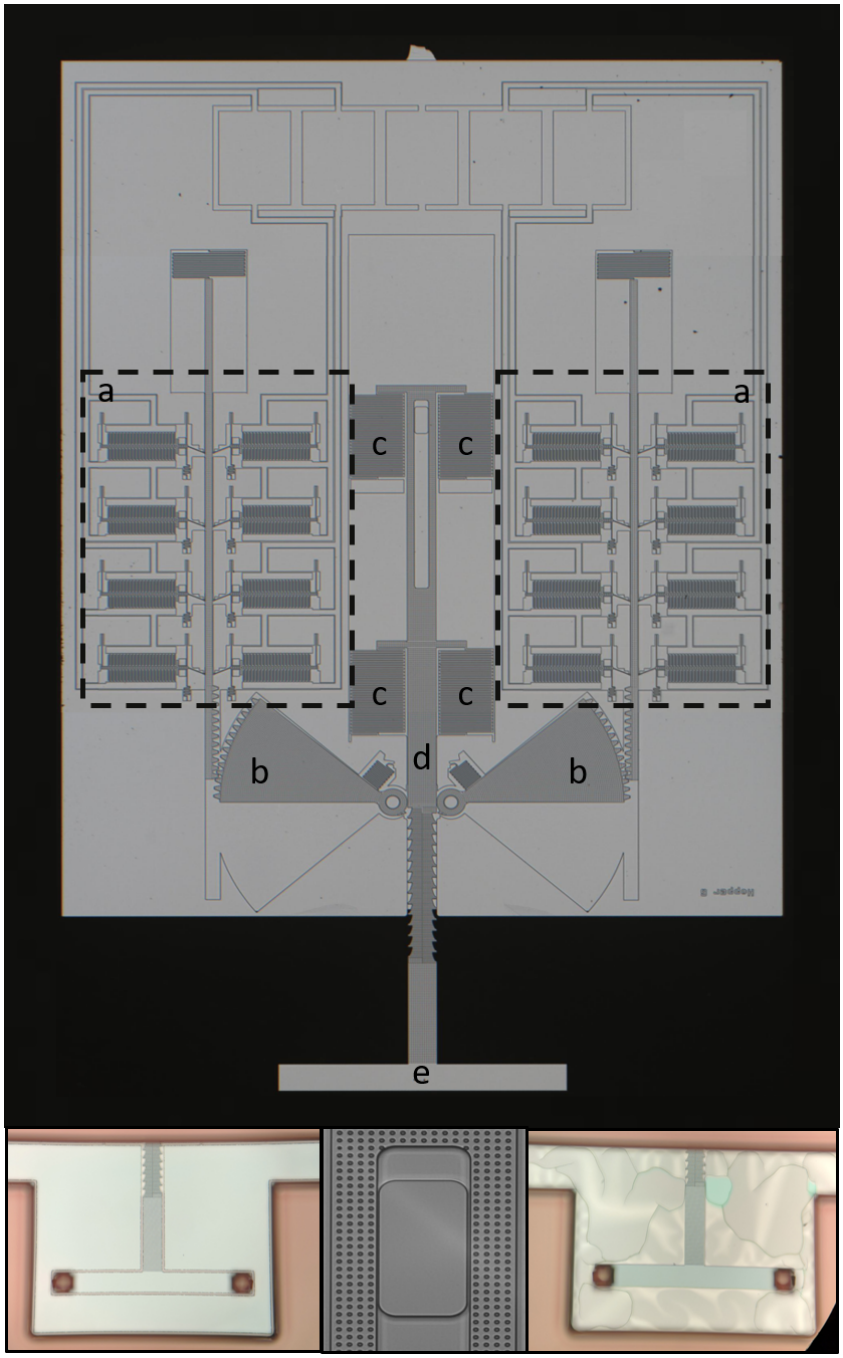

Figure 3.15: The 5.0 x 6.4 mm^2 jumping microrobot. Top: a) Electrostatic inchworm motors b) Pinions c) Energy storing serpentine springs d) Central shuttle e) Foot. Bottom: Microrobot foot immediately after processing with (left) and without (right) an SOI protective screen in the field. The bottom center image shows the details of the central guide of the central shuttle.

oxide would crack and fracture in areas where both the handle silicon and device silicon were etched, as seen in the bottom right image of this figure. When this occurs the etch gases can attack the protective oxide on the device layer and eventually start attacking the underlying silicon as well. This problem gets worse the longer the central shuttle becomes. An additional layout feature was added to these large areas to help mitigate this problem. Shown in the bottom left image of Figure 3.15, device layer silicon was left over these large areas where the handle silicon was etched. A 10 μm gap is left around the edges of other device layer features. This keeps the small areas of exposed oxide from breaking during the backside etch and therefore stops the etch gases from attacking the device layer features. Although this protective silicon does help create longer deflection shuttles, it can not do so indefinitely. Once the shuttles exceed 2 mm of extension off the edge of the chip, they break often when being removed from the wafer.

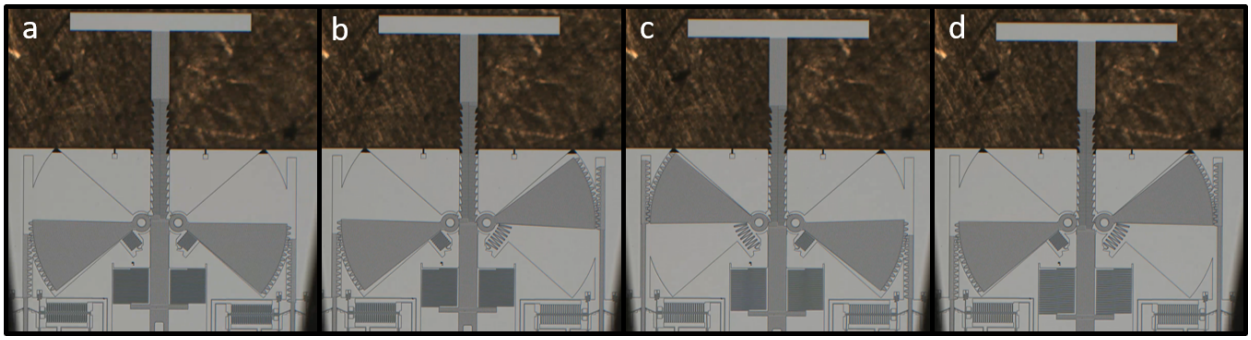

Figure 3.16: The 5.0 x 6.4 mm^2 jumping microrobot under a probe station. Each macro step of the IoI deflects the main shuttle by 40 μm. a) The device as fabricated b) After the first macro step of the IoI c) After the second macro step of the IoI d) After the third macro step of the IoI

To verify the operation of the motors and mechanisms, the microrobot was inspected at a probe station. While this obviously would not allow the microrobot to jump, it did allow for easy testing of each component of the design. Five probes were landed on each of the 5 bond pads. The five signals required are one ground signal shared between the two sides of the IoI, and two control signals for each inchworm motor. Tests were performed on the microrobot with motor control signals ranging between 60 and 100 V. These signals were sent to the motors through high voltage transistors with pull-up resistors. A microcontroller was programmed with the desired stepping pattern and controlled the gates of these high voltage transistors. A benchtop power supply controlled the 60-100 V drive voltage. The voltage range corresponded to a measured force output of between 1.0 and 2.5 mN as measured by the deflection of the serpentine springs. The chip can be seen in Figure 3.16 as the IoI is run under the probe station. Each macro step pulls the shuttle and foot down by 40 μm. The pinions have no trouble returning to their rest positions as soon as the voltage from their respective motors is released. With 60 V applied to the motors, the microrobot could take

28 macro steps, pulling the central shuttle down a full 1120 μm, storing 4.0 μJ of energy and requiring 7.1 mN of force. We can confirm from this that the force amplification through the pinion structure is at least 7:1. In initial designs of the jumping microrobot, rotation of the main shuttle would limit its total deflection because the shuttle would scrape against various sidewalls and become impossible to advance. The addition of a central guide, shown between the top two serpentine springs in Figure 3.15, helped fix this problem by constraining the rotation of the central shuttle.

Figure 3.17: An array of lines printed onto a piece of paper used to accurately space out 60 μm copper wires onto a piece of Kapton tape.

