CHARACTERIZATION OF MASS-LOADED SILICON NITRIDE ON-CHIP RESONATORS FOR TRACEABLE SENSING OF LOW AMPLITUDE ACCELERATION

TIMOTHY HODGES

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Applied Science in Mechanical Engineering

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

Abstract

Low frequency (<100 Hz) acceleration sensing with low noise and traceability is critical in seismology, military surveillance, and emerging technologies. Typically, MEMS (Microelectromechanical systems) are not ideally suited for low frequency accelerometry since their fundamental thermomechanical fluctuation noise limit is higher than in macroscopic accelerometers of higher proof mass. However recent work on thin film MEMS resonators shows promising development in the reduction of damping which in turn reduces fundamental thermomechanical fluctuation noise limit. We aim to harness these low damping thin films in the context of accelerometry, by mass loading them to make them sensitive to acceleration. This work reports an experimental characterization of a mass-loaded silicon nitride membrane-based resonator, which is investigated towards the development of accelerometers for acceleration sensing at low frequencies. We experimentally demonstrate a 1.1×10^{-6} kg proof mass system achieving a 17,950 mechanical quality factor for a 526 Hz natural resonance frequency, which compares favorably to other optically interrogated on-chip accelerometers [1]–[3]. The inferred acceleration noise floor of the device is currently limited by the displacement noise of the optical fiber displacement readout, yielding a noise amplitude spectral density of $1\ \mu g/\sqrt{Hz}$ at 10 Hz. This thesis first details a literature review of various high-performance, mass-loaded MEMS accelerometers categorized by their transduction methods, followed by a comprehensive overview of our devices design and fabrication process. Followed by an overview of the performance of our devices under different mounting conditions. Finally, a custom finite difference simulation is presented to determine the limiting factor in our device's performance along with concluding remarks and potential future work.

Acknowledgements

This thesis was accomplished through the invaluable collaboration and guidance of numerous individuals.

First and foremost, I express my gratitude to all my colleagues at the UOttawa MEMS laboratory, who played a vital role in the success of this thesis. In particular, I would like to thank Nikaya Snell and Albert Mu for their insightful help throughout the fabrication process. Additionally, I am grateful to Alexandre Bouchard and Michel Stephan for their invaluable assistance during the characterization stage of this thesis.

Furthermore, I extend my sincere appreciation to our collaborators at the National Research Council of Canada Metrology Research Center, whose advice and expertise were instrumental in the successful completion of this thesis. Specifically, I am grateful to Dr. Richard Green, Dr. Lixue Wu, and Dr. Triantafillos Koukoulas for their valuable contributions to this research.

Finally, I would like to express my gratitude to my supervisor, Raphael St-Gelais, for his unwavering support, guidance, and kindness throughout the entire process of this thesis.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

Low noise and traceable acceleration sensing at low frequencies (< 100 Hz) is of fundamental importance in well-established fields, such as seismology, military surveillance, and for the International Monitoring System (IMS) by the Comprehensive Nuclear-Test-Ban Treaty Organization (CTBTO). Likewise, it is also an important building block in more emerging technologies such as cold atom interferometry [4] and inertial navigation [5]. The IMS is a global network composed of seismic, hydroacoustic, infrasound and radionuclide detectors with the aim of detecting nuclear weapons testing and deployment worldwide [6]. The goal of the CTBTO is to enforce the Comprehensive Nuclear-Test-Ban Treaty that bans all nuclear explosions, whether for military or peaceful purposes, since it was established in 1996. Figure 1 shows a global map of seismic stations deployed in the IMS. These stations monitor for underground nuclear explosions by measuring the shockwaves in the ground using several low noise and traceable accelerometers [6].

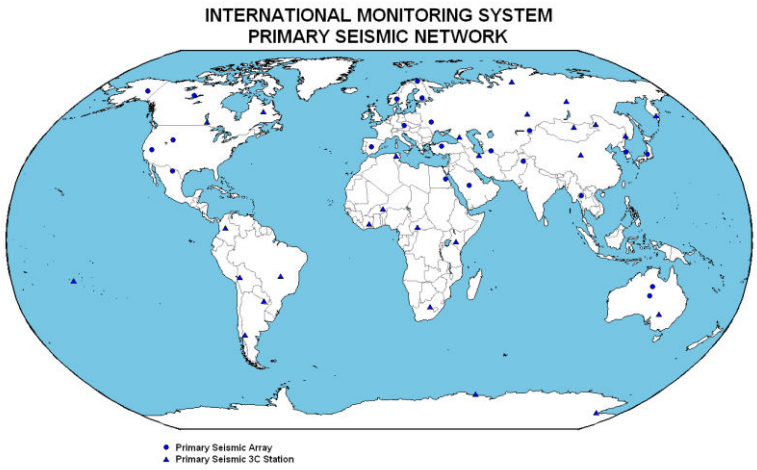

Figure 1: Map of the IMS Seismic Network [7]

Emerging technologies such as cold atom interferometers and inertial navigation also rely in part on high performance accelerometers. Cold atom interferometers promise to provide high precision absolute measurements of accelerations and fundamental quantities such as the gravitational constant. Figure 2 shows a cold atom interferometer deployed dynamically for gravimetry. This instrument measures the acceleration of a free-falling gas of ultracold (i.e., stable) atoms. This atomic measurement of acceleration must also be complemented by a conventional accelerometer which is used to give a first rough estimation of the acceleration and to fill in the dead times that come from the discrete number of atoms [4] involved in the measurement. To this end, conventional accelerometers with low noise floors, such as those developed in this work, are desirable to improve cold atom interferometry [8].

Another emerging application needing high performance accelerometers is inertial navigation, which aims at tracking position reliably with the use of inertial measurement units (IMUs) as opposed to the Global Navigation Satellite Systems (GNSS, or GPS in North America) that could be vulnerable to military and terrorist sabotage. Inertial navigation would also be beneficial for firefighters and first responders that operate in environments where the GNSS have degraded performance or are unavailable [5]. In principle, constantly measuring the acceleration of an object allows tracking of its exact position in real time via double integration. However this is extremely sensitive to noise and will grow quadratically with time, thus requiring accelerometers with an extremely low noise floor [9].

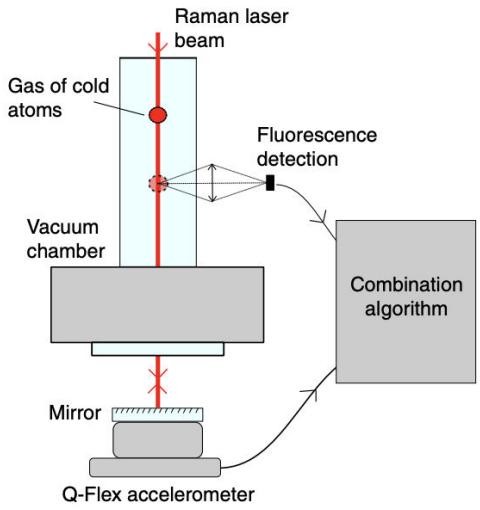

Figure 2: Gravimetry with an Absolute Cold Atom Sensor, where deadtime between atom detection is filled in by a classical high performance "Q-Flex" accelerometer. Taken from [4]

1.2 Working Principle and Objective

We aim to create accelerometers for these low frequency, high precision, applications using MEMS (Microelectromechanical systems) material and platforms. MEMS constitute the technology of microscopic devices, particularly those with moving parts. The advent of MEMS technology was propelled by the materials, lithography and etching techniques used in the microchip industry to mass manufacture transistors at low cost. These same fabrication techniques and materials are used today to manufacture all types of MEMS devices from pressures gauges to thermometers. MEMS accelerometers are already in widespread use today, they are found in all kinds of technology from smartphones to aircraft, however they have not yet reached the performance levels necessary for precise low frequency accelerometry.

Due to the nature of their fabrication process, MEMS accelerometers have many advantages over macroscopic accelerometers, such as portability, low cost, low power consumption and small footprint. Their portability is fundamentally much greater than macroscopic accelerometers. This is of notable importance to the IMS where seismic sensors need to be

installed in remote locations around the globe. The low-power consumption inherent in their small size is also a critical advantage since they can potentially be powered with renewable resources enabling off-grid installations, which is also beneficial to the IMS. In addition, the large batch fabrication techniques perfected in the microchip industry allow MEMS fabrication processes to benefit from economies of scale that drastically reduce the per unit cost as demand increases.

Unfortunately, MEMS also have fundamental drawbacks—because of their small size, they have low proof-mass (m) and high natural frequency (ω_0), which both lead to high noise compared to larger scale accelerometers. The basic building blocks of an accelerometer are presented in Figure 3, where a proof mass' (m) movement is measured upon application of an external acceleration a. The springs have a stiffness (k) and hold the mass in position. The quality factor (inverse of the damping ratio) (Q) is the ratio of the energy stored in the system to the energy dissipated per cycle of oscillation. The inertia forces move the mass, and this movement is measured to determine the acceleration of the entire system.

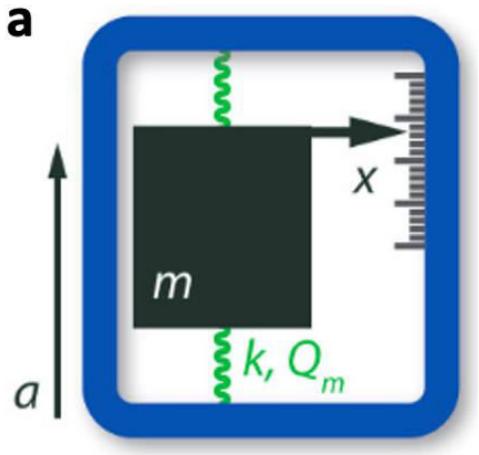

Figure 3: Building blocks of an accelerometer [5].

Assuming that all non-fundamental noise sources are minimized, the noise spectral density (S_a, in units of acceleration per root Hz, i.e., $\frac{\mathrm{m}}{\mathrm{s}^2\sqrt{\mathrm{Hz}}}$ or $\frac{\mathrm{g}}{\sqrt{\mathrm{Hz}}}$) of any accelerometer is ultimately dictated by the fundamental thermomechanical fluctuation noise limit given by:

$$S_{a,tmech} = \left(\frac{4k_B T\omega_0}{mQ}\right)^{\frac{1}{2}} \#\#$$

Equation 1

where k_B is the Boltzmann's constant, T is the temperature and Q is the mechanical quality-factor of the mass spring system pictured in Figure 3. Reaching low noise means using accelerometers with a high proof mass m, high Q, and low ω_0 to minimize $S_{a,tmech}$. High mass is naturally contrary to the on-chip nature of MEMS. Likewise, the natural frequency (ω_0) of mass spring systems is inversely proportional to dimensions, making ω_0 higher in MEMS compared to macroscopic systems. High resonance frequency (ω_0) is useful in some cases, as it increases sensing bandwidth (e.g., for machine vibration analysis [10]), but for high precision applications such as seismology and navigation, minimizing ω_0 for reducing noise is desirable. As a results of this relation, commercially available MEMS typically suffer from high noise (e.g., $\mathrm{S_a}$ = 20 $\frac{\mu\mathrm{g}}{\sqrt{\mathrm{Hz}}}$) for state-of-the-art ADXL354 by Analog devices [11]) that make them uncompetitive compared to larger scale accelerometers and seismometers (e.g. $\mathrm{S_a}$ = 3 $\frac{\mathrm{ng}}{\sqrt{\mathrm{Hz}}}$ for Nanometrics Titan [12]).

Thankfully, development in MEMS resonators over the last 20 years, more specifically of silicon nitride (SiN) resonators, provide a path to reducing noise by maximizing the Q-factor in MEMS resonators. Recent work has pushed quality factors in MEMS to extremely high values (>10^9 in [13]). Our goal is to translate such recent advances in high Q devices to accelerometers, such that we minimize their noise. While high Q was achieved in very low proof mass devices

not aimed at accelerometry [13] (Figure 4 b), our goal is to add an integrated proof mass to these high-Q devices and thus reach a high $m \times Q$ product suitable for low noise accelerometry (Figure 4 a). This mass will also decrease the natural frequency which will further decrease the fundamental noise limit and of low frequency sensing.

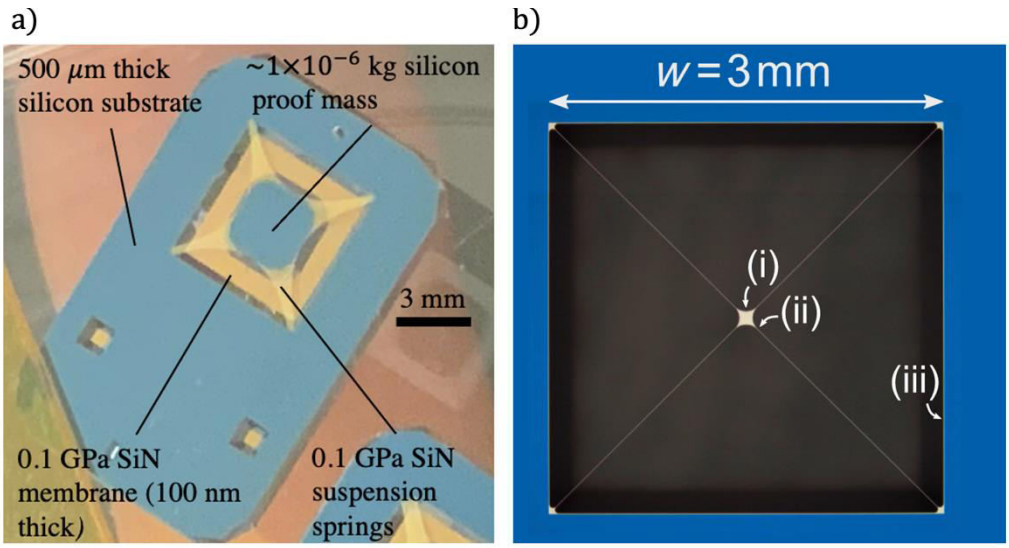

Figure 4: (a) 6 mm × 6 mm accelerometer fabricated in-house ($Q \cong$ 18000). (b) Trampoline Resonator from [13] ($Q \cong 4.5 \times 10^6$).

In summary, the objective of this research project is to harness recent developments in ultra-high Q-factor nanomechanical resonator devices to create resonators with a mass, Q-factor and natural frequencies combination that can compete with macroscopic accelerometers. To do so, we mass-load a typical (i.e., low mass) ultra-high-Q SiN resonator (e.g., Figure 4 b), and observe whether a high mechanical quality factor can be maintained after the mass is added. Devices are designed, fabricated, and characterized in-house to infer their acceleration noise S_a. These results are then benchmarked against a custom-built numerical simulations algorithm to validate their performance against theoretical models and identify performance limits.

2. Literature Review

2.1 Overview of MEMS accelerometers

As described above, there is a need for low frequency, high precision accelerometers in many current and emerging applications. MEMS would be ideal candidates for these applications due to their inherent portability, low power consumption and mass manufacturability. However, they suffer from high thermomechanical fluctuation noise relative to their macroscopic (larger) counterparts. These advantages have outweighed the noise limitation in the eyes of many researchers, such that there has been substantial effort in developing low frequency MEMS accelerometers in the past decade.

Within the general topic of MEMS accelerometers, there exists various transduction methods for measuring the mass displacement, and/or for active feedback mass stabilization. Two notable methods are capacitive and optical. Capacitive interrogation relies on electrodes on the proof-mass and frame (see comb-like structure in Figure 5). In many cases, the mass does not actually move, but is dynamically stabilized in place. In such "closed-loop" accelerometers, the magnitude of the stabilization force is recorded in real time and is used to infer acceleration. It is worth noting that closed-loop accelerometers remain bound by the performance limit given by Eq. (1). Their closed-loop nature is mostly beneficial to the sensing dynamic range (stabilized mass can tolerate greater acceleration) and to prevent ringing at the resonance frequency [14]. Optical interrogation is another technique used to measure the displacement of a proof mass in a MEMS accelerometer. In this method, a light source is used to illuminate the proof mass, and the displacement of the mass is determined by measuring the change in the reflected or transmitted

light. It is typically in an "open-loop" configuration [15], i.e., the mass is free to move and is not stabilized in place. This does not require any connections to the MEMS device itself.

Capacitive interrogation is simple and allows for high performances reaching the ng/√Hz level in state-of-the-art commercial products [11]. A lot of these devices are unpublished commercial secrets, with the notable exception of Laine and Mougenot's [14] published work on a capacitive MEMS accelerometer with improved detection at low frequencies shown in Figure 5. It can deliver broadband (0 to 800 Hz) and high-fidelity measurements of low-level ground motion.

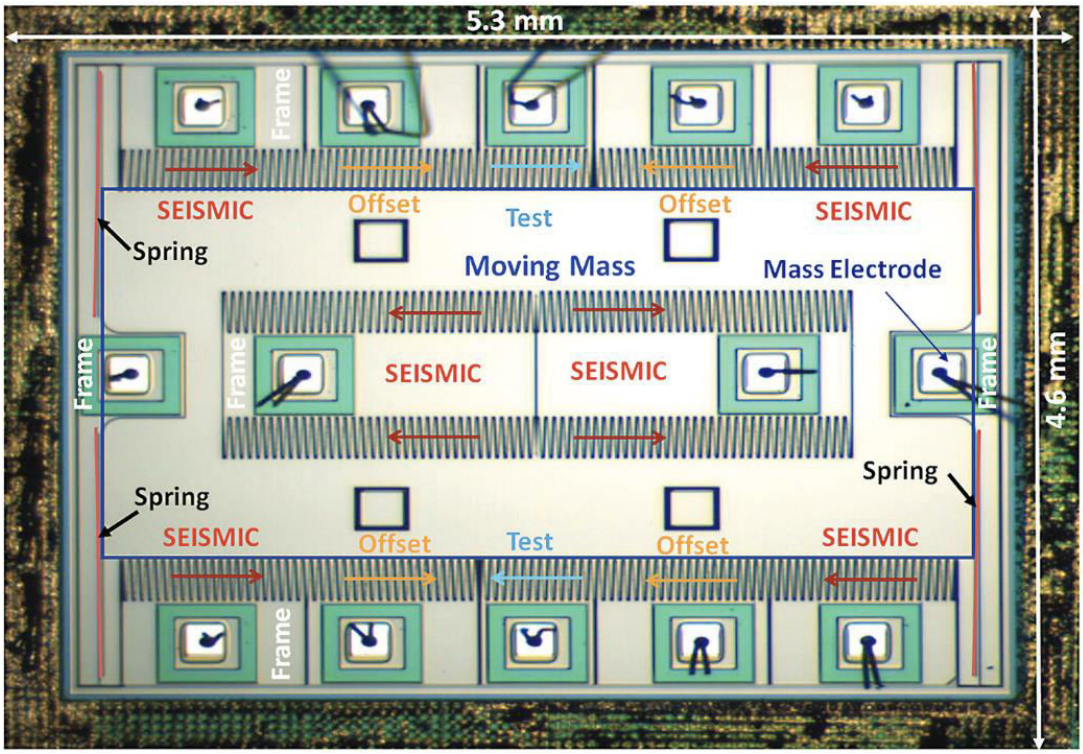

Figure 5: MEMS architecture of a single vector sensor from a digital unit (DSU) as seen by microscope [14]

This capacitive MEMS device has several combs of electrodes arranged in different groups. These groups serve different functions, such as measuring the acceleration signal, compensating for gravity, and performing active built-in tests. Each group consists of at least two combs, one for upward motion and one for downward motion. At the end of each axis of the MEMS device,

there are two springs that allow the mass to move along its sensitive axis (indicated by arrows in Figure 5). It features a noise floor of 30 ng/√Hz at 1 Hz, making it one of the most performant devices reviewed.

Another high-preforming low frequency MEMS capacitive accelerometer was presented by Zhang, *et al.* [16]. The device utilizes an asymmetrical anti-spring structure and is fabricated using silicon-on-insulator (SOI) technology (Figure 6). A simplified model of each anti-spring is a static curved beam with one end fixed and the other end simply supported, leading to lower stiffness compared to regular (straight-beam) springs (Figure 6 b). As a result, three anti-spring structures are more sensitive and stable than a symmetrical four-spring structure. The best resolution of the sensor was found to be 51.8 ng/√Hz at 1 Hz. The results demonstrate the high sensitivity and low noise performance of the proposed MEMS accelerometer, making it a potential candidate for use in seismic applications.

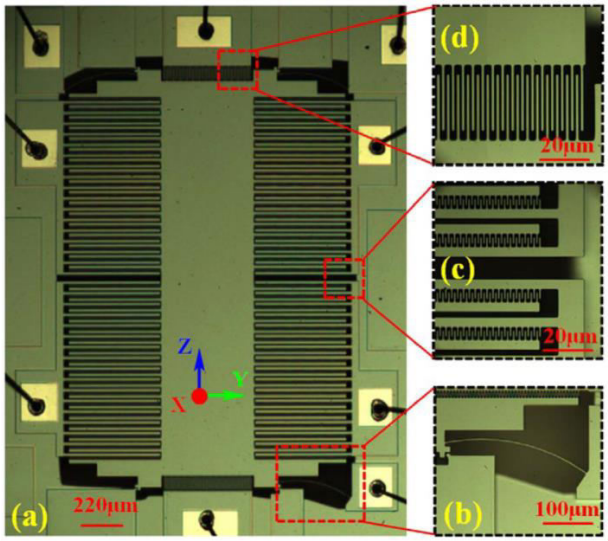

Figure 6: A low noise capacitive MEMS accelerometer with anti-spring structure[16]

While capacitive devices have good performances and wide dynamic range, optical devices have the advantage of being traceable through the calibration of the laser and time base, i.e., they can be calibrated by tracing them back to standard units, making them ideal for high

fidelity measurements over the long term. Several optically interrogated MEMS accelerometers have recently been developed, notably by metrology and standards institutes such as NIST (National Institute of Standards and Technology). Likewise, our partners at NRC (National Research Council) are interested in such devices, whose traceability would make them eligible for integration in the IMS (International Monitoring System). One such notable accelerometer was produced by F. Zhou, *et al.* [3]. Where a silicon mass is suspended by SiN tethers and its displacement is measured using a Fabry-Perot cavity as shown in Figure 7. The device is designed for out-of-plane vibration with a mechanical quality-factor of 565 and natural frequency of 9.6 kHz. The accelerometer has a resolution of 32 ng/√Hz over a frequency range of 6.8 kHz above 100 Hz. The combination of a single vibrational mode above 100 Hz and broadband thermally limited resolution allows for accurate conversion from sensor displacement to acceleration. This also enables the measurement of acceleration in terms of laser wavelength, allowing the sensor to calibrate internally and act as an intrinsic standard. The accelerometer achieves the thermodynamic limit of resolution over a frequency range greater than 13 kHz. However, given high noise under 5 kHz (above the seismic range) and lower Q-factor relative to other devices on the market, it is currently not ideally suited for low frequency high precision accelerometry.