Now that the basic operation has been tested, the microrobot was prepared to take its first leaps. To allow the microrobot to jump, the power and control signals must be sent through compliant connections similar to the MEMS hammer and the passive jumping microrobot. The same silver epoxy and 60 μm copper wires was used to make a flexible electrical connection to the device, however the increase in number of connections from 2 to 5 required a new technique to align the wires to the bond pads. As shown in Figure 3.15, the bond pads are equally spaced and adjacent to each other. An array of lines with equal spacing was printed on a piece of paper and a piece of double sided Kapton tape was applied over the array of lines. From here, the individual copper wires were aligned to the array and held in place by the tape. Each of the 5 wires was aligned and the ends were trimmed to equal length, as shown in Figure 3.17. The tape was then peeled from the paper and adhered to the movable part of a micromanipulator. This allowed the wires to be carefully aligned to the pads under a stereo microscope. A drop of silver epoxy was applied to the pads before the wires were touched down, and depending on the viscosity of the epoxy (this changes with the amount of time it has been exposed to air) a drop may also be applied to the dangling wires themselves. Generally, if the epoxy is new and its viscosity is low, the epoxy readily wicks down the sidewalls of SOI structures, so the epoxy should be used more sparingly. However if the epoxy is older, it tends not to wick as much, so more epoxy can be used without the fear of devices being shorted to the substrate. Once the wires were aligned and placed onto the epoxy covered bond pads, the hotplate was turned on to 150 C for 30

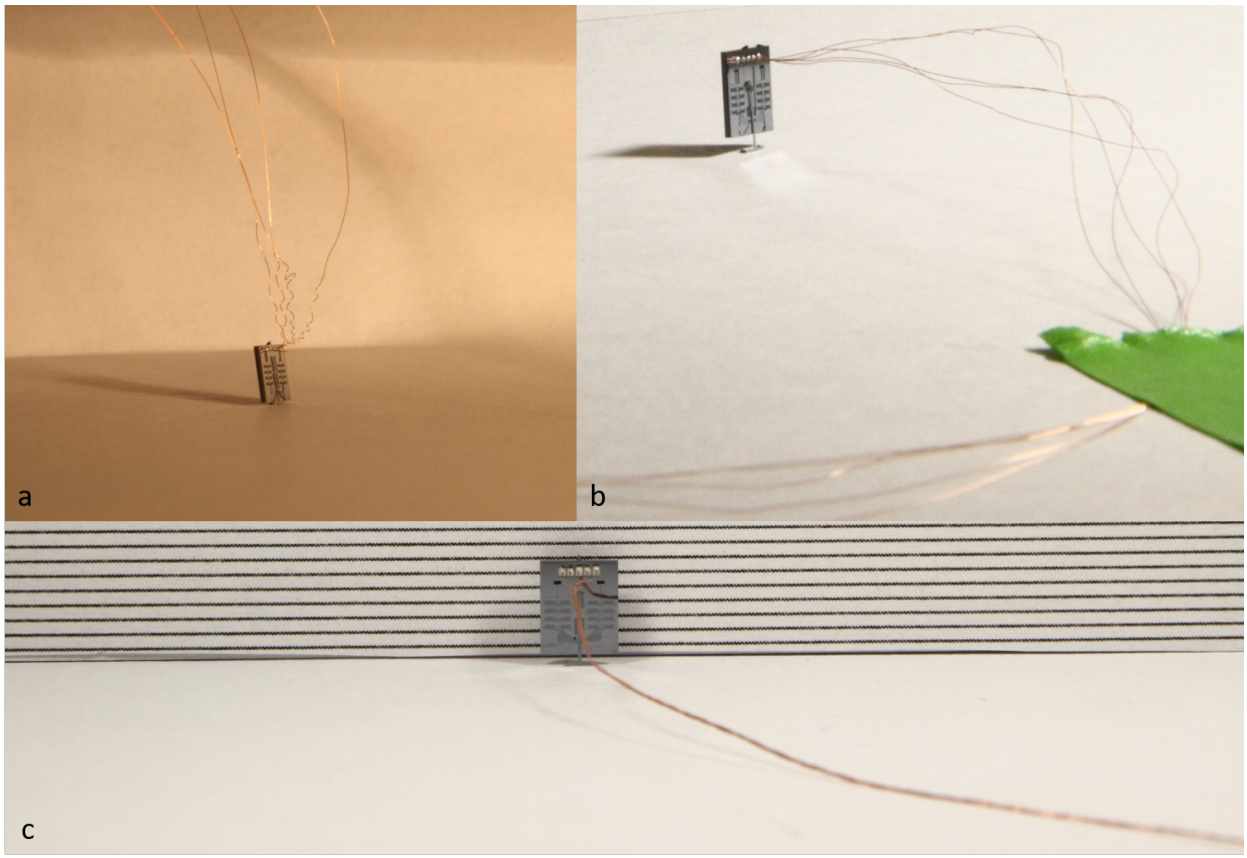

Figure 3.18: Various degrees of success in wiring the jumping microrobot a) Copper wires were positioned vertically above the microrobot b) Wires protruding horizontally from the microrobot c) Wires braided together protruding horizontally from the microrobot

minutes so the epoxy can cure. Once this is done the wires were carefully peeled from the tape and microrobot is ready to be tested.

These wires made testing this jumping microrobot tedious and difficult. Initially the wires were positioned vertically over the microrobot as in Figure 3.18a. The copper wires are stiffer than the serpentine springs of the microrobot, by a factor of about 10, and restricted any vertical movement once the motors released. To solve this, the wires were pulled perpendicular to the device side of the microrobot. This way, the wires act as a stabilizing cantilever and should allow vertical movement of the device. The ends of the wires were taped to the testing surface as shown in Figure 3.18b. While this did not inhibit vertical movement of the microrobot, the kinks in the wires constantly applied torques to the microrobot and pulled its foot out of plane. Fortunately, this would not break the microrobot, but it did make it impossible to test. The pinions can not make contact with the central shuttle unless the shuttle is sitting in the alignment groove. The positive news was that the copper wires made good electrical connection to the motors. The pinions could be driven

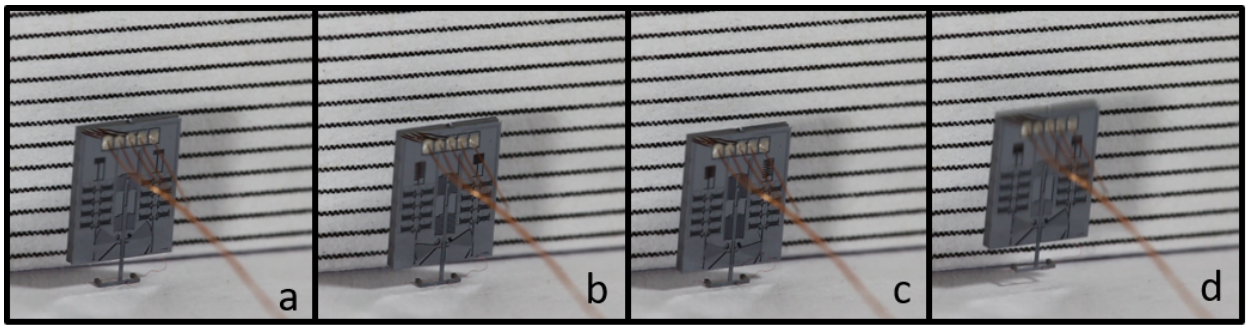

Figure 3.19: Frames taken from a video of the jumping microrobot as it loads its serpentine springs and jumps 1.0 mm high. a) The initial position of the microrobot b) The microrobot after 7 macro steps c) The microrobot after 14 macro steps d) The microrobot after release, 1 mm off the ground

with voltages applied to the free ends of the copper wires.

To remove these bends and kinks in the wires, all five wires were braided together before being bonded to the microrobot. After the epoxy was cured, the braided strand could be pulled manually with tweezers to straighten out all wires at once. This worked exceptionally well and is shown in Figure 3.18c. Roughly 15 cm of braided wire was used as slack before the bundle was taped down to the lab bench. One concern was that the enamel on these wires would not be strong enough to prevent leakage or breakdown between wires, but no breakdown events occurred. The microrobot is now ready to test.

To test the microrobot, it was placed on its foot in front of a backdrop that had an array of lines with a 1 mm pitch. The IoI is actuated using a drive voltage of 100 V. This iteration of the jumping microrobot has the ability to take 28 macro steps and store 4.0 μJ of mechanical energy, however it took many assembly attempts to get an assembled microrobot that could take more than one or two macro steps. During the IoI operation, when the inchworm motors on one side of the microrobot released their hold of the rack, it would remain in place instead of being pulled back to its rest position by the serpentine spring at the end of the inchworm shuttle attached to the rack. This is not behavior that was seen during the tests at the probe station. One possible explanation for this sticking is that some gaseous byproduct of the epoxy reaction sticks to the silicon surfaces and causes movable structures to be more likely to stick down to the substrate. To attempt to fix this, a soldering fume extractor was placed next to the microrobot during the curing process. This microrobot still suffered from some sticking issues, but it happened less frequently. This microrobot took 14 macro steps, deflecting the central shuttle 560 μm, and storing 1.0 μJ of mechanical energy all using its on-board motors. When the central shuttle was released after the 14^{th} step, the microrobot jumped at least 1.0 mm high. Theoretically it should have jumped about 2.5 mm, not taking into consideration any effect from the wires. The frames of the video recorded during this test can be seen in Figure 3.19. As the microrobot

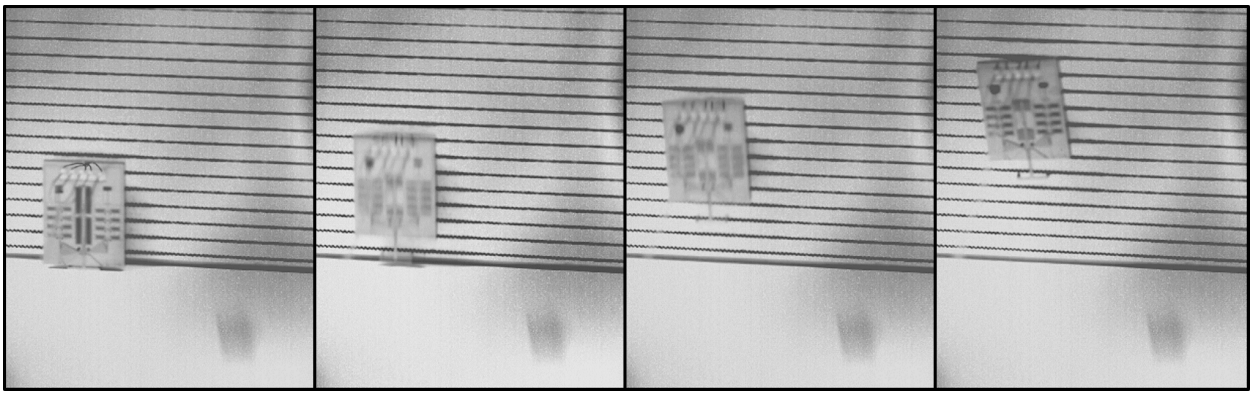

Figure 3.20: Frames taken from a 300 FPS video of the jumping microrobot after it was manually loaded to store 4.0 μJ of energy. The microrobot jumped 6.5 mm.

takes more and more macro steps, it pulls down its body down as in Figure 3.19b and 3.19c. This video was recorded at 60 FPS so it was difficult to capture the microrobot at the peak of its jump. In fact there were only 2 frames in the video in which the microrobot was in the air, so it is possible that the microrobot did jump higher than 1 mm.