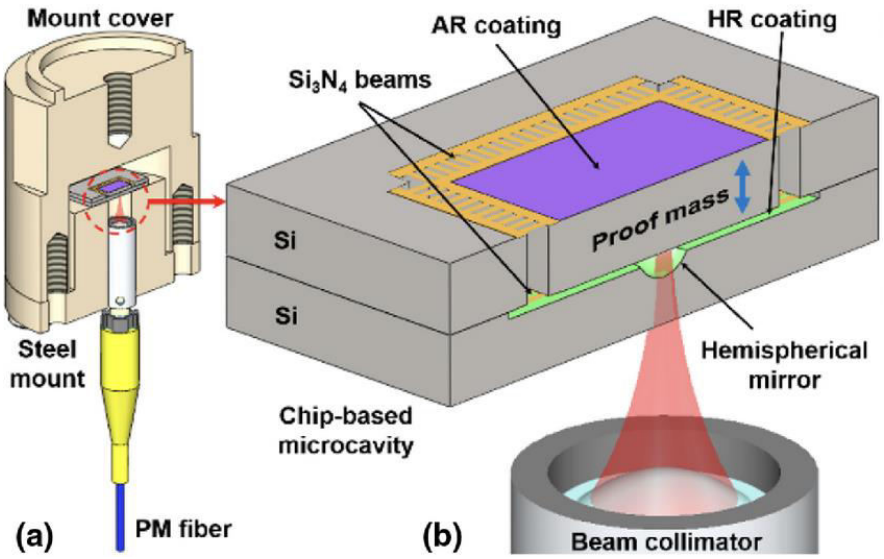

Figure 7:Out-of-plane mass loaded SiN accelerometer. [3]

A similar optically interrogated device was developed by G. Krause, *et al.* [1]. Using SiN tethers and an Si mass. However, this device is designed for in-plane motion (mass moves laterally) as shown in Figure 8 with high stress silicon nitride tethers (800 MPa). This device has on-chip integration and can achieve a broadband acceleration resolution of 10 μg/√Hz, at a bandwidth greater than 20 kHz using sub-milliwatt optical power. The resulting quality factor is 1.3 million which is promising to reduce the fundamental noise limit, although the natural frequency is 27.5 kHz resulting in degraded sensitivity at lower frequencies. Nanoscale optomechanical cavities, like the zipper cavity used in this work can provide strong radiation pressure back-action, limiting ringing at the resonance frequency.

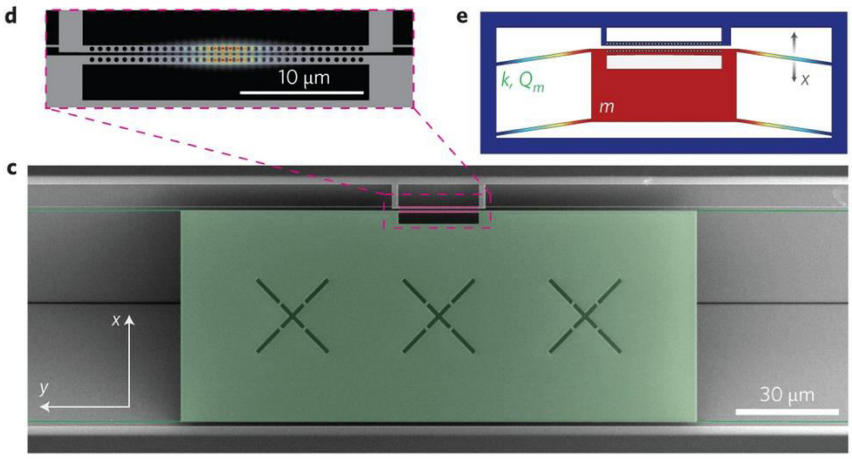

Figure 8: In-plane mass loaded SiN accelerometer [1]

Another mass-loaded, optically interrogated MEMS accelerometer is based on a graphene and was made by Fan, *et al.* (Figure 9). This device features a double-layered graphene membrane, supporting an Si proof mass from the top plane [2]. Graphene's strength, high surface-to-volume ratio, and electronic properties make it a promising material for use in MEMS. However, incorporating graphene into suspended structures such as proof masses on graphene membranes presents several technological challenges, including the collapse and rupture of the graphene. The device has a quality factor of 63 and a natural frequency of 78.8 kHz. Once again due to its high natural frequency and low Q-factor it is not best suited for low frequency accelerometry, the lack of support from the bottom plane could also potentially allow for the rocking motion of the mass during acceleration. This is not ideal as all the acceleration should be converted into linear motion in the direction of measurement.

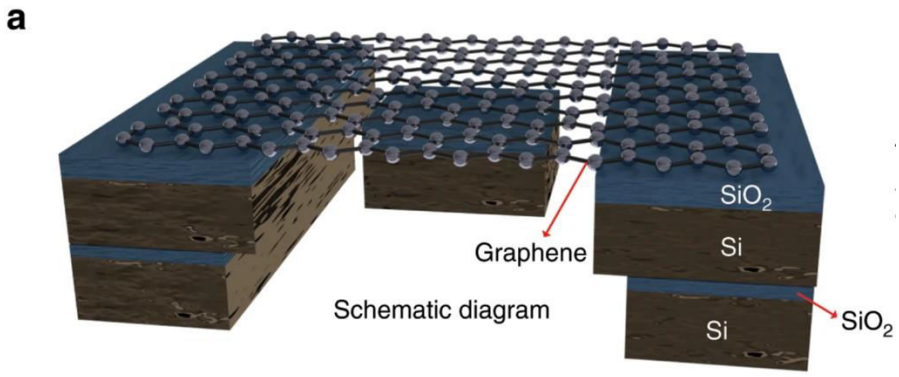

Figure 9: Graphene mass loaded accelerometer [2]

Another optically interrogated MEMS proof-mass accelerometer was developed by Z. Qu *et al.* [17]. The proposed sensor is a silicon flexure accelerometer with a spring-mass structure consisting of compliant spring beams, a proof mass and a sensor frame (Figure 10). When acceleration is applied, the in-plane motion of the proof mass can be measured by an optical interferometer with a Fabry-Perot (F-P) cavity formed between the fiber end-face and the proof mass sidewall. The sensitivity of the micro-machined optical accelerometer is measured to be 100 ng/√Hz at 1 Hz, with a natural frequency at 31 Hz.

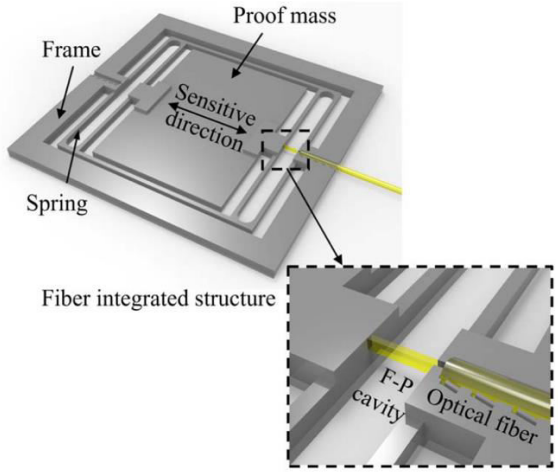

Figure 10: Si spring optically integrated accelerometer [17]

Another promising interrogation method is resonant accelerometers that use resonators coupled to a suspended proof mass to convert the input acceleration into an inertial force. The inertial force on a proof mass changes the natural frequency of the coupled resonators. Therefore acceleration is determined by measuring a change in frequency of the resonators as opposed to the movement of the proof mass. This leads to a frequency-modulated output response, which can provide increased immunity to noise as well as better resolution at lower frequency since it is not limited by the frequency of the fundamental mode of vibration [18]. One such accelerometer (Figure 11) uses strain gauges as the coupled resonator [19].

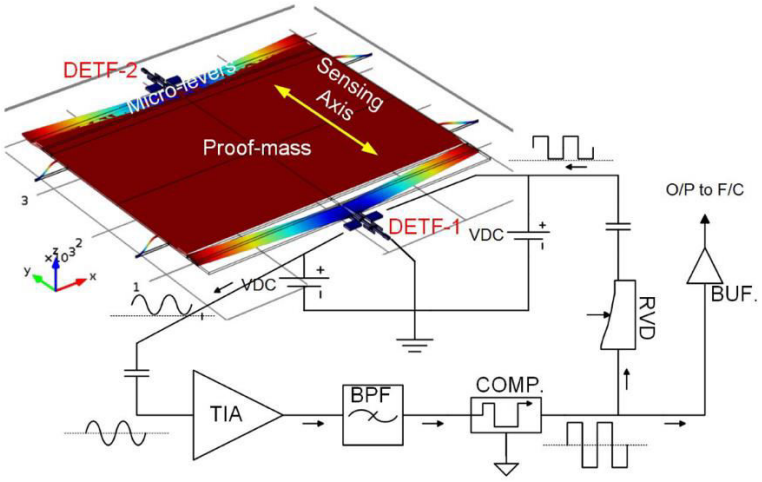

Figure 11: Strain-gauge resonant accelerometer [19]

This accelerometer meets the requirements of seismic-grade applications. Initial characterization shows a noise-limited resolution of approximately 3.2 μg/√Hz, at frequencies up to 5 Hz. It compares favorably to state-of-the-art capacitive and optical accelerometers in terms of device volume, which is reduced by a factor of 15 to 300 compared to previously reported capacitive designs of similar resolution.

Another such resonant accelerometer was developed in [18] (Figure 12). The accelerometer works by detecting the frequency shift of two vibrating beams. These beams, known as clamped-clamped (C-C) beams, are positioned on either side of a proof mass. The proof mass is connected to the beams through a lever that amplifies the force applied to them. When the device is subjected to an input gravitational acceleration, the proof mass experiences a differential axial force, causing a shift in the frequency of the beams. The quality factor of resonator is 60,000, and the natural frequency of the beams operating in the second transverse mode of vibration is approximately 420 kHz. This results in a device with a noise floor of 17.8 ng/√Hz in the seismic range.

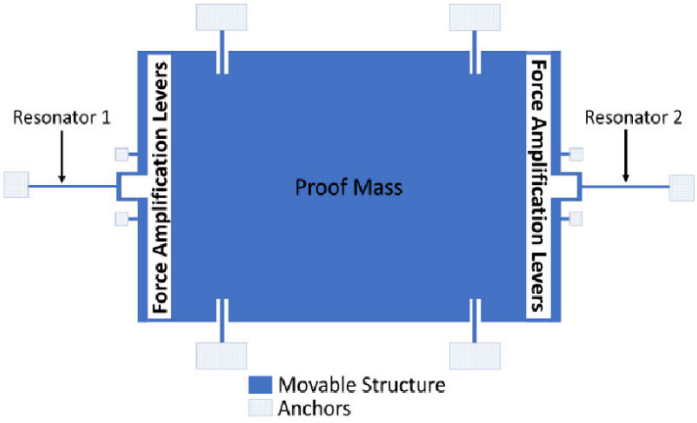

Figure 12: Schematic of (C-C) Resonant accelerometer [18]

2.2 Tensile silicon nitride as a material platform for accelerometry.

Research on SiN resonators has yielded promising results for applications in MEMS optical accelerometry. A study [20] analyzed commercial SiN membranes to evaluate their optical and mechanical properties. The results showed that membranes that were 50 nm thick and 1 mm^2 in size had a mechanical Q-factor of 10^6 at 293 K, which is significantly higher than other MEMS resonator devices. Additionally, the near-infrared optical loss at 293 K was less than 2×10^{-4}, and the membranes were able to accommodate large laser spot sizes, making them well-suited

for optical interrogation. More recent work on SiN resonators with integrated waveguides produced even higher Q-factors reaching 10^7 [21]. This bodes well for the fundamental thermomechanical fluctuation noise of mass-loaded SiN resonators that harness these high Q-factors and mass loading to reach lower noise limits (see Eq. 1). However, most SiN high mechanical-Q resonator also have low mass, limiting their performances for accelerometry.

Work in [22] studied the effect of transitioning from low mass SiN resonators to a mass loaded structure. Using an analytical model and finite element analysis, it was revealed that as the mass load increases, the resonator mode shape becomes independent of the specific mass. Additionally, it was discovered that for a large mass, the mode shape converges to a specific shape, implying that the quality factor reaches a saturation point in the case of a large mass load. The findings were validated through measurements of the quality factor of a SiN trampoline resonator for different load masses. This is promising in the case of mass-loaded SiN resonators since it demonstrated that mass can be increased without degradation of the Q-factor after a certain threshold.

2.3 Comparative summary

Table 1 summarizes the accelerometers presented in this chapter[1], [3], [14], [16]–[19], categorizing them by noise floor, measurement range and interrogation method. While the capacitively interrogated devices [14], [16] demonstrate the lowest noise floor in the seismic range (0-100Hz), optically interrogated SiN devices [1], [3] show equally good performance in a higher frequency range. This higher range can be attributed to their higher natural frequency. The silicon flexure optically interrogated device [17] has performance equal to that of the capacitive device in the seismic range; this is probably due to its lower stiffness than tensile stress SiN devices, allowing for a lower natural frequency. However, in our approach, these low

frequencies can also be obtained by increasing the mass loading in SiN based devices. Finally, resonant devices [18], [19] also have desirable performance in the seismic range and use resonators coupled to a suspended proof mass to convert the input acceleration into an inertial force.

Table 1: Noise floor, range, and interrogation method of relevant works

Work	Noise floor	Range	Interrogation Method
J. Laine and D. Mougenot [14]	**30 ng/√Hz**	**1-100 Hz**	**Capacitive**
Zhang *et al*. [23]	**51.8 ng/√Hz**	**0.01-100 Hz**	**Capacitive**
F. Zhou *et al*. [3]	**32 ng/√Hz**	**5 - 11.8k Hz**	**Optical**
G. Krause *et al*. [1]	**10 µg/√Hz**	**25 kHz**	**Optical**
Fan *et al*. [2]	**NA**	**NA**	**Optical**
Z. Qu *et al*. [17].	**100 ng/√Hz**	**<31 Hz**	**Optical**
X. Zou, P. Thiruvenkatanathan *et al*. [18]	**3.2 µg/√Hz**	**<5 Hz**	**Resonant**
M. Pandit *et al.* [18]	**17.8 ng/√Hz**	**<5 Hz**	**Resonant**

2.4 Silicon Nitride (SiN) Membrane Fabrication Techniques

As discussed in section 2.2, SiN thin-film membranes and other SiN thin-film structures are good candidates for mass-loaded accelerometry. The fabrication of SiN membranes involves several critical steps, including deposition, patterning, etching, and post-processing.

SiN deposition methods play a crucial role in determining the quality and characteristics of the membranes. Chemical Vapor Deposition (CVD) is a commonly used technique that allows for the growth of high-quality SiN films. Low-pressure CVD (LPCVD) and Plasma-Enhanced Chemical Vapor Deposition (PECVD) offer excellent control over film thickness, stoichiometry,

and stress. For instance, Sharma et al., [24] utilized LPCVD to deposit uniform and low-stress SiN membranes for MEMS applications. PECVD has also been employed for the deposition of SiN films with precise control over film properties [25].

Patterning SiN membranes is a crucial step in defining their geometries and structures. Photolithography techniques, including electron-beam lithography and UV (Ultra-Violet) lithography, have been extensively employed for patterning SiN membranes with submicron-scale features. For instance, [26] demonstrates the use of electron-beam lithography combined with lift-off process to achieve well-defined SiN membranes with precise pattern control. On the other hand, UV lithography has been used to fabricate SiN trampolines [13].

Etching processes are essential for the release of SiN membranes from the substrate. Wet etching methods, such as phosphoric acid or potassium hydroxide (KOH) etching, offer simplicity and isotropic etching characteristics. For instance, a KOH solution was utilized to selectively etch a sacrificial layer and release SiN membranes [27]. Dry etching techniques, such as reactive ion etching (RIE) and inductively coupled plasma (ICP) etching, provide better control over etch rates and sidewall profiles. ICP etching has been employed to fabricate ultra-thin SiN membranes with high precision and low roughness [28].

Post-processing steps are employed to enhance the mechanical properties and surface characteristics of SiN membranes. Annealing at elevated temperatures has been found to reduce film stress, improve crystallinity, and enhance mechanical stability. For instance, an investigation into the effects of annealing on the mechanical properties of SiN membranes demonstrated improved mechanical stability with reduced residual stress [29]. Deposition of coatings, such as silicon dioxide (SiO_2) or polymeric materials, can enhance strength, chemical

resistance, and surface functionalization. Deposited SiO_2 coatings on SiN membranes, can also improve their radiation stability [30].

SiN membrane fabrication techniques have witnessed significant progress, driven by the demand for high-performance devices in various applications. Deposition techniques have been refined to achieve high-quality SiN films with desired properties [24], [25]. Patterning and etching methods have been developed for precise membrane geometries and release from the substrate [13], [26], [27], [28]. Post-processing steps and surface modifications enhance mechanical properties and surface characteristics [29], [30].

2.5 Mounting Methods for MEMS Resonators

The successful integration and mounting of MEMS resonators are crucial for their optimal performance and reliability. This section provides an overview of the state-of-the-art mounting methods for MEMS resonators, including adhesive bonding, flip-chip bonding, wafer-level packaging, and anisotropic conductive films.

Adhesive bonding is a widely used mounting method for MEMS resonators. Epoxy-based adhesives, such as SU-8, have been commonly employed due to their high bonding strength and compatibility with MEMS fabrication processes. [31] investigated the effects of epoxy adhesive bonding parameters on the performance of MEMS devices, highlighting the importance of optimizing adhesive thickness and curing conditions. Other studies have explored the use of low-temperature bonding techniques, such as anodic bonding and wafer-level bonding, to minimize thermal stress and achieve hermetic sealing [32].

Flip-chip bonding offers precise alignment and high bonding strength by directly connecting the MEMS resonator to the substrate or packaging platform. Gold-to-gold thermocompression bonding is a common technique used for flip-chip bonding [33].

Additionally, solder-based flip-chip bonding techniques, such as indium bonding, have been explored for their high-temperature stability and excellent electrical conductivity [34].

Wafer-level packaging methods enable batch processing and cost-effective assembly of MEMS devices. Underfill encapsulation techniques using polymers, such as epoxy or polyimide, have been employed to protect the microelectronics. [35] investigated the effects of underfill encapsulation on the performance and reliability of microsystems, highlighting the importance of selecting appropriate materials and optimizing encapsulation process parameters.

Anisotropic Conductive Films (ACFs) could be an alternative mounting method for MEMS resonators. ACFs consist of conductive particles dispersed in an insulating adhesive matrix and provide both electrical and mechanical connections. [36] explored the feasibility of ACF bonding for MEMS devices and demonstrated the potential of this method for reliable and cost-effective packaging. Further research is focused on optimizing ACF formulation, interconnection density, and bond strength for various MEMS designs and applications.

Mounting methods play a critical role in the successful integration and performance of MEMS resonators. Adhesive bonding, flip-chip bonding, wafer-level packaging, and anisotropic conductive films are among the commonly employed mounting techniques.

3. Design & Fabrication

3.1 Fabrication Process Overview

High Q-factor SiN resonators are typically fabricated using KOH etching [37]. Our proposed design relies on the same process but leaves a silicon mass still attached to the SiN resonator (Figure 13 a). The typical standard SiN resonator fabrication process for a rectangular membrane is shown in Figure 13. Starting with a silicon wafer with a thin layer of SiN on the surface that has been pre-deposited, it is coated in PR (photoresist) that is patterned with UV light using a photomask (Figure 13 a-c). The UV exposed resist is removed threw development to expose the SiN layer underneath as shown in Figure 13 d. The exposed SiN is then removed via RIE (reactive ion etching) before the final step of hot KOH (Potassium hydroxide) etching (Figure 13 e-g). During hot KOH etching the Si under the removed areas of SiN gets etched away to free the final SiN thin film structure.

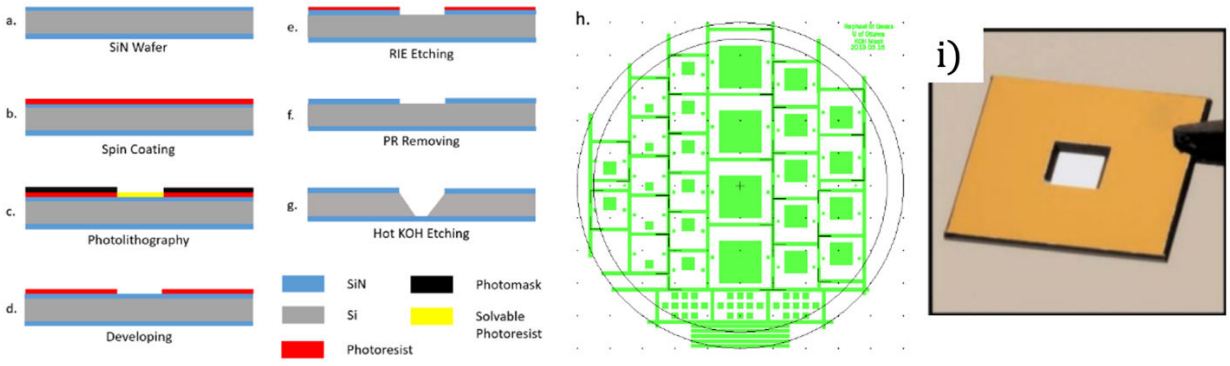

Figure 13: (a-g) General Procedure of Standard Lithographic Method for Rectangular Shape SiN Membrane Fabrication (i.e., our standard process, without proof mass). (h) Photomask for Membranes. (i) Rectangular membrane [37].

Our approach is to adopt a similar fabrication process but to leave a piece of Si substrate still attached to the final SiN shape (Figure 4 a). Utilizing a Si wafer that is coated with SiN on both sides ensures that the released Si mass will be fully supported. On the bottom plane the proof

mass is supported by a full rectangular membrane like the ones described above. On the top surface a SiN trampoline structure will support the proof mass. This fabrication process will be presented later in Figure 16.

3.2 Photomask Design

The accelerometers final geometry is governed by the photomask used to pattern the PR coated wafer during exposure to UV light shown in Figure 13 c. In the case of rectangular SiN membranes, the photomask simply has a rectangular cutout that is roughly the same size as the final membranes (Figure 13 i). This allows for the removal of the entirety of the SiN on the top surface of the wafer that is inside this rectangular area, ensuring that all the Si above the final SiN membrane is removed during the KOH etching (Figure 13 c). On the other hand, the accelerometers must have some SiN on the top surface of the wafer to preserve the proof mass during the KOH etching and to shape the SiN trampoline.

Figure 14 shows examples of regions of the photomask design for one accelerometer of each size produced. The accelerometers feature 4 Bezier curves to outline the shape of a SiN trampoline that will support the proof mass from the top. Bezier curves were chosen since they offer a low potential of undercutting as explained further in the following section. Figure 14 a shows the largest size of accelerometer. It has a 3×3 mm proof mass suspended by a 6×6 mm SiN membrane on the backside, all within a 18×12 mm chip. The two other accelerometers (Figure 14 b-c) both have square chips with side length of 12 mm and 9 mm respectively. They feature 4×4 mm and 3×3 mm membranes with 2×2 mm and 1.5×1.5 mm proof masses.

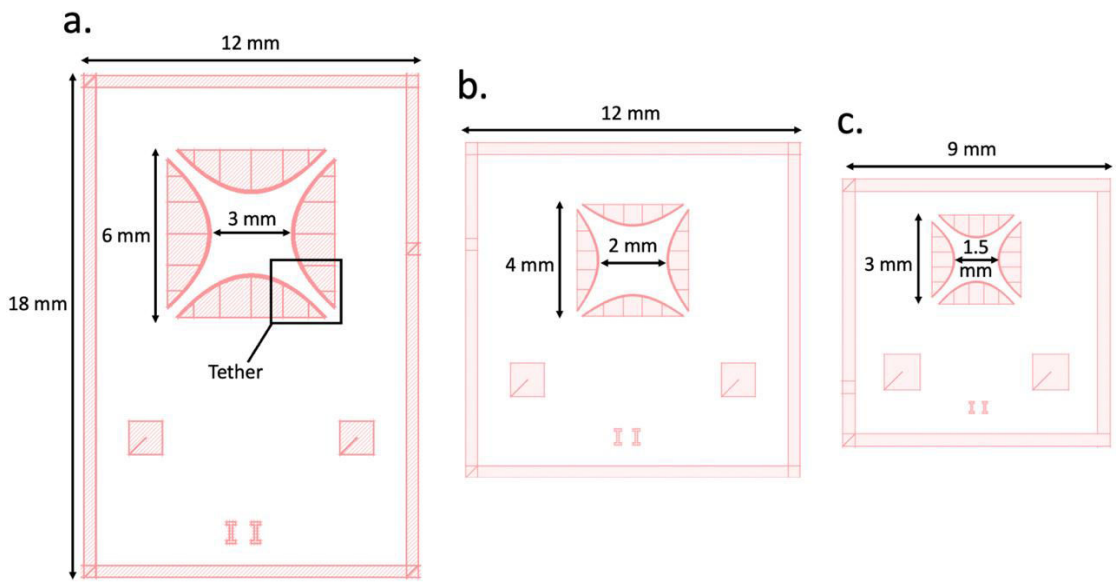

Figure 14: Accelerometer Chip examples. (a) 6 × 6 mm accelerometer. (b) 4 × 4 mm accelerometer. (c) 3 × 3 mm accelerometer.