The individual inchworm motors take micro steps at a rate of 80 hz and operate at 100 V. The total capacitance that must be driven during each micro step is 21 pF, which corresponds to an energy of 0.21 μJ per micro step. The energy required per macro step is then 94.5 μJ. If all 28 macro steps were taken, 2646 μJ of electrical input energy is required to store 4.0 μJ of mechanical energy. Currently it takes the microrobot 5.6 seconds to complete each macro step, meaning it can theoretically jump once every 2.5 minutes. This motor speed is not a fundamental limit, the motors are run at this relatively low speed to reduce the likelihood of the pinions sticking to the substrate. These electrostatic inchworm motors have been shown to move at speeds up to 3.4 cm/s [31], which could theoretically lead to a jump rate of 1.3 jumps per second!

While this result of a 1.0 mm jump is a long way from the desired goal of jumping 10's of centimeters, it shows the first time a MEMS jumping microrobot has used onboard motors to store mechanical energy and jump. An additional test was performed to see how high this microrobot could jump had it been able to actuate all 28 macro steps and store 4.0 μJ of energy. To do this, a micromanipulator was used to depress the microrobot body fully. After this, one side of the IoI was actuated electrically to engage the pinion with the central shuttle. From here, as soon as the voltage on the inchworm motors was released, the rack and pinion are both pulled back and the energy in the serpentine springs is released. Shown in Figure 3.20, a high frame rate camera was used to record a video of this release at 300 FPS. Upon release, the microrobot jumps 6.5 mm high. Theoretically it should have reached a height of 10 mm, excluding any effect from the tethers. The braided tether wires have a mass of 0.143 mg/mm, and using the measured stiffness of 2.6 mN/m, it was determined that approximately 65 mm of wire were lifted from the surface at the peak of the jump. The

effective mass of the lifted portion of the wires comes to 4.6 mg. This reduces the theoretical jump height to 9.1 mm, giving a jump efficiency of 71%.

This microrobot was actuated many times both manually and with its on-board motors storing the mechanical energy. Interestingly, the microrobot landed on its foot every time it jumped and could be actuated multiples times in a row without needing any manual input. Although this was heavily influenced by the braided wires, it is promising that the central shuttle remains in place even after impact with the ground. After the initial jump, the microrobot would often bounce on its foot, and with each bounce the central shuttle remained in place.

Chapter 4

Microrobot Design Library

Throughout the course of this work many different MEMS devices were designed, laid out, fabricated, and tested. Generally this process begins with an idea of what the device looks like and some fundamental equations that describe the device performance. From here, a set of device parameters is decided upon and the layout for those devices is completed. The mask file, typically in the Graphic Database System (GDS) format, which contains this layout is often shown at a design review where colleagues look at the designs and provide feedback. After modification and corrections are implemented, the GDS file is turned into a photomask and the devices can be fabricated and tested. In a perfect world, no layout mistakes would be made, the fundamental equations would exactly predict device behavior, and all devices would need to be fabricated only once to get publishable results. Obviously this is not the case, and multiple iterations of this process typically take place before usable devices are produced.

In the arena of integrated circuits, there are software packages in place to mitigate these errors. When a new circuit is designed, it is first simulated at the schematic level. From here the circuit is laid out and additional simulations can be performed taking into account the parasitics introduced by the layout. Additionally, a verification step is done to ensure that the layout and the schematic match. This layout versus schematic (LVS) verification is a powerful tool that greatly increases the likelihood of the physical post-fabrication circuit matching the performance given by the schematic simulations. Of course there are other factors that can lead to unexpected performance in the actual chip, but the simulation and LVS verification are crucial to any successful IC tapeout.

While there is certainly simulation software that exists for MEMS, it is often tailored towards specific and well funded fields within the community. Accelerometers [32], bulk acoustic wave resonators [33], gyroscopes [34], and microphones [35] all have excellent simulation packages. These devices are core products of many companies and it was critical that they could be well simulated before fabrication. SoftMEMS [36] is an excellent example of this type of software. This software package is capable of modeling both MEMS and IC components to ensure successful integration of the two. Analyses can be done using finite element analysis (FEA) or through nodal analysis. FEA packages are a good example of

general simulation tools for the MEMS community. This software handles mechanical deformations of bending and stretching beams, but if the structure becomes too complicated a general FEA may not accurately describe the deformation. Additionally FEA is very resource and time intensive, so running an FEA simulation on a complicated structure can be difficult and in some cases effectively impossible. Modeling the contact between two features in FEA is also difficult. Nodal analysis is generally much faster than FEA and can be highly accurate in describing device behavior for common MEMS devices. SoftMEMS for example has models for MEMS dog-bone resonators that have accurately predicted post-fabrication device behavior in the Pister group. Nodal analysis becomes less useful when new types of devices need to be simulated. If the underlying models do not accurately describe the underlying physics, nodal analysis will not be able to predict device behavior. SUGAR [37] is a software package that uses nodal analysis to analyze circuits, mechanical beams, electrostatic gaps, and more. While it handles linear beam theory well, it struggles with non-linear beams and cannot model contact between objects.

To help solve these issues and limit the number of iterations of the design-fabricate-test cycle, a MATLAB library was generated. This library is composed of many MATLAB functions that can generate layout structures as simple as a rectangle and as complicated as an entire jumping microrobot. While a library of standard shapes and objects is functionality that exists in other layout packages, this MATLAB library is different in a few key ways. First of all, embedding the layout library into MATLAB allows for calculations and layout to be done in the same script. MATLAB is a powerful computation tool, so high level inputs can be defined for a function. Instead of giving a serpentine spring its exact dimensions for how many meanders, beam widths, etc, a total layout area can be specified and MATLAB can generate a spring that is optimized for a particular characteristic such as energy density. Additionally, this library allows the user to script parts of or the entire layout. This is a powerful tool especially when designs have many interconnected components. It becomes trivial to double the length or width of a component by changing a parameter, recompiling, and seeing the surrounding structures update accordingly. This is faster and easier than manually changing that length or width and manually adjusting the surrounding features. This library is also written in a well known and easy to learn programming language. Many engineers already are familiar with MATLAB so the barrier to entry is low. Users can easily create their own structures and functions based on the primitive functions provided without needing to know anything about the library's backend.

Programmatically generating the layout has an additional benefit in that a simulation file can be automatically generated at the same time. When a high level layout function is used, such as a motor or a joint, the operation of that component is well understood: a motor will apply a linear force and a joint will allow some degree of rotation. This behavior can be programmed into the simulation file and provide constraints to the system as a whole. The particular simulation environment was chosen for its ability to model a large number of objects as well as physical contact between those objects. The environment used is a solid body physics simulator made specifically for macro scale robotics called Virtual Robot Experimentation Platform (VREP) [38]. While this is not the perfect platform for these

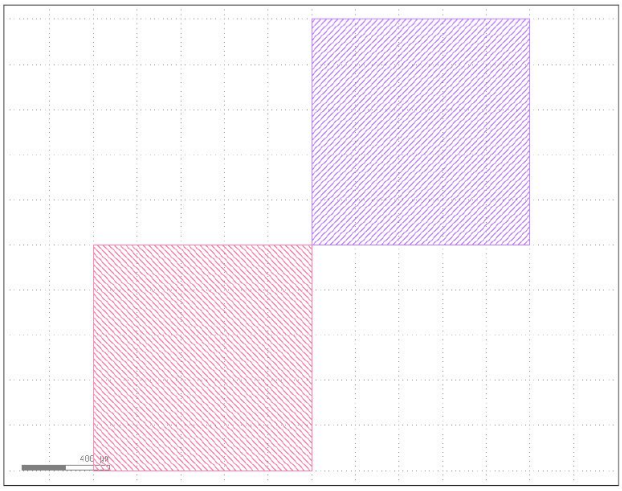

Figure 4.1: The layout file, 'basic,gds', that is output by the MATLAB script shown in Listing 4.1.

microrobot simulations, it does allow for some verification of the mechanical movement and interactions of the various MEMS devices. One main drawback to VREP is that it cannot simulate bending or stretching of components, however some of this functionality can be supplemented in software.

One final benefit of this library is that it enables a simple and thorough transfer of knowledge between cohorts as well as between different research groups. Too often, the institutional knowledge built up by one group of students is lost in transition to the next. The next generation of students will be able to pick up where this group left off with the exact designs and scripts used to create these microrobots.

4.1 MATLAB to GDS

While this MATLAB library is one single library, the main functionality can be separated into the generation of GDS files and the generation of VREP files. In this section the code that generates GDS files will be outlined and explained. Additionally guidelines are given for creating a new layout function to be added to the library.

GDSII Toolbox

The MATLAB library developed here is built using a MATLAB toolbox created by Ulf Griesmann [39]. This toolbox contains general functions that allow MATLAB to write a GDS file. The GDS file is in a binary format that, although being the industry standard, is poorly documented. This toolbox contains functions that can write primitive shapes to a

library and then write that library to a GDS file. An example of this is shown in Listing 4.1 and Figure 4.1.

```
1 % create a structure to hold elements
2 gs = gds_structure('BASIC');
3
4 % create two closed polygons
5 xy1 = 1000 * [0,0; 0,1; 1,1; 1,0; 0,0]; % 1mm x 1mm
6 xy2 = bsxfun(@plus, xy1, [1000, 1000]);
7
8 % create boundary elements and add to the structure (on different layers)
9 gs(end+1) = gds_element('boundary', 'xy',xy1, 'layer',1);
10 gs(end+1) = gds_element('boundary', 'xy',xy2, 'layer',2);
11
12 % create a library to hold the structure and add the structure
13 glib = gds_library('TWO_BLOCKS', 'uunit',1e-6, 'dbunit',1e-9, gs);
14
15 % finally write the library to a file
16 write_gds_library(glib, 'basic.gds');
```

Listing 4.1: Ulf's example MATLAB script that creates two rectangles and writes them to a file called 'basic.gds'.

After the GDS toolbox is installed, Listing 4.1 shows all the required code to generate a GDS file that contains two rectangles. Going through this code line by line, the code begins by defining a GDS Structure that will contain the polygons we will define later, line 2. Lines 5 and 6 define these polygons. The polygons are made up of a matrix were each row is a set of (x,y) points. This matrix can have any number of rows, but only two columns. We also see an example of a low level function capable of shifting all of the points in a matrix by a set amount, 'bsxfun'. Following this, these two matrices are used to create a GDS Element, lines 9 and 10. A GDS Element is a single enclosed shape, typically a boundary or a path. It also has a layer property. This corresponds to which layer in the GDS file this polygon will show up on. It should be noted that the GDS Structure defined earlier, 'gs', is in fact a structure. This means the 'end' keyword can be used to index the last element in that structure. It is useful to append a new GDS Element to a GDS Structure by using the index 'end + 1'. Next, in line 13, a GDS Library is created that contains the GDS Structure. This library can hold any number of GDS Structures, however here there is only one. The GDS Library requires a name, as well as a user unit, 1 μm in this case, and a database unit, 1 nm. Finally, this GDS Library and all of its contents are written to a GDS file, 'basic.gds'. This file will be generated in the current directory open in MATLAB. This GDS file can be opened using many different programs, but KLayout [40] is free, effective, and simple to use. The GDS file is shown visually using KLayout in Figure 4.1.

While this example technically contains all of the functionality required to generate any arbitrary layout, defining every polygon with manual points is tedious and will inevitably lead to bugs and mistakes. So, wrapper functions that use these low-level functions were written to simplify the layout process for users. This has the additional benefit of hiding

some of the complications that can arise when adding multiple GDS Structures together and referencing. This allows the user to focus on generating the layout and structures and keeps the actual creation of the GDS structure behind the scenes.

Microrobot Design Library Overview

The Microrobot Design Library (MDL) can be found at the following location: `https://github.com/pinxisimitu/MEMS_Microrobot_Library`. In that directory, there are instructions for downloading and installing the software as well as tutorials to get started. MDL can be split into two main types of .m files, main scripts that generate GDS files and functions that define GDS Structures. Generally a main script has as least three separate blocks. To describe these blocks, the example that Ulf uses, shown in Listing 4.1, will be recreated using the functions and style of MDL.

```
% This script creates an example layout file (Basic.gds) that contains two
% identical rectangles. Shifted from each other by 1000 um.

% 03/22/2018
% Joey Greenspun

clc
close all
clear

tic                             % Timing Command, starts timer
gdsii_units(1e-6,1e-9)          % Sets units (user, database) for GDS file

default_layer_properties;       % Define the GDS layer names

% Do not run VREP simulation
global VREP_ignore;
VREP_ignore = 1;

%------------------------------------------------------------------------------
%% Create the GDS structures here

% Create first rectangle
h_rect.x = 0;                   % X coordinate of bottom left point
h_rect.y = 0;                   % Y coordinate of bottom left point
h_rect.w = 20;                  % Width (dimension in x direction)
h_rect.l = 100;                 % Length (dimension in y direction)
h_rect.layer = SOI;             % Set layer of rectangle
first_rect = rect(h_rect);      % Create GDS structure

% Create second rectangle
h_rect.x = h_rect.x + 1000;     % Shift the x coordinate
h_rect.y = h_rect.y + 1000;     % Shift the y coordinate
second_rect = rect(h_rect);     % Create GDS structure
```

```
35
36 ---------------------------------------------------------------------------
37 %% Collect all GDS structures and write to a GDS file
38
39 root = 'Basic';
40
41 collect_and_write;
42
43 toc %Timing Command, stops timer
```

Listing 4.2: An example MATLAB script that creates two rectangles and writes them to a file called 'basic.gds' using the Microrobot Design Library.

The example shown in Listing 4.2 generates the same two rectangles that Listing 4.1 generates. Each block of code, which would be naturally highlighted in a native MATLAB environment, is separated here by a dashed line. The first block, lines 1-20, clears the workspace and sets up some defaults that are typically not changed. The user and database units are set by line 12. Line 14 defines layer names, specific to the two-mask SOI process, as layer numbers. While a user is free to specify any number as the 'Layer' input for a GDS Structure or GDS Element, using names is helpful in remembering what each layer is used for. This library was built for a two-mask SOI process, so there will eventually be two masks generated. The default layer file, however, contains at least 7 layers. These layers are: SOI, SOIHOLE, RESOI, DUMMY, NOTDUMMY, TRENCH, and RETRENCH. Boolean operations are performed on these layers to ease the layout process, and generate the two mask files after all the layout is complete. On the device side of the wafer, most features are drawn with the SOI layer. Any shape drawn in the SOI layer corresponds to an area of device layer silicon that will remain on the wafer after the front side DRIE. The SOIHOLE layer is generally used to add etch holes to features in the SOI layer, however it can be used in any place to remove silicon from an SOI structure. The Boolean operation performed here is a simple subtraction (SOI - SOIHOLE). RESOI is used in areas to add device layer silicon back to an area where it has been removed with the SOIHOLE layer. This, while perhaps seeming obscure and useless, is an important layer that makes creating structures such as annuli and pins trivially easy. This layer gets added back in after the previous Boolean operation is performed (SOI - SOIHOLE + RESOI). The DUMMY and NOTDUMMY layers are used to reduce the exposed area on the device side mask. To have a uniform and relatively fast DRIE process, the exposed area on a wafer should be reduced as much as possible. These layers help achieve this goal. The NOTDUMMY layer is used to cover all SOI features. This allows the area between these features to be filled with DUMMY silicon, reducing the exposed area. These two layers are subtracted from each other and added to the Boolean equation to produce the final front side layer that will be used to generate the device side mask (SOI - SOIHOLE + RESOI + [DUMMY - NOTDUMMY]). The backside mask is much simpler and has only two layer that require subtracting. The TRENCH layer is used to etch away back side silicon, and the RETRENCH layer is used to add back side silicon into areas within a TRENCH section. The Boolean operation that generates this mask is

simply that subtraction, (TRENCH - RETRENCH). The last lines in this first block of code are used to ignore any VREP commands. If a simulation file is not desired lines 17 and 18 are included to ignore all VREP commands. This layer math happens after the generation of the GDS file using the KLayout software. The KLayout script that performs this math is in the MDL directory, but could be easily reproduced using any layout editor. The design rule checking (DRC) is also done by the KLayout software. This file is also in the MDL directory.

The next block of code, lines 21-36, defines the GDS Structures containing the two rectangles. The function capable of generating a rectangle is called 'rect'. All of the functions will be explained in the following section, but there are some important commonalities among them. Instead of passing individual arguments to these function, a handle variable is passed. This enhances the readability of the code, and allows easy editing and updating of the function. If a new parameter is defined and passed in, instead of editing the number of inputs to the function, this parameter can simply be added to the handle, 'h_rect'. This also enables that handle to be reused and updated as shown in lines 32-34.

In the final block of code, lines 37-43, the GDS structures are collected and written to a file. The 'root' variable defines the root of the file name. The 'collect_and_write' script gathers all of the GDS Structures produced by the second block of code and writes them to a file. Often many iterations of the same design are compiled and written to a file, so instead of overwriting this file every time, the script increments a counter on the file name. The first time this script runs it will generate a file named 'Basic.gds', followed by a file named 'Basic1.gds', then 'Basic2.gds' and so on. The tic and toc commands measure the amount of time it takes to generate the GDS Structures and write them to a file. If a mask is being generated with many features, somewhere in the 10's of thousands range, this can take up to 30 seconds or more.