The full photomask design is shown in Figure 15, It has 22 distinct accelerometer designs with 2 copies of each design. The channels between each chip are 425 µm wide to be able to split the accelerometers into individual chips during fabrication. There are alignment rectangles on the bottom part of the photomask to help align it with the wafer's straight edge (i.e., the silicon crystal plane) during photolithography. The mask also features alignment crosses near its center and each chip also has its own alignment squares to give the user the possibility to line up the mask with a previously exposed section of wafer in the future, if necessary. The photomask is designed using the GDSPY library in python as it has several advantages over a GUI centric software such as Tanner Tools L-Edit. Most notably, one can implement complex shapes such as parametric curves and easily iterate over different geometries.

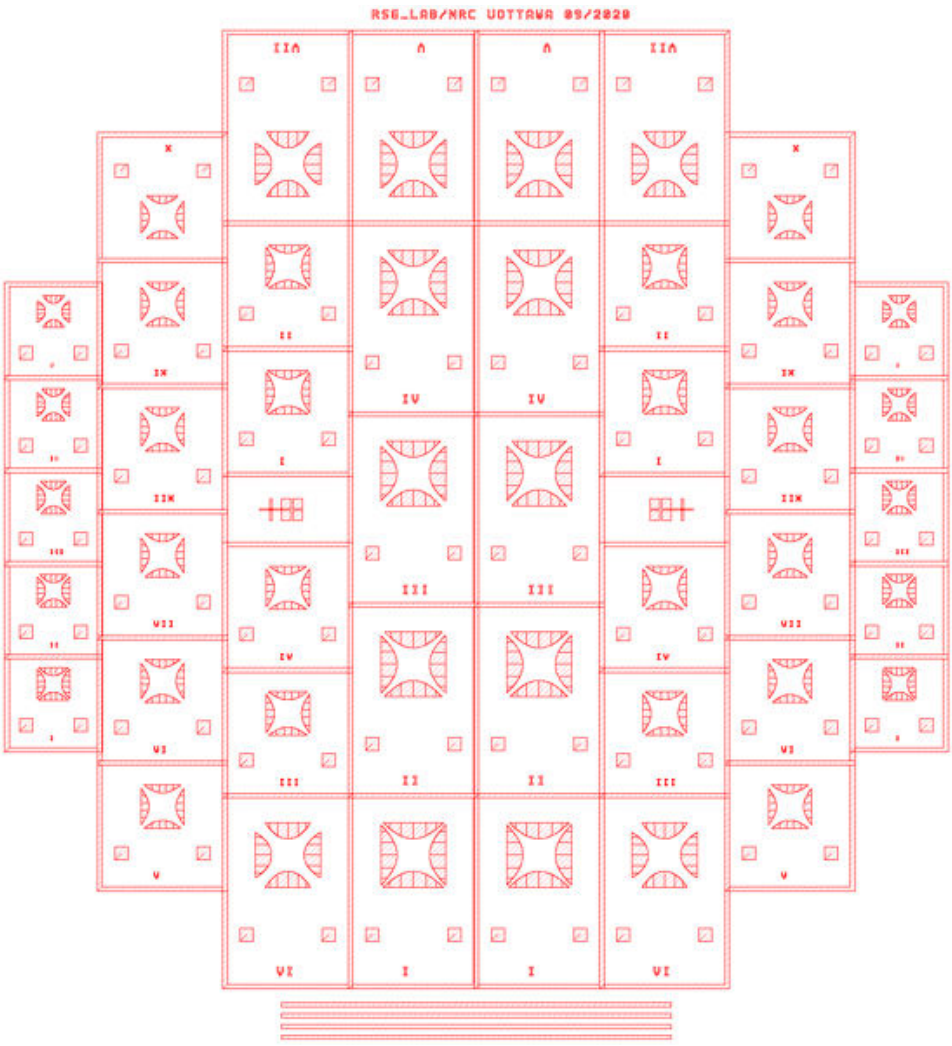

Figure 15: Photomask Design

Each chip size has several versions on the photomask with varying tether widths (see tether in Figure 14 a). For each size of accelerometer, a range of tether width is defined. Each pair of accelerometers has a roman numeral indicating their tether width starting at I for the minimum tether width for that accelerometer size. Table 2 shows the range of tether widths and number of design widths for each size of accelerometer. Each accelerometer design width is included twice on the photomask for redundancy.

Table 2: Range of tether widths and number of pairs for each size of accelerometer

Accelerometer Identifier	6 × 6 mm Accelerometer tether width	4 × 4 mm Accelerometer tether width	3 × 3 mm Accelerometer tether width
I	300	200	200
II	500	261.11	337.5
III	700	322.22	475
IV	900	383.33	612.5
V	1100	444.45	750
VI	1300	505.56	750
VII	1500	566.67	
IIX		627.78	
IX		688.89	
X		750	

The range of tether widths of the accelerometers is determined mainly with consideration to the last step of fabrication, the hot KOH etching. As previously discussed, (Section 3.1) the KOH etches the Si. However, it does also etch the SiN at a comparatively slow rate of [38]. Therefore, it's ideal to have a shorter overall etching time and SiN features that have similar etching rates. If some features have lower etching times than others, it leads to the risk of having some features etch faster than others. This is undesirable since the features that etch faster have both sides exposed to KOH and are submerged until the other features finish etching (Figure 18). It is hard to predict the exact outcome of the etching process with a specific geometry, therefore several tether widths and corresponding Bezier curves were tested on a single photomask.

3.3 Detailed Fabrication Process

A critical requirement of this thesis is the in-house fabrication of the accelerometers for which the design has been presented in the previous section. The current section presents the detailed fabrication process. The process starts with a four-inch single-crystal substrate, 525 µm thick silicon wafer oriented in the <100> lattice plane according to the miller convention, coated

on both sides with 100 nm low-stress low pressure chemical vapor deposition (LPCVD) SiN (Figure 16 a). These substrates were purchased commercially, with the LPCVD coating already on. The fabrication process can be subdivided into 4 overarching steps: spin coating, photolithography, radiative ion etching and KOH etching (Figure 16).

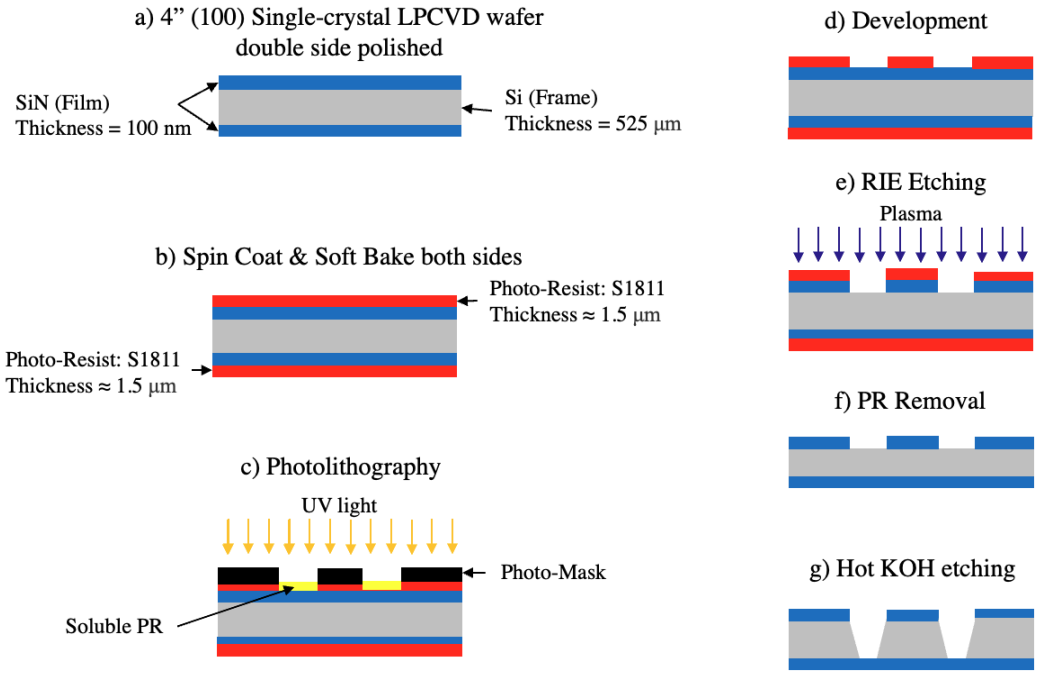

Figure 16: Schematic of Accelerometer Fabrication Process

The wafer is first spin coated with positive S1811 photoresist on both sides (Figure 16 b). Positive PR is a polymer that once cured, becomes soluble to certain developers after being exposed to UV light. More than the desired amount of PR to coat the wafer is placed at its center, before putting it into the spin processor. The spin processor uses a vacuum powered chuck to hold the wafer in place while spinning it to allow centrifugal forces to spread the PR evenly over its surface and remove the excess PR. To control the final thickness of PR on the surface of the wafer we follow the spin speed curve for our specific type of photoresist. An example of such curves can be found in Figure 17. The final recipe used in the spin coating consists of 10 seconds of acceleration up to a speed of 2000 RPM, followed by 60 seconds of constant speed rotation

and 20 seconds of deceleration back to 0 RPM, resulting in a final layer of PR that is approximately 1.5 µm thick.

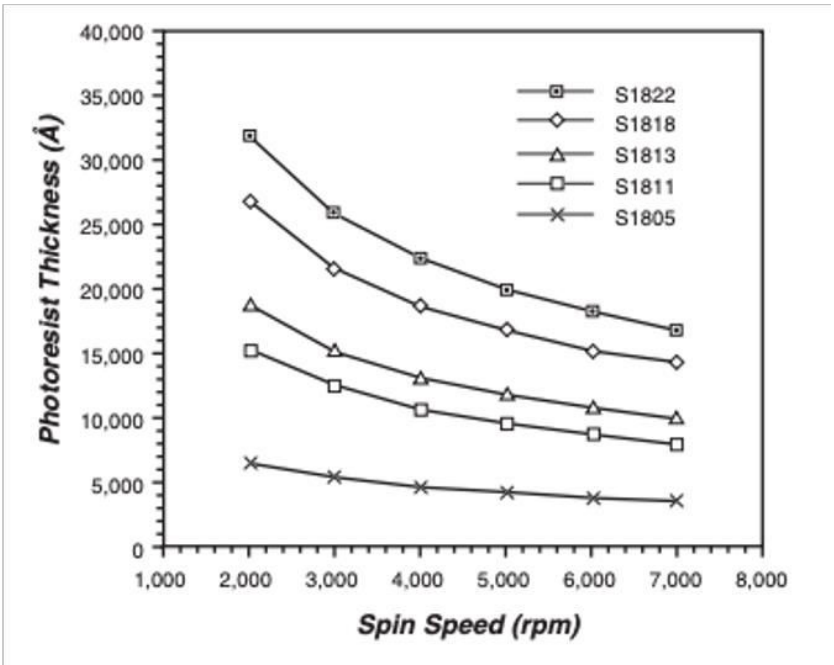

Figure 17: Spin Speed Curves for Photoresists including S1811 [39]

Once the frontside of the wafer is coated in PR it is placed on a heating element at 115 ℃ to cure for a minute. This is known as the soft bake. The information for the duration and temperature of this process can be found in Table 3. Once complete, the spin-coating and baking is then repeated for the back side of the wafer. The frontside photoresist coating is used for photolithographic processing, while the backside coating is used solely to protect the SiN membrane during fabrication.