Function Overview

Now, a brief overview of each function in the library will be given. To begin, a simple function will be dissected and explained. This function generates an alignment mark, which is composed of two rectangles. The code for this function is shown in Listing 4.3 and the main script that calls this function and generates the GDS containing the alignment mark is shown in Listing 4.4.

```
1  function out = align_mark(h_align)
2  % Function to create an alignment mark
3  % h_align.p0            = central point of the cross
4  % h_align.layer         = GDS layer of the alignment mark
5  % h_align.theta         = angle of rotation of cross
6  % h_align.w             = width of beams
7  % h_align.l             = length of beams
8
9
10 if ~isfield(h_align,'theta')
```

```
    h_align.theta = 0;
end

if ~isfield(h_align,'w')
    h_align.w = 5;
end

if ~isfield(h_align,'l')
    h_align.l = 100;
end

h_rect.x = h_align.p0(1) - h_align.l/2;
h_rect.y = h_align.p0(2) - h_align.w/2;
h_rect.w = h_align.l;
h_rect.l = h_align.w;
h_rect.layer = h_align.layer;
h_rect.theta = h_align.theta;
h_rect.p0 = h_align.p0;
vertical_bar = rect(h_rect);

h_rect.x = h_align.p0(1) - h_align.w/2;
h_rect.y = h_align.p0(2) - h_align.l/2;
h_rect.w = h_align.w;
h_rect.l = h_align.l;
h_rect.layer = h_align.layer;
h_rect.theta = h_align.theta;
h_rect.p0 = h_align.p0;
horiz_bar = rect(h_rect);

%% Grab all the GDS structures and arrays of structures

%Find all gds structures
a=whos();
b={};
c = 0;
for i=1:length(a)
    if(strcmp(a(i).class,'gds_structure'))
        c = c+1;
        str = sprintf('b{c} = %s;',a(i).name);
        eval(str);
    elseif(strcmp(a(i).class,'cell'))
        str = sprintf('temp = %s;',a(i).name);
        eval(str);
        if(isempty(temp))
            fprintf('Empty Cell! Something went wrong with %s!!\n',a(i).name)
            break;
        end
        str = sprintf('strcmp(class(%s{1}),''gds_structure'');',a(i).name);
        if(eval(str))
            str = sprintf('temp = %s;',a(i).name);
```

```
61              eval(str)
62              for i=1:length(temp)
63                  c = c+1;
64                  b{c} = temp{i};
65              end
66          end
67      end
68  end
69
70  % Outputs a cell array of GDS structures
71  out = b;
```

Listing 4.3: An example MATLAB function that generates an alignment mark. This function outputs a cell array of GDS structures.

```
1  clc
2  close all
3  clear
4
5  tic                              % Timing Command
6  gdsii_units(1e-6,1e-9)           % Sets units for the GDS file
7
8  % Define the GDS layer names
9  default_layer_properties;
10
11 % Etching/Release parameters
12 default_etch_properties;
13
14 % Don't do any VREP simulation
15 global VREP_ignore;
16 VREP_ignore = 1;
17
18
19 %% Define an alignment mark
20
21 h_align.p0 = [0 0];
22 h_align.layer = SOIHOLE;
23 am1 = align_mark(h_align);
24
25 %% Collect all structures and save them to a GDS file
26
27 root = 'Align';
28 collect_and_write;
29 toc %Timing Command, stops the timer and prints how long the sript ran for
```

Listing 4.4: An example MATLAB script that calls a function to create an alignment mark and creates a GDS file.

The main script, Listing 4.4, shows that there are only two inputs to the the align_mark function: an initial coordinate, [0,0], and a layer, SOIHOLE. The function itself, Listing 4.3, shows the rest of the information required to create the alignment mark. The organization

of a GDS function generally follows this pattern. At the top of the function, there is a block of comments that describes the purpose of each field in the function handle. Following this, the default values for some or all of those fields are given. This is the key that allows new functionality to be added to a GDS function without needing to edit all instances where that function is used. If a new field needs to be added, a default value can be defined such that the function does not break in places where that field is not used or needed. For example, creating rotated alignment marks is not necessary for many users, so the default value for 'h_align.theta' is set to 0. If a user wanted to create a rotated alignment mark, they could add a line such as '$h\_align.theta = \pi/2$;' between lines 22 and 23 in the main script. The existence of this field would cause the program not to execute line 11 in the GDS Function, and a rotated alignment mark would be created. After the default values comes the main body of the GDS function. Here, any number of functions can be executed: other GDS functions, mathematical analyses, etc. In this simple example, two instances of the 'rect' function are called to generate the alignment mark. In more complicated GDS functions, there are many calls to different functions inside this block. As long as these functions output a GDS Structure or an array of GDS Structures, the last block of this function, lines 42-71, will collect them and create the output array of GDS structures. This output will be passed from the GDS function to the main script where it will be written to the GDS file. This schema can be used to create new GDS Functions by users of this library.

Now, a brief overview of the basic GDS Functions within the library will be given. For a full description, look at descriptions in the source code of the functions in the MDL directory. Examples using these basic functions can be found in the 'Tutorial' folder in the library.

add_label: Creates text in layout for easy labeling of device parameters.

align_mark: Creates cross marks used for aligning different masks to each other.

annulus: Creates an annulus often used in the fabrication of membranes.

cavity: Creates a cavity in the backside of the wafer below a suspended membrane.

chevron: Creates a thermal Chevron actuator.

circle: Creates a circle or arrays of circles. Used in the etch hole script.

coil_spring: Creates a coil spring.

etch_hole: Creates etch holes in a variety of shapes. Can be used to automatically generate etch hole arrays.

fillet: Creates filleted corners to ease stress concentrations.

gcaws: Creates the layout for the mask template for the GCAWS6 tool in the UC Berkeley Nanolab.

gear_arm: Creates a pinion.

joint: Creates a pin joint.

latch: Creates the lever arm used in the electrostatic latches.

m_path: Creates an arbitrary path from an array of coordinates. Used in routing.

m_shape: Creates an arbitrary polygon from an array of coordinates.

make_jumper: Creates a fully synthesized jumping microrobot.

make_rot_spring: Creates rotary serpentine springs.

midpt: Function to find the midpoint between two points.

motor: Creates an electrostatic inchworm motor.
rect: Creates a rectangle.
rotate_pts: Function to rotate coordinates around an arbitrary point.
s_spring: Creates a serpentine spring.

4.2 MATLAB to VREP

The second half of the Microrobot library adds code to the GDS functions described above that outputs a VREP simulation file along with the GDS file. VREP is a powerful robotics simulation tool that can be scripted using the LUA programming language [41]. While MATLAB can also directly plug into the VREP simulation environment for control, it was more straightforward to generate a LUA file that is used by VREP to initialize and perform the simulation. At a high level, the GDS functions generate both the polygon in the GDS environment as well as the 3D shape in the VREP environment at the same time. While the shapes in the GDS domain require no additional information to be turned into a viable mask, the 3D shapes in the VREP environment do require additional information to produce a useful simulation. This is where this software differs from what has been done in the past. Many software packages have the ability to turn layout into a 3D model. SoftMEMS [36] has a package that can do this. Coventor has a product called SEMulator3D [34] that can simulate full processes to help inform device design. These types of products are helpful when the fabrication process is complicated and contains many etch and deposition steps. This software shows what the final product will look like given the GDS inputs and the process parameters. In the simple two-mask SOI process used in this work, it is intuitive what the final device looks like, but uncertain how the device will move and interact with its surroundings. The MDL simulation is different because it allows the user to simulate the movement of these 3D structures to verify how they interact with each other. These 3D objects are grouped together and constrained by the software. For example a pin joint can only rotate a certain amount. This constraint is defined by the layout geometries and can be programmed into the LUA script without needing any manual input from the user.

To generate these 3D objects from MATLAB, VREP Object functions are defined that can generate a primitive VREP object in a simulation. These VREP primitives have different simulation parameters that cause different behavior in the simulation. Additionally these objects can be grouped together either rigidly or through joints. An example main script that creates an inchworm motor both in layout and for VREP is given in Listing 4.5.

```
1 % This script creates a GDS as well as a VREP file for simulation
2
3 % 2/14/2017
4 % Joey Greenspun
5
6 clc
7 close all
8 clear all
```

```
9
10 tic                              % Timing Command, starts timer to see how long
       the sript runs for
11 gdsii_units(1e-6,1e-9)           % Sets units for (user, database) of the GDS
       file
12
13 % Define the GDS layer names
14 default_layer_properties;
15
16 % Define text label properties
17 h_label.layer = SOI;
18 h_label.height = 10;
19
20 % Etching/Release parameters
21 default_etch_properties;
22
23 %% Initialize VREP file
24 VREP_Properties;
25
26 global fid groups joints s_motor VREP_ignore
27
28 VREP_ignore = 0;
29
30 s_motor.motor_count = 1;
31
32
33 % Open LUA file
34 fid = fopen([VREP_path '\' VREP_fn '.lua'],'w');
35 fprintf(fid,'function inititalize_ojects()\n');
36 fprintf(fid,'pi = 3.14159265;\n');
37
38 %% Create a motor
39
40 % Generate an Inchworm motor
41 load Motor_TT
42
43
44 h_motor.pos = [700 700];                 % Shuttle position
45 h_motor.layer = [30, 31];                % Layers for [SOI and SOIHOLE]
46 h_motor.N = 30;                          % Number of comb finger gaps per
       motor
47 h_motor.num_inch_sets = 1;               % Number of replicate GCAs
48 h_motor.travel = 200;                    % Total travel of the shuttle
49 h_motor.angle = 45;                      % Angle of shuttle in degrees
50 [m2 moto_pts]= motor(h_motor);
51
52
53 %% Finish and close VREP file
54
55 % Process the groups
```

```
56 VREP_process_groups;
57
58 % Process the joints
59 VREP_process_joints;
60
61 % Process all parameter modifications
62 VREP_SetParam(0,0,0,1);
63
64
65 fprintf(fid,'end');
66 fclose(fid);
67
68 %% Collect all structures and save them to a GDS file
69
70 root = 'VREP';
71
72 collect_and_write;
73
74 toc %Timing Command, stops the timer and prints how long the sript ran for
```

Listing 4.5: An example MATLAB script that generates an inchworm motor GDS file as well as VREP simulation.