Table 3: Process Conditions for S1813 applicable to S1811

High Resolution Process Parameters (Refer to Figure 1)	
Substrate:	Polysilicon
Photoresist:	MICROPOSIT®S1813® PHOTO RESIST
Coat:	12,300Å
Softbake:	115°C/60 sec. Hotplate
Exposure:	Nikon 1505 G6E, G-Line (0.54 NA), 150 mJ/cm^2
Develop:	MICROPOSIT® MF®-321 DEVELOPER
	15 + 50 sec. Double Spray Puddle (DSP) @ 21°C

Following the soft bake of the backside of the wafer, the frontside is exposed to UV light to pattern the photoresist and thus control the accelerometer's final geometry (Figure 16 c). To this end, the OAI model 204IR mask aligner is used. The mask aligner allows for the positioning of the specifically designed photomask above the wafer, to transfer the photomask pattern into the photoresist using UV light. During the mask aligning process, the photomask must have the chrome side facing the wafer to prevent light from scattering when it crosses the glass part of the photomask. To achieve this, the photomask is pressed against the wafer after it is aligned at its center. This is called "hard contact". It occurs after a screw under the wafer is tightened to bring it up to the photomask and a jet of air is turned on above the photomask to further push it against the wafer. This minimizes any gaps and thus any lights scattering during the exposure. The exposure occurs when a UV light above the photomask is turned on. The exposure is controlled by the power of the UV light and the duration it is exposed. Both control the final amount of energy reaching the PR and thus the thickness that is made soluble. We want all exposed PR to be soluble down to the layer of SiN. We use to Equation 2 to calculate the energy input of the exposure and cross check it with the PR's energy requirements and thickness.

$$Exposure\ Time\ = \frac{Desired\ Energy\ Input}{UV\ light\ power}$$

Equation 2

Desired exposure occurs when the pattern on the photomask is copied perfectly to the PR. Underexposure happens when not enough energy reaches the PR and its entire thickness is not soluble. Overexposure takes place when too much energy reaches the PR and the features are larger on the PR then the photomask. Both under and over exposure are undesirable, the former results in improper SiN removal during RIE and the latter causes distorted features during the RIE etching. The final photolithography process uses 23 mW of UV light for 10 seconds for correct exposure to occur. Following the exposure to UV light, the wafer is immersed in MF-321 developer for one minute to remove the soluble PR that has been exposed (Figure 16 d).

Subsequently, RIE etching is performed using the SAMC RIE-10N to remove the layer of SiN that has been exposed by the development of the PR (Figure 16 e). RIE etching uses plasma formed by various inlet gases to attack the SiN layer on the wafer. In this case oxygen and CF_4 (Carbon tetrafluoride) are used to attack the SiN. The time, pressure and quantity of gases are carefully controlled according to the developed recipe as to insure the removal of only the desired SiN features. In this case, 25 SCCM (Standard cubic centimeters per minute) of CF_4 and 5 SCCM of oxygen, at a pressure of 8 pa and a power of 100 w for two minutes. After the RIE, the wafer is rinsed in acetone, IPA (isopropyl alcohol) and DI (de-ionized) water to remove the remaining PR (Figure 16 f).

Following the rinse, hot KOH etching is used on the exposed Si substrate to release the suspended proof mass (Figure 16 g). While all the previous steps were relatively conventional and straightforward, KOH etching of such delicate structure was by far the most custom and challenging process step. We thus discuss it in greater details below.

The KOH etching is performed in several steps. Initially the entire wafer is placed in the hot KOH bath at 85 ℃ for two hours. The goal of this step is to etch down the channels in

between the chips to half of their original thickness (Figure 18). Following this, the wafer is removed from the KOH bath and placed in DI water. This allows for the wafer to be broken up into individual chips by splitting it along the channels via the application of bending pressure with tweezers right along the channel. After the individual chips have been separated, they are placed in a custom holder (Figure 19) to be returned to the KOH bath at a less aggressive etching temperature of 75 ℃. The lower temperature ensures the KOH is more selective in its etching of the substrate then the thin film. Since more released SiN is exposed during the second half of etching, the lower temperature allows it to keep closer to its full thickness.

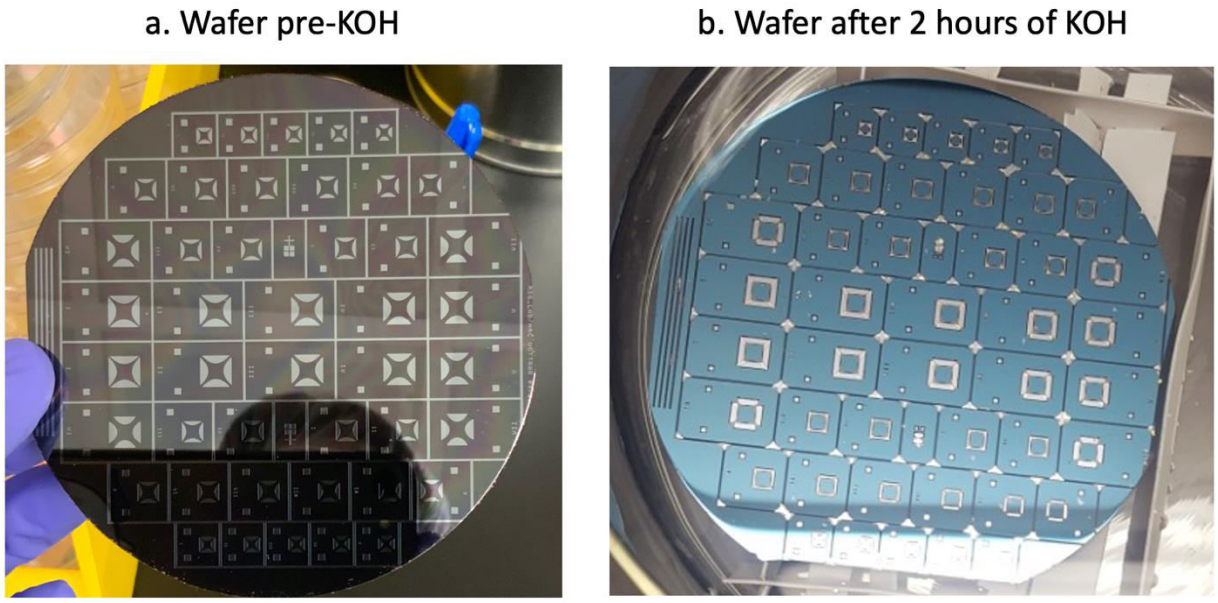

Figure 18: Full wafer before and after 2 hours of hot KOH etching

A custom PTFE (Polytetrafluoroethylene) holder for holding the chips during the second step of the etching is shown in Figure 19. After approximately two hours of etching in this holder, the chips are removed from the bath once the bubbles produced from the KOH etching are no longer visible around the desired featured, indicating that the etching is complete.

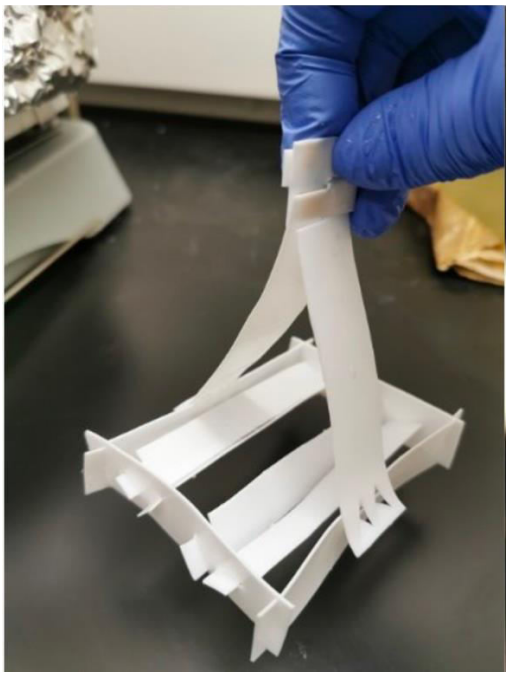

Figure 19: Original custom PTFE holder

3.4 Discussion

Several challenges were encountered during the initial fabrication runs. The first challenge was the segregation of the wafer into individual chips after the initial 2 hour etch of the wafer. It proved difficult to split the wafer along the channels between the chips (see Figure 18). This indicates that our initial designed width of 425 µm is insufficient. Since the etching of the Si frame happens along the crystal planes of the Si, the width of the channel drops as it is etched deeper as shown in Figure 20. A wider channel would ensure that the wafer is easier to split, as a wider piece of the channel is etched to half its original thickness. A better channel width for the next photomask design will be 1000 µm. This is the same width a colleague recently used in the fabrication of a batch of plain membranes and the separation proved to be much more successful than for the accelerometers.

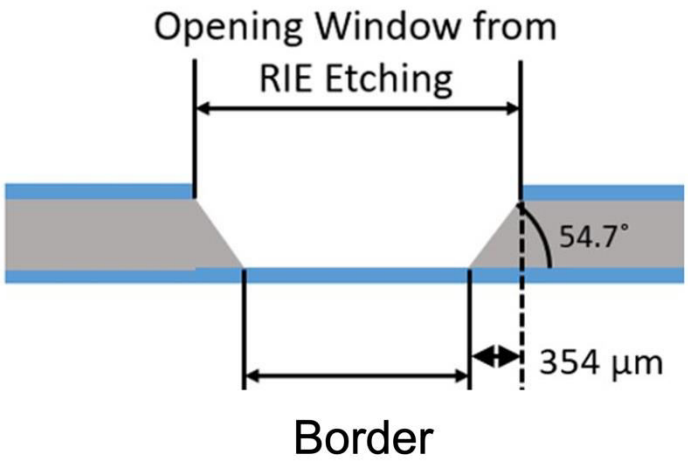

Figure 20: Boarder width between chips during KOH etching

The next obstacle encounter during the hot etch was floating of the chips in the holder during the second step of the etching, when the chips are separated and returned to the KOH. During the etch, hydrogen bubbles form as a by-product of the reaction with Si. These bubbles can make the chips float around in the holder. Since the accelerometers have a SiN trampoline on the top plane, it traps hydrogen bubbles and consequently the devices float around in the KOH solution. Because of the floating, the SiN features on the top and bottom plane of the accelerometers often collide with the sharp Si frame of another chip and break. To prevent this issue, we conducted a test using a new sample holder that has a PTFE strip placed on top, which is screwed in lightly with PTFE screws (see Figure 21). The purpose of this modification is to prevent the samples from floating around during the fabrication process. Unfortunately, it was discovered that the PTFE strip is too soft and not strong enough to hold the samples in place. As a result, the chips still floated around, and the test was not successful. Although the modification did not work as intended, this experience provided valuable information that can be used to improve future sample holder designs.

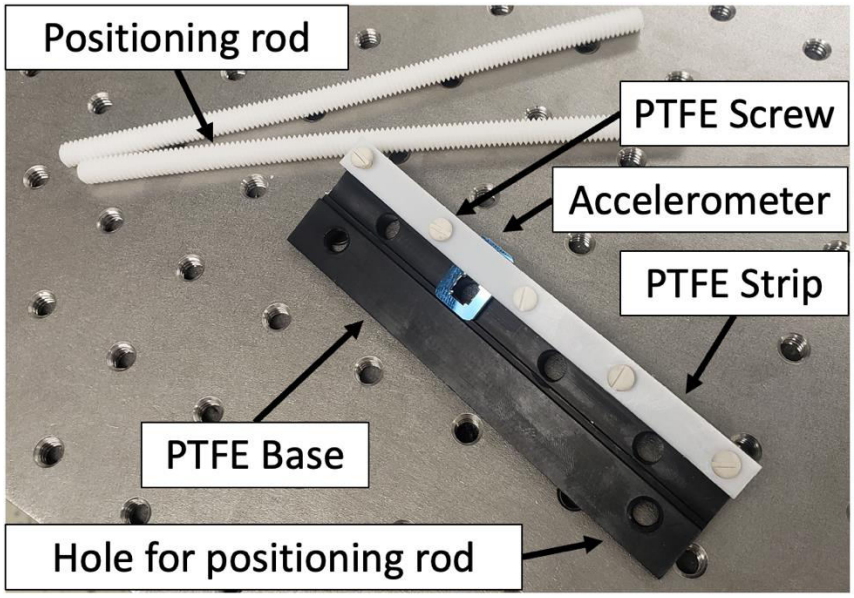

Figure 21: New KOH holder

Based on our recent experiment, the next time we design a sample holder, we will not use a strip that needs to be screwed in place. Instead, we plan to incorporate inset holes in which to place the accelerometers at an angle, which will prevent them from floating around during the etching process. The holder will also be designed as one piece, eliminating the use of screws that made the previous holder more difficult to use. By using inset holes, we can ensure that the chips are held firmly in place, reducing the risk of any damage occurring during the fabrication process. Additionally, the one-piece design will make it easier to use and eliminate the need for any additional assembly.

Figure 22 illustrates an additional challenge encountered during the KOH etching process, which is the insufficient under-etching of the top tethers (see Figure 14) of the accelerometer. Insufficient under-etching occurs when the substrate has been completely etched through, but some SiN features, specifically those surrounding the tethers of the accelerometer, still have Si attached. To achieve the desired, under-etch, longer etching times are required for some of the samples. However, this may not be beneficial for the SiN membrane and

trampoline's thickness, as they have a higher probability of being damaged by KOH. In several cases, the etching rate decreased significantly after the Si substrate had been etched through its entire thickness, compelling us to halt the process to safeguard the SiN features. Consequently, sharp Si corners were created at the junctions of the SiN tethers and frame, resulting in sharp bending points that could lower the quality factor [40] due to increased bending radius. These corners are formed when the Si wafer planes take longer to meet during etching (indicated by black lines in Figure 22). Since accelerometers with different tether widths were manufactured on the same wafer, we were able to observe the effect of tether width on under-etching. The most successful accelerometers during fabrication were those that could etch through the 525 um Si frame and the top tethers simultaneously (as seen in Figure 22 a).

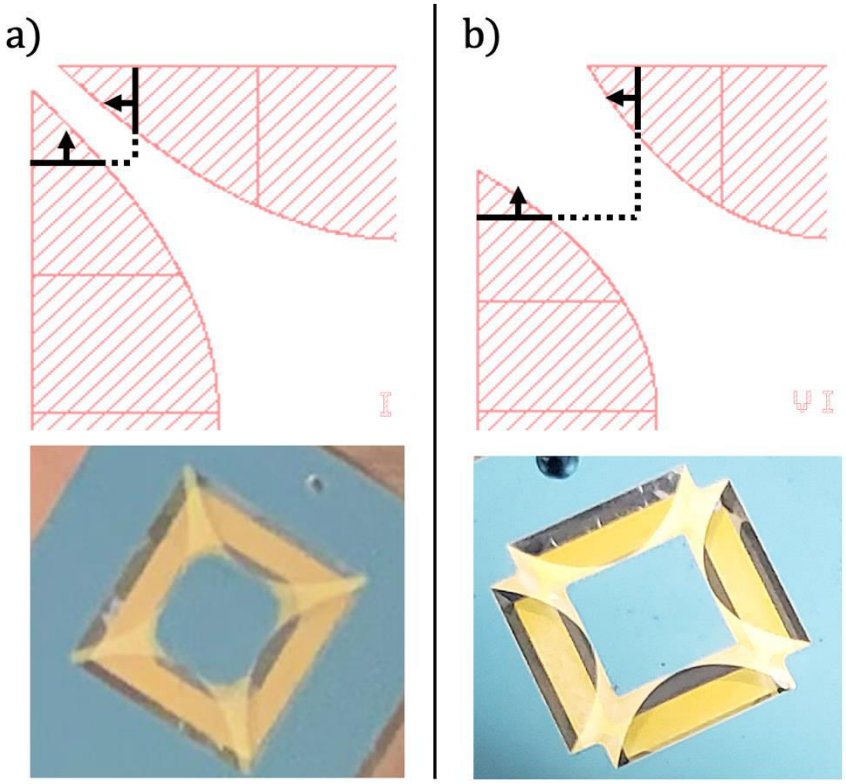

Figure 22: Illustration of Under-Etching in the Corners of the Accelerometers. (a) shows a sample that has been completely under-etched. (b) Shows a sample with incomplete under-etch

After the initial batch of fabrication, it is worth noting that the accelerometers with narrower tethers had a better success rate. Of the 6×6 mm accelerometers those numbered I, II and III (see Table 2) were the only truly viable samples with full under-etching around the corners of the tethers. These samples had tether widths ranging from 300 to 650 um. Of the 14 large accelerometers on the wafer only four were deemed usable, all with identifier III or lower. Surprisingly the small accelerometers proved unsuccessful. This was in large part due to their siz making them difficult to manipulate during fabrication.

4. Experimental characterization

4.1 Experimental setup

To quantify the fundamental thermomechanical limit of our devices we measure the Q-factor and the natural frequency of our device using a fiber-interferometer. To achieve this, the samples are placed in a custom-made chamber under vacuum to eliminate air damping. The vacuum chamber can reach pressures below 10^{-4} Pa through continuous turbo molecular pumping. The samples are mounted on a glass slide or a piece of low-carbon steel, which is then attached to an optical stage using double-sided tape, as shown in Figure 23. The optical stage is connected to a positioner that can move laterally to position the accelerometer in front of an optical fiber (positioner 2 in Figure 23). The optical fiber is mounted on two other optical positioners (positioner 1 and 3 in Figure 23), which moves toward or away from the sample. The first positioner allows for precise alignment of the sample relative to the laser spot, while the second positioner allows the laser to be brought in proximity to the sample to minimize optical losses.

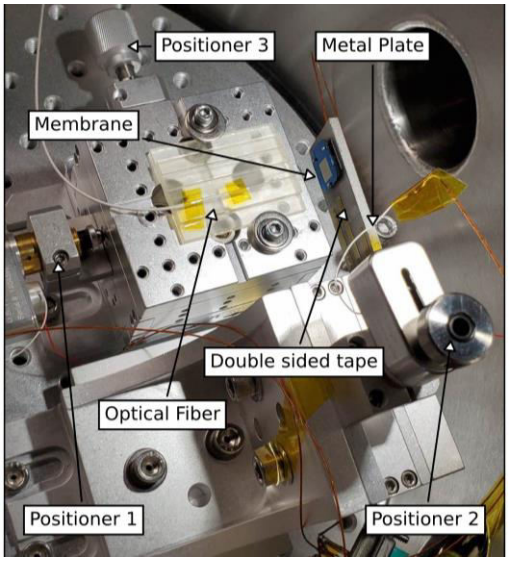

Figure 23: Top view of optical stages and sample in the vacuum chamber

The movement of our resonators is measured using a Fabry-Perot laser interferometer similar to [41] and schematized in Figure 24. The interferometer relies on a narrow line width laser (RIO ORION, RIO0175-3-00-1) that gets fed into a 90:10 optical coupler. The coupler splits the signal into two, with the stronger portion going to an unused port and the weaker portion going towards the sample. As the laser light reaches the sample it gets fed through a cleaved fiber tip that reflects around 4 % of the laser light immediately to the photodetector back through the coupler. The rest of the light makes its way to the sample and is then reflected towards the photodetector. The interferometer relies on the optical interference between the light reflected by the fiber tip and that reflected by the accelerometer to determine the displacement of the latter.

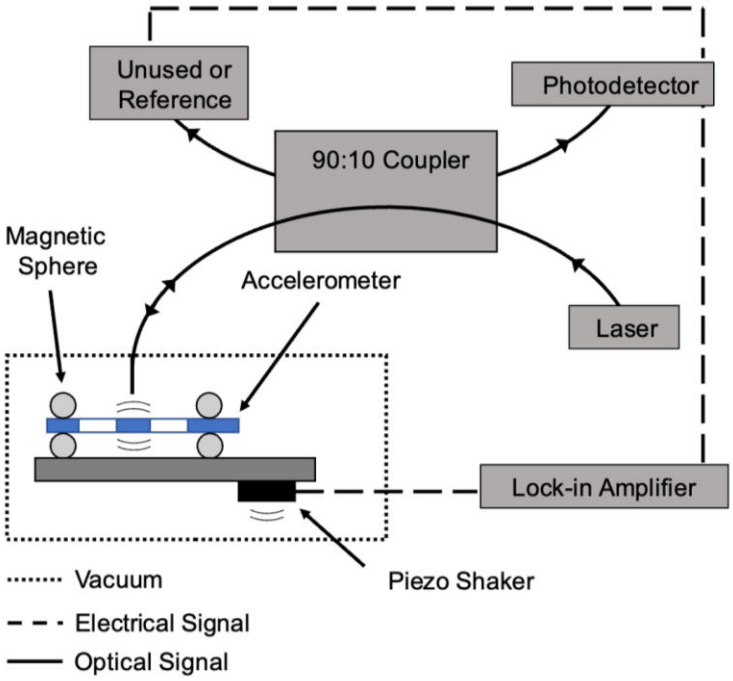

Figure 24: Experimental schematic of the optical detection and electronic actuation systems for characterizing the accelerometers vibration.

The resulting interference pattern is depicted in Figure 25, where the output voltage is proportional to the distance (*d*) between the fiber tip and the sample. The two signals of the interferometer are denoted in Figure 25 (a) as 1 and 2. Signal 1 reflects off the fiber tip and back towards the photodetector. Signal 2 travels to the accelerometer then is reflected towards the photodetector. The interference pattern formed by these two signals is show in Equation 3. It provides the relationship between the voltage read from the photodetector, and the distance between the proof mass of the accelerometer and the fiber tip:

$$V = V_{mid} + (V_{max} - V_{min})\, cos(\frac{4\pi d}{\lambda}).$$

Equation 3

As shown in Figure 25 (b), where the reflection is analogous to the voltage read by the photodetector (V), it can be observed that this type of interferometer has a limited linear region. This refers to the section of the sinusoidal pattern that exhibits a quasi-constant slope. The limitations of this interferometric scheme are twofold. There is no way of extracting the direction of movement of the sample from the output voltage, and the range of displacement of the readout is limited to a quarter of the wavelength of the laser (i.e., approximately 390 nm in the present case).

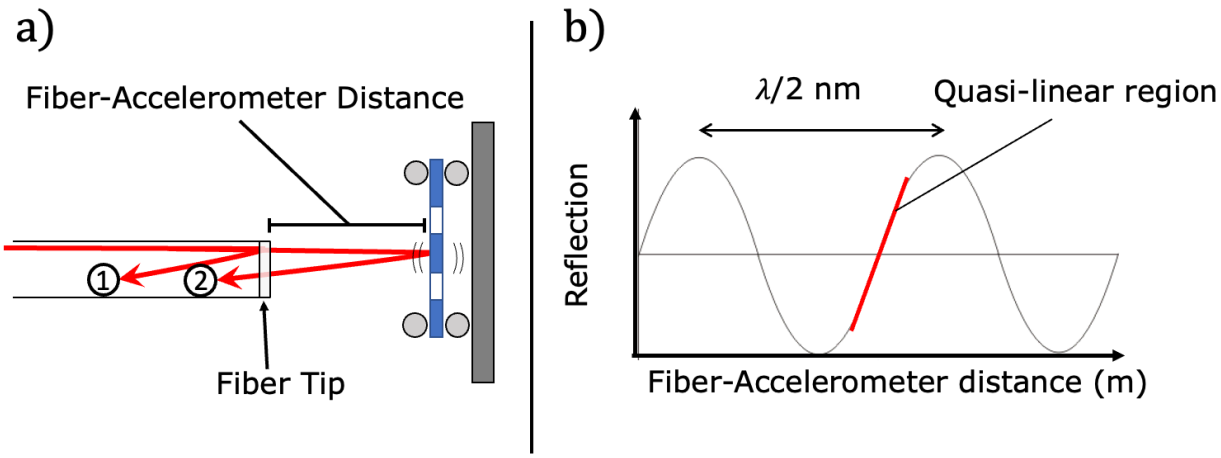

Figure 25: Relation between the distance of the fiber tip to the sample and the readout voltage. (a) Sample-fiber arrangement. (b) Interference pattern.