This code is very similar to previously mentioned main scripts used for this library that only generate a GDS file. The additional code before and after the standard GDS functions sets up and edits the LUA file that will eventually be used as input to the VREP simulation environment. The VREP initialization block, lines 24-36, will be unchanged and at the top of all main scripts that aim to generate a VREP simulation. This code sets up global variables, as well as initializes the LUA file. From here, the standard GDS function is called to create an inchworm motor. Lastly, after all of the GDS structures have been created, the groups and joints must be processed and written to the LUA file. This is done by lines 56-66. The LUA script generated has all of the information required to recreate the layout in the VREP environment. This LUA script can be called from within VREP, and the 3D rendering of the layout will appear in the VREP scene as shown in Figure 4.2. The example used to generate this can be found in the 'VREP_Example' folder in the library.

The LUA file generated by the MATLAB script is too long to show here, but it mostly consists of three commands: simCreatePureShape(), simSetObjectPosition(), and simSetObjectName(). These functions in the VREP API are capable of generating a 3D object in a VREP scene, placing the object, and naming it respectively. These objects can be cuboids, spheres, cylinders, or cones. Various flags on the objects are set to make them collidable and renderable. An example from this LUA script is given in Listing 4.6.

```
1 T684_70830 = simCreatePureShape(2,8,{0.025,0.025,0.5},0.000245437,nil);
2 simSetObjectPosition(T684_70830,-1,{6.8409,7.08485,0.27});
3 simSetObjectName(T684_70830,'T684_70830')
```

Listing 4.6: A snippet from the the LUA script that creates the 3D objects inside VREP. These lines of code generate a cylinder.

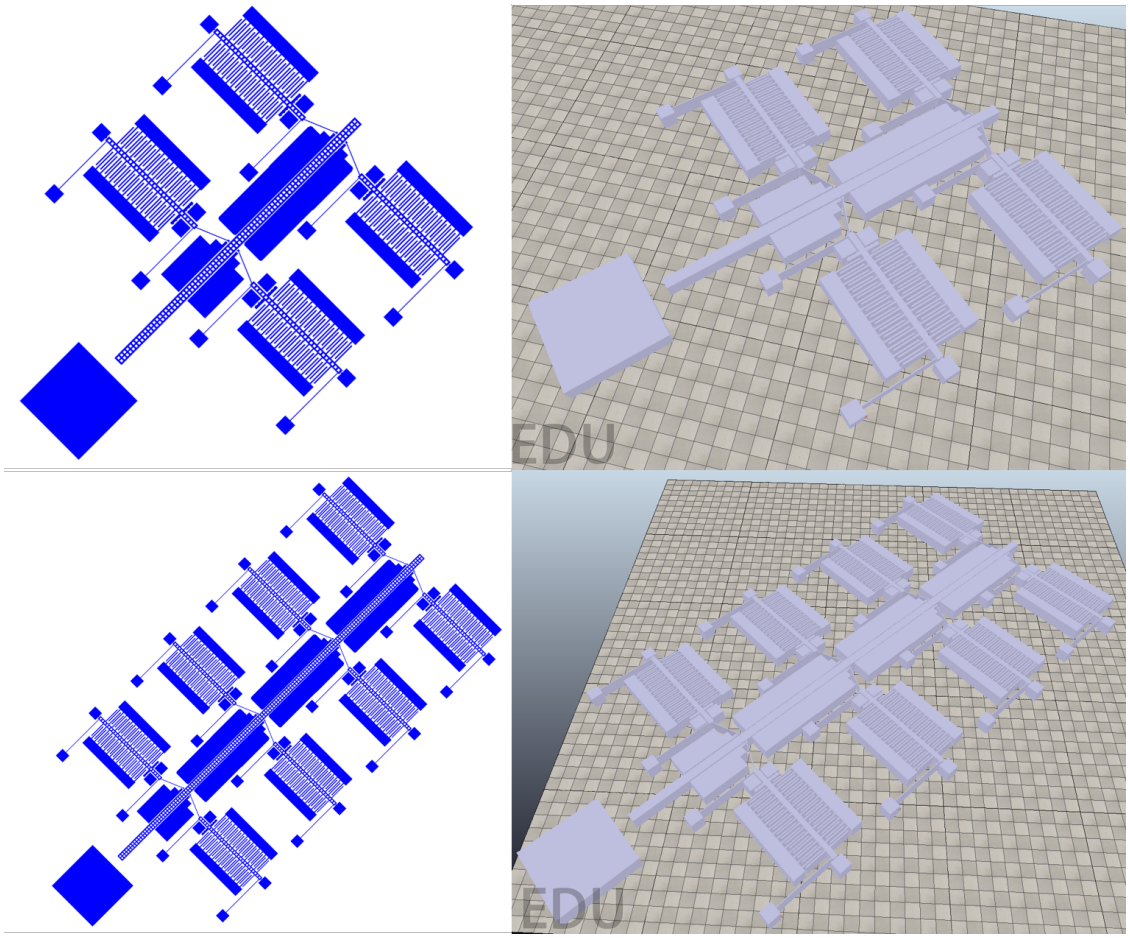

Figure 4.2: The layout files (left) and VREP simulations (right) for two different motor layouts. Everything shown here was generated programmatically from MATLAB and LUA scripts.

The inputs to the simCreatePureShape function define the shape of the object (2 corresponds to a cylinder), a simulation options flag (8 means the shape is respondable to collisions), the size of the object (radius,radius,length for a cylinder), and the mass of the object. Line 2 of this code moves the object to its layout position. Line 3 sets the name of the object so it can be referenced later in the LUA code if needed. These three lines are repeated and modified for every individual shape generated by the main MATLAB script.

One of the main goals in developing this VREP simulation pipeline was to find layout mistakes before the designs are fabricated. The inchworm motors are an excellent example of the utility of a simulation such as this. As stated previously, it is important that the front set of pawls and the rear set of pawls on an inchworm motor are offset by one half period of the shuttle teeth. If this offset is not correct the motors will be unable to push the shuttle forward. When a VREP simulation is run on the inchworm motors with properly offset teeth,

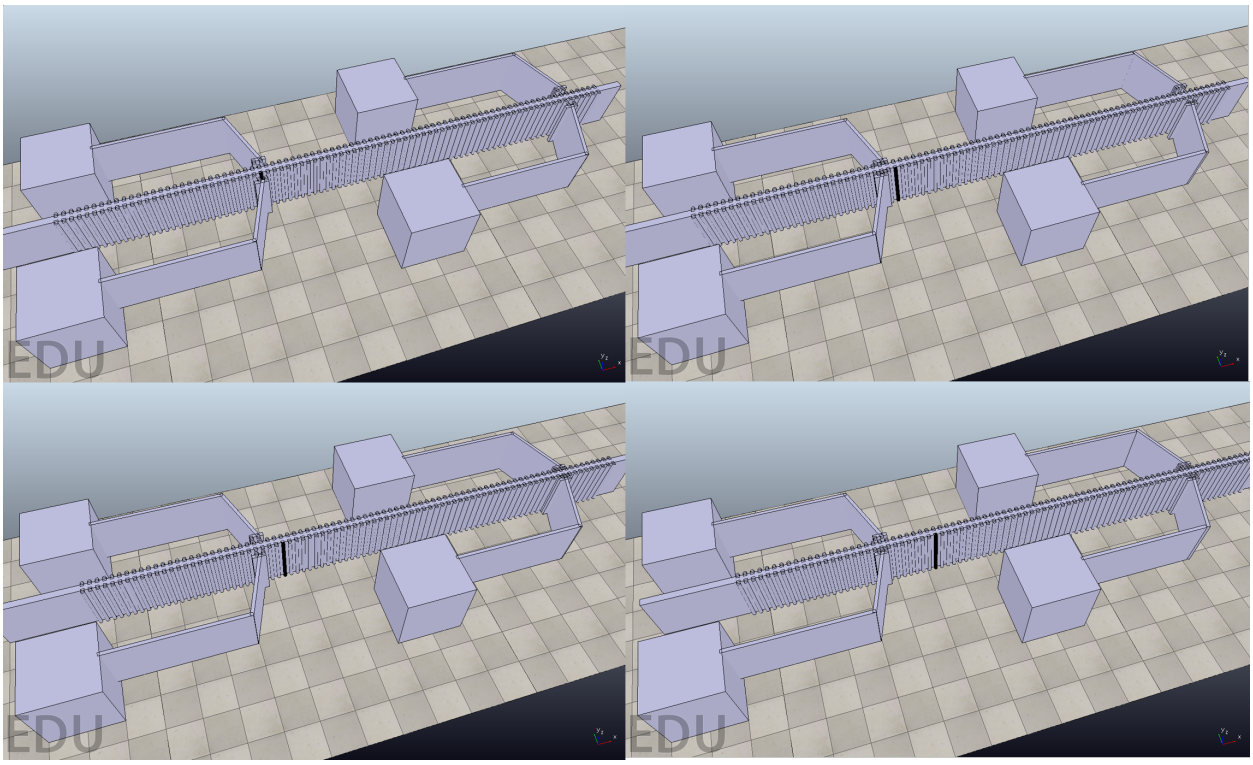

Figure 4.3: Successive screen captures of a simplified electrostatic inchworm motor pushing a central shuttle forwards. One tooth is colored black to more easily show the advancement of the shuttle.

the shuttle is continually advanced and the motors operate as expected. This is shown in the VREP simulation in Figure 4.3. The shuttle is pushed forward in each successive screen shot of the simulation. If one of the pawls is not properly aligned, the motors do not work in simulation. The pawls jam and the state machine driving the motors is unable to drive the shuttle forward. Had this been implemented earlier, it would have caught this exact mistake in one of the masks made during this work. The pawls were not offset from each other, the motors could not run, and it was only discovered once the devices were fabricated.

One additional area that this simulation environment helps find mistakes is in determining if features are anchored to the substrate or movable. VREP allows for collisions between objects to occur and transfers forces and energy appropriately. This allows for system level simulations to be performed on relatively complicated layout designs. If other movable objects are placed in front of the inchworm motor shuttle, the shuttle can push into these objects and deflect them. While there is no bending or stretching simulated in VREP, determining where an object moves given the forces on it and the constraints is very useful information. Additionally, if there is an area in the layout where a movable component was accidentally laid out on top of an anchored component, both objects become immovable. When running a simulation this would become readily apparent as the motor would be

unable to push this now anchored object. Again, had this been implemented earlier it would have saved a costly layout mistake and many hours in the cleanroom.

The framework developed here sets the stage for VREP to play a central role in simulating MEMS microrobots at the layout level. The program could also be used once the layout is confirmed to simulate actual operation and control of a jumping microrobot. VREP allows for the integration of sensing and control into the simulation environment. Before there are sensors physically added to the jumping microrobot, they can be prototyped in a simulation environment to determine their utility.

Chapter 5

Conclusion and Future Work

This work has described various ways to store and rapidly release mechanical energy at the MEMS scale. Initially this energy storage was studied to create a self-destructing circuit. Energy was stored in the MEMS device using a manual input and retained in the device using an electrostatic latch. Upon release of the latch, the energy was used to fracture a membrane and begin a chemical etching process. An identical energy storage and release method was used for an initial prototype of a jumping microrobot. While this microrobot was ultimately not successful in jumping, it provided a useful starting point for designing a working device. This second generation jumping microrobot improved on the initial prototype in key areas. It contained on-board motors that were used to load the energy storage elements. Those storage elements were composed of linear springs which allowed the MEMS motors to load them more efficiently. Most importantly, this second iteration microrobot was capable of jumping one full body length in the air. Lastly, a MATLAB library was created to aid in the design and simulation of all the MEMS devices developed throughout the course of this work.

5.1 Autonomous Jumping Microrobots

The long-term goal of this work is to create a system capable of autonomously jumping 20 cm at a rate of multiple times per minute. While this goal was not achieved during the course of this work, a foundation from which to build upon has been demonstrated. The research presented here focuses on mechanical energy storage as well as high force motors, however there are other key areas that will require further work to realize this goal.

Energy Storage and Mass Reduction

The energy storage element used in this work is the serpentine spring. While this spring is convenient and has some desirable properties, it is far from optimal. In the beginning of Chapter 2, the energy density of a bending cantilever was compared with that of a stretching

cantilever. It was determined that 9 times the energy can be stored in a stretching cantilever versus a bending cantilever. Ideally in a future iteration of a jumping microrobot, some or all the mechanical energy could be stored in stretching beams. The difficulty here is that achieving large displacements with these types of beams is inherently limited to the strain limit of the material. To create a beam that could stretch 1 mm, it would require a beam length of 125 mm. This is one of the reasons that Churaman [7] decided to integrate an elastomer into their jumping microrobot design. Elastomers can have strain limits of well over one hundred percent and could be used to create a more compact microrobot that uses stretching energy to jump. However, adding a new material into a process can be a time consuming and troubling ordeal.

An additional way to improve on the energy store in these beams is to fabricate them in the handle wafer as opposed to the device layer. Preliminary work has been done to show that beams can be successfully fabricated and used to store and release energy in the handle wafer. However, with the current state of the back side DRIE process, these beams must be 200 μm wide with a spacing of 400 μm. This significantly limits the design space. However, through tuning the DRIE recipe and changing the thickness of the handle wafer, springs could be fabricated in the backside that would free up space in the device layer for more motor area. This would have the added benefit of reducing the overall mass of the system, especially if a thinner handle thickness is used, which helps the microrobot jump higher.

One interesting feature of this energy storage method is that it is lossless. Single crystal silicon does not deform plastically at room temperature, so any energy stored in the beams will remain there indefinitely. Most battery chemistries have a self discharge rate that causes the energy stored in the battery to be lost over time. It is conceivable to use the mechanical energy stored in these silicon beams to do work on charged plates and generate electrical power!

High Force Motors

One of the main contributions of this work is the Inchworm of Inchworms motor topology. This motor allows the force from a standard inchworm motor to be amplified. The force output from this motor is currently the limiting factor for the height of these jumping microrobots. More energy dense springs could be used, however the IoI has only been shown to output on the order of 10 mN of force. To achieve the desired jump height of 20 cm, this force output must be increased to 100 mN or more. The IoI structure has been optimized to give a mechanical advantage of 10, which means the output at the inchworm motor itself must be increased to about 10 mN. Currently in the group, there are inchworm motors in development that have output 5 mN of force at 100 V with the ability to run at 0.4 m/s [42]! This motor was developed using the standard two-mask SOI process. To ensure that the comb fingers do not touch each other, a 1 μm gap remains between the comb fingers at the end of their actuation. One idea to increase the force output of these motors is to reduce the initial gap by 1 μm and use an ALD alumina coating on the sidewalls as the gap stop. This alumina layer would only need to be 50 nm thick to prevent breakdown when

the two sets of fingers come into contact. This reduction in initial gap could lead to an increase in the electrostatic force by over a factor of 2, which should be sufficient in creating a 10 mN inchworm motor. Additionally, this process is identical to the alumina coating of the low-voltage electrostatic latches shown in Chapter 2, so there is no process development required!

Control, Communications and Sensors

For a microrobot to be autonomous, it must be able to send, receive, and process signals. The Single Chip Micro Mote [43] has been in development in the group for over 3 years and is fully capable of being the brain for all the microrobots in development in the group. The chip has a wireless transceiver that will allow the microrobots to talk to each other. The embedded microprocessor can be programmed to send control signals to the motors through its general-purpose outputs. It can also receive signals with its general-purpose inputs from any sensors that might be incorporated into the microrobot in the future. One sensor already in use in some microrobot designs is a simple resistive end-stop detection. This allows one to know when the inchworm motor has reached the end of its travel. SCμM is a powerful platform that can be used and reconfigured to meet the needs of various microrobot designs now and in the future.

Power

The biggest research effort required to make this microrobot autonomous is in the area of power. No work was done here to address this issue, however we can again look to the Hollar microrobot for inspiration. Hollar ran a process, initially developed by Bellew [44], in the UC Berkeley Nanolab that could create high voltage transistors as well as solar cells. Throughout this work there was talk of recreating this process, until it was discovered that there was a company that ran a process nearly identical to Bellew's. X-FAB offers a high voltage SOI process that has been used to create solar cells and high voltage buffers by a collaborator. Their X-FAB chip was tested with the MEMS devices presented here, and it is fully capable of driving the inchworm motors. Furthermore, it has been shown that SCμM can drive the inputs of the high voltage buffers, and therefore SCμM can drive the inchworm motors!

While solar cells are a great means of producing energy in the sun, these microrobots will likely travel in shaded or dark areas as well. Energy storage will be required in addition to these energy-producing cells. There is no lack of thin film battery technology out there today, however there is a promising technology that is being developed in the same lab space as these microrobots. Ostfeld et al. have developed printed organic batteries [45] that have an energy density of 360 mJ/mg. These batteries in combination with the X-FAB solar cells could enable the microrobot to move indefinitely on scavenged and stored energy.

Integration and Assembly

Finally, all these components must be assembled and integrated, so they can work together. This is historically a difficult problem in microrobotics without a good solution. Gomez is developing a zero insertion force (ZIF) MEMS socket that should enable this integration effort [46]. The MEMS ZIF socket can mechanically and electrically connect these various chips that are created in wildly different processes. The characteristics, from the spacing to the stiffness, of the electrical probes can be modified and tailored to each specific chip allowing customization and optimization of these connections. This ZIF socket allows modularity in the integration of designs and makes it trivial to add a new chip from a different process.

While there is still some distance to go before an autonomous jumping microrobot will be hopping its way across the lab bench, the picture is starting to come into focus. All of the components required for autonomous operation are constantly being refined. It is this author's hope that in one more cycle of graduate students these components will be mature and ready to integrate into an autonomous jumping microrobot. And hopefully, the work presented has brought the dreams of those future graduate students one small hop closer to reality.

Bibliography

[1] Richard F Rizzolo, Thomas G Foote, James M Crafts, David A Grosch, Tak O Leung, David J Lund, Bryan L Mechtly, Bryan J Robbins, Timothy J Slegel, Michael J Tremblay, et al. "IBM System z9 eFUSE applications and methodology". In: *IBM Journal of Research and Development* 51.1.2 (2007), pp. 65–75.

[2] Chris Ziegler. *Motorola responds to Droid X bootloader controversy, says eFuse isn't there to break the phone*. 2010. URL: https://www.engadget.com/2010/07/16/motorola-responds-to-droid-x-bootloader-controversy-says-efuse/ (visited on 03/08/2018).

[3] Michael Hitt. *Strategic Management: Competitiveness and Globalization, Cases*. Cengage Learning, 2008.

[4] Richard P. Feynman. "There's plenty of room at the bottom [data storage]". In: *Journal of microelectromechanical systems* 1.1 (1992), pp. 60–66.

[5] Richard P. Feynman. "Infinitesimal machinery". In: *Journal of microelectromechanical systems* 2.1 (1993), pp. 4–14.