By utilizing a differential photodetector that obtains its signal from both the unused port and the detector port (see Figure 24) our interferometer achieves its lowest noise floor, shown in Figure 26. The noise density is smaller than 1 pm/√Hz at frequencies above 1 Hz and generally follows a 1/f trend, which is typical of laser phase noise. The differential port is used only when evaluating the performance limit. In other cases where the accelerometer is strongly actuated, single-ended detection (i.e., unconnected reference port in Figure 24) is sufficient and simpler to implement.

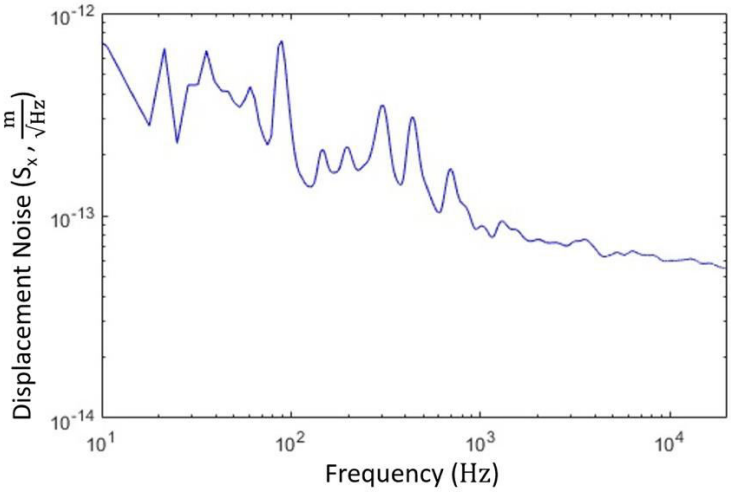

Figure 26: Displacement Noise using balanced photodetector.

The displacement noise floor (S_x) of the photodetector (Figure 26) and the fundamental mode of our device (ω_0) are used to infer are readout noise floor ($S_{a,readout}$):

$$S_{a,readout} = S_x * \omega_0^2.$$

Equation 4

Figure 26 shows the inferred readout noise floor ($S_{a,readout}$), as well as the noise floor from a group that employed a similar interferometer and purported to be the shot noise of the photodetector [41].

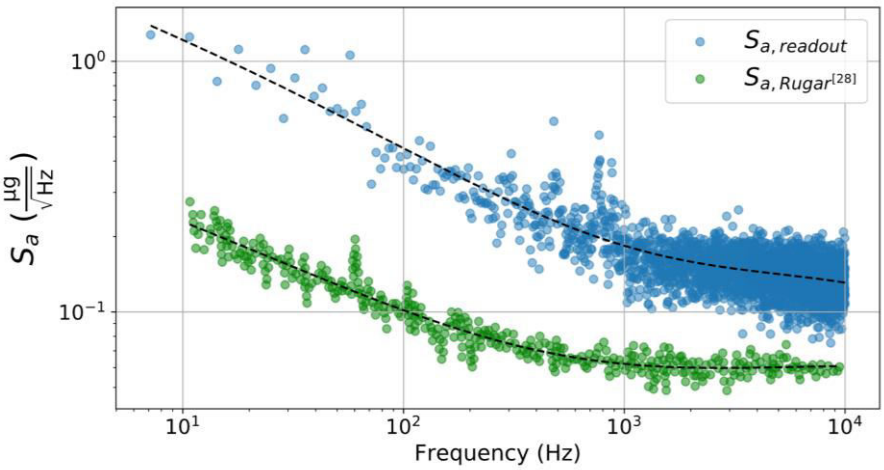

Figure 27: Our inferred noise floor compared with Rugar [41].

From Figure 27 we see our readout noise floor is approximately 1 $\mu g/\sqrt{Hz}$ at a frequency of 10 Hz. In section 4.3 we discuss the thermomechanical noise floor ($S_{a,mech}$) in comparison with the readout noise floor ($S_{a,readout}$). The readout noise floor is a result of the interferometer and devices used to measure its signal, while the thermomechanical noise floor is property of the accelerometer itself. From Equation 1 we know that the thermomechanical noise floor depends on the Q-factor of the accelerometer, which will be experimentally determined in the next section.

4.2 Q-factor measurement for various mounting methods

To verify the thermomechanical noise limit of our devices using Equation 1, we must measure their Q-factors. To determine the Q-factor, a ringdown test was performed by actuating the accelerometer at its resonance frequency and then abruptly interrupting the actuation signal.

The time it takes for the resonance to decay is then measured. This test determines the characteristic "ringdown" time, represented by the variable τ, which can be used to calculate the quality factor using:

$$Q = \frac{\tau \cdot \omega_n}{2}$$

Equation 5

with ω_n the natural frequency and Q the Q-factor.

Initial characterization was performed using a custom sample mount (see Figure 27 a) intended for use in a portable vacuum chamber. This custom mount holds the fiber tip in front of the sample and has a plate that is screwed in on top of the sample. The accelerometer under test is first glued to the top plate of the holder using nickel paste (Ted Pella, 16059) and then the top plate is screwed into place on the holder with spring loaded screws to reduce the force acting on the plate.

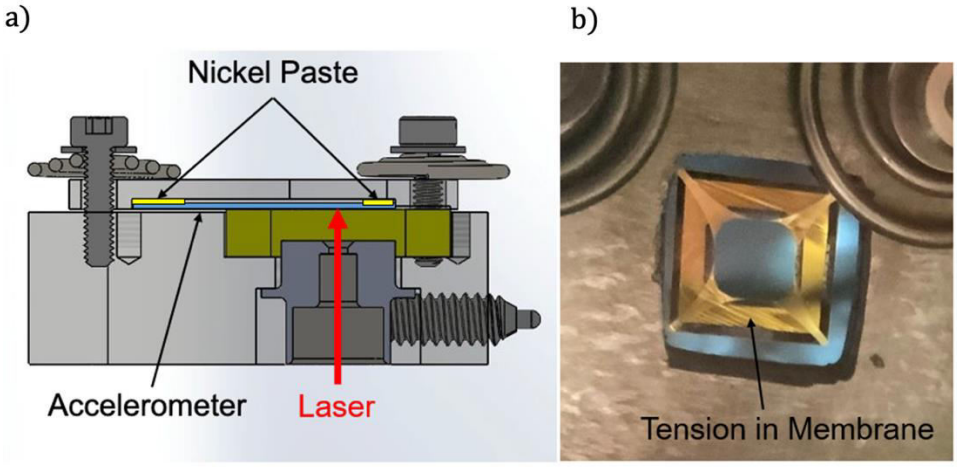

Figure 27: Spring loaded mount for portable vacuum chamber. (a) Solidworks Schematic. (b) Mounted Accelerometer.

Although promising, the holder proved ineffective for characterizing accelerometers and no mechanical resonance peaks could be found. This was likely due to the tension present in the chip that deformed the membrane when it was mounted using this holder (see Figure 27 b). The

uneven leveling of the nickel paste and pressure from the top plate likely caused the Si frame of the accelerometer to bend, inducing tension in the membrane.

The initial test revealed that the method used to mount the sample plays a crucial role in the performance of accelerometers. For the next test, a corner of the accelerometer was mounted using Crystalbond^{TM} 509 an organic adhesive on a glass slide (as shown in Figure 28). The 6 × 6 mm accelerometer identified as Roman numeral II (see Table 2) had a natural frequency of 516 Hz and a Q-factor of 768. Despite the Q-factor being lower than expected (e.g., a plain membrane has $Q \sim 1E6$), the natural frequency of the accelerometer (516 Hz) was within the range expected from our model (see Chap 5). We attributed the relatively low Q-factor to dissipation by the CrystalBond adhesive and therefore investigated other mounting methods.

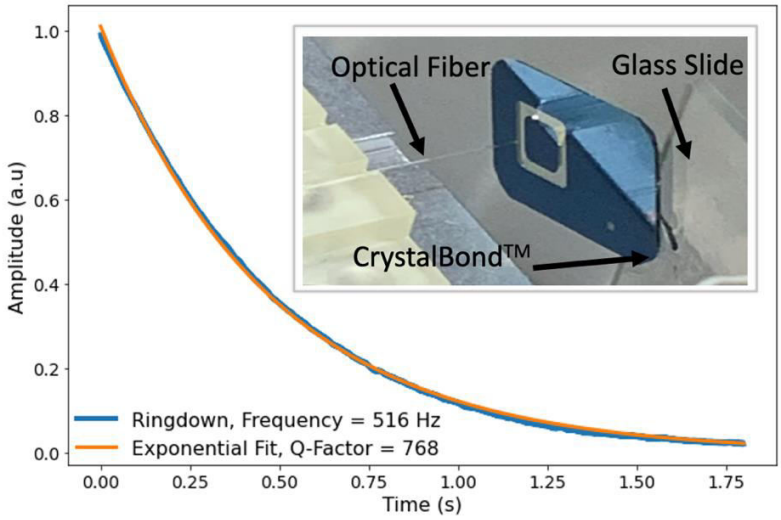

Figure 28: Accelerometer mounted using CrystalbondTM 509 on a glass plate.

To improve the quality factor, a three-point magnetic mounting method was tested. This involves using three spherical magnets placed underneath the accelerometer on a piece of low carbon steel (replacing the glass slide), and three spherical magnets on top of the accelerometer to hold it in place (as shown in Figure 29). These magnets are more rigid than the Crystalbond organic adhesives and minimize contact with the sample; two factors generally expected to

decrease damping in high Q resonators [42]. The accelerometer tested was a 6 × 6 mm and identified by roman numeral III (see Table 2). It has a natural frequency of 526 Hz and a quality factor of 17,950. These results show that Q-factor of devices is heavily influenced by chip mounting since we demonstrate an improvement in the Q-factor of nearly two orders of magnitude simply by changing the mounting method.

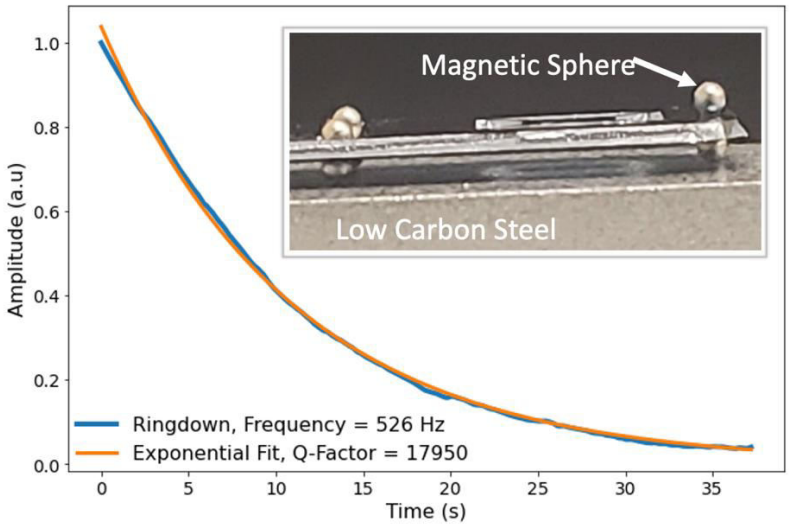

Figure 29: Accelerometer mounted using six magnetic spheres on low-carbon steel.

From these results, a question naturally arise: Does the Q-factor achieved with the most successful mounting method (i.e., Q = 17,950) represent the ultimate limit of our device, or are performances still limited by mounting? Despite the use of tensile SiN, which typically yield high Q-factors in non-mass loaded devices, the devices in this experiment did not achieve multi-million Q-factors. Notably, previous experiments with plain membranes fabricated from the same LPCVD-coated substrate batch achieved Q-factors exceeding 1 million [43], suggesting that the substrates used in this experiment are not responsible for the observed lower Q-factors. It is plausible that mass loading these thin films led to a significant decrease in the Q-factor of our resonator. Alternatively, it is possible that our Q-factor is still limited by the mounting technique.

In the next section of this thesis, we will attempt to address this inquiry using finite difference simulations.

4.3 Performance Benchmark

Now that we have obtained an experimental Q-factor, we perform an initial benchmark of our resonators by comparing them to previous work [1]–[3] in Figure