[6] M. Noh, S. W. Kim, S. An, J. S. Koh, and K. J. Cho. "Flea-Inspired Catapult Mechanism for Miniature Jumping Robots". In: *IEEE Transactions on Robotics* 28.5 (Oct. 2012), pp. 1007–1018. ISSN: 1552-3098. DOI: 10.1109/TRO.2012.2198510.

[7] W. A. Churaman, A. P. Gerratt, and S. Bergbreiter. "First leaps toward jumping microrobots". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2011, pp. 1680–1686. DOI: 10.1109/IROS.2011.6095090.

[8] HC Bennet-Clark. "The energetics of the jump of the locust Schistocerca gregaria". In: *Journal of Experimental Biology* 63.1 (1975), pp. 53–83.

[9] Malcolm Burrows. "How fleas jump". In: *Journal of Experimental Biology* 212.18 (2009), pp. 2881–2883.

[10] Seth Hollar, Anita Flynn, Colby Bellew, and KSJ Pister. "Solar powered 10 mg silicon robot". In: *Micro Electro Mechanical Systems, 2003. MEMS-03 Kyoto. IEEE The Sixteenth Annual International Conference on*. IEEE. 2003, pp. 706–711.

[11] John Keller. *IBM and PARC to design sensitive electronics for military that shatter to dust on command*. 2014. URL: http://www.militaryaerospace.com/articles/2014/02/parc-ibm-vapr.html (visited on 03/08/2018).

[12] V. Gund, A. Ruyack, A. Leonardi, K. B. Vinayakumar, C. K. Ober, and A. Lal. "Individually detachable polymer-silicon micro-parts for vaporizable electronics". In: *2017 19th International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS)*. June 2017, pp. 686–689. DOI: 10.1109/TRANSDUCERS.2017.7994141.

[13] Raymond J Roark, Warren C. Young, and Richard G. Budynas. *Roark's formulas for stress and strain; 7th ed.* New York, NY: McGraw Hill, 2002.

[14] P Temple-Boyer, C Rossi, E Saint-Etienne, and E Scheid. "Residual stress in low pressure chemical vapor deposition SiN x films deposited from silane and ammonia". In: *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films* 16.4 (1998), pp. 2003–2007.

[15] S. Bergbreiter and K. S. J. Pister. "Design of an Autonomous Jumping Microrobot". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. Apr. 2007, pp. 447–453. DOI: 10.1109/ROBOT.2007.363827.

[16] MJ Howard. "Elastomeric Materials (San Diego, CA: The International Plastics Selector)". In: (1977).

[17] Stefan Johansson, JanAAke Schweitz, Lars Tenerz, and Jonas Tiren. "Fracture testing of silicon microelements insitu in a scanning electron microscope". In: *Journal of applied physics* 63.10 (1988), pp. 4799–4803.

[18] V. Hatty, H. Kahn, and A. H. Heuer. "Fracture Toughness, Fracture Strength, and Stress Corrosion Cracking of Silicon Dioxide Thin Films". In: *Journal of Microelectromechanical Systems* 17.4 (Aug. 2008), pp. 943–947. ISSN: 1057-7157. DOI: 10.1109/JMEMS.2008.927069.

[19] WN Sharpe, J Pulskamp, DS Gianola, C Eberl, RG Polcawich, and RJ Thompson. "Strain measurements of silicon dioxide microspecimens by digital imaging processing". In: *Experimental Mechanics* 47.5 (2007), pp. 649–658.

[20] Ryan M Shih, Daniel S Contreras, Travis L Massey, Joseph T Greenspun, and Kristofer SJ Pister. "Characterization of electrostatic gap-closing actuator arrays in aqueous conditions". In: *Micro Electro Mechanical Systems (MEMS), 2018 IEEE*. IEEE. 2018, pp. 596–599.

[21] M. C. M. Lee and M. C. Wu. "Thermal annealing in hydrogen for 3-D profile transformation on silicon-on-insulator and sidewall roughness reduction". In: *Journal of Microelectromechanical Systems* 15.2 (Apr. 2006), pp. 338–343. ISSN: 1057-7157. DOI: 10.1109/JMEMS.2005.859092.

[22] Steven M George. "Atomic layer deposition: an overview". In: *Chemical reviews* 110.1 (2009), pp. 111–131.

[23] MD Groner, JW Elam, FH Fabreguette, and Sm M George. "Electrical characterization of thin Al2O3 films grown by atomic layer deposition on silicon and various metal substrates". In: *Thin Solid Films* 413.1-2 (2002), pp. 186–197.

[24] Ming-Chang M Lee and Ming C Wu. "3D silicon transformation using hydrogen annealing". In: *Proc Solid-State Sens., Actuator, Microsyst. Workshop*. 2004, pp. 19–22.

[25] Sarah Elizabeth Bergbreiter. *Autonomous jumping microrobots*. University of California, Berkeley, 2007.

[26] HC Bennet-Clark and GM Alder. "The effect of air resistance on the jumping performance of insects". In: *Journal of Experimental Biology* 82.1 (1979), pp. 105–121.

[27] Richard Yeh, Seth Hollar, and Kristofer SJ Pister. "Single mask, large force, and large displacement electrostatic linear inchworm motors". In: *Micro Electro Mechanical Systems, 2001. MEMS 2001. The 14th IEEE International Conference on*. IEEE. 2001, pp. 260–264.

[28] I Penskiy and S Bergbreiter. "Optimized electrostatic inchworm motors using a flexible driving arm". In: *Journal of Micromechanics and Microengineering* 23.1 (2012), p. 015018.

[29] Daniel S Contreras and Kristofer SJ Pister. "Durability of silicon pin-joints for microrobotics". In: *Manipulation, Automation and Robotics at Small Scales (MARSS), International Conference on*. IEEE. 2016, pp. 1–6.

[30] Juan Manuel Munoz-Guijosa, Daniel Fernandez Caballero, Vctor Rodrguez de la Cruz, Jose Luis Munoz Sanz, and Javier Echavarri. "Generalized spiral torsion spring model". In: *Mechanism and Machine Theory* 51 (2012), pp. 110–130.

[31] Daniel S Contreras and Kristofer SJ Pister. "Dynamics of electrostatic inchworm motors for silicon microrobots". In: *Manipulation, Automation and Robotics at Small Scales (MARSS), 2017 International Conference on*. IEEE. 2017, pp. 1–6.

[32] A Lam Research Company Coventor. *MEMS Accelerometer Design and Simulation*. 2018. URL: `https://www.coventor.com/mems-solutions/accelerometers/` (visited on 07/19/2018).

[33] A Lam Research Company Coventor. *MEMS Resonator Design and Simulation*. 2018. URL: `https://www.coventor.com/mems-solutions/resonators/` (visited on 07/19/2018).

[34] A Lam Research Company Coventor. *MEMS Gyroscope Design*. 2018. URL: `https://www.coventor.com/mems-solutions/gyros/` (visited on 07/19/2018).

[35] A Lam Research Company Coventor. *Microphone Design and Simulation*. 2018. URL: `https://www.coventor.com/mems-solutions/gyros/` (visited on 07/19/2018).

[36] SoftMEMS. *SoftMEMS Brining MEMS to the Mainstream*. 2018. URL: `http://www.softmems.com/index.html` (visited on 07/10/2018).

[37] Jason Clark. *Sugar: A Simulation Tool for MEMS Devices*. 2018. URL: `http://www-bsac.eecs.berkeley.edu/cadtools/sugar/` (visited on 07/10/2018).

[38] M. Freese E. Rohmer S. P. N. Singh. "V-REP: a Versatile and Scalable Robot Simulation Framework". In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. 2013.

[39] Ulf Griesmann. *Microphone Design and Simulation*. 2018. URL: `https://sites.google.com/site/ulfgri/numerical/gdsii-toolbox` (visited on 07/19/2018).

[40] Matthias Kofferlein. *KLayout - High Performance Layout Viewer And Editor*. 2018. URL: `https://www.klayout.de/` (visited on 07/19/2018).

[41] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. "The evolution of Lua". In: *Proceedings of the third ACM SIGPLAN conference on History of programming languages*. ACM. 2007, pp. 2–1.

[42] Daniel S. Contreras and Kristofer S. J. Pister. "A SIX-LEGGED MEMS SILICON ROBOT USING MULTICHIP ASSEMBLY". In: *Hilton Head Workshop 2018: A Solid-State Sensors, Actuators and Microsystems Workshop*. 2018.

[43] Brad Wheeler, Filip Maksimovic, Nima Baniasadi, Sahar Mesri, Osama Khan, David Burnett, Ali Niknejad, and Kris Pister. "Crystal-free narrow-band radios for low-cost IoT". In: *Radio Frequency Integrated Circuits Symposium (RFIC), 2017 IEEE*. IEEE. 2017, pp. 228–231.

[44] Colby L Bellew, Seth Hollar, and KSJ Pister. "An SOI process for fabrication of solar cells, transistors and electrostatic actuators". In: *TRANSDUCERS, Solid-State Sensors, Actuators and Microsystems, 12th International Conference on, 2003*. Vol. 2. IEEE. 2003, pp. 1075–1078.

[45] Aminy E Ostfeld, Abhinav M Gaikwad, Yasser Khan, and Ana C Arias. "High-performance flexible energy storage and harvesting system for wearable electronics". In: *Scientific reports* 6 (2016), p. 26122.

[46] Hani Gomez, Daniel Contreras, Joseph Grenspun, and Kristofer Pister. "Zero Insertion Force MEMS Socket for Microrobotics Assembly". In: *Manipulation, Automation and Robotics at Small Scales (MARSS), 2017 International Conference on* (2017), p. 26122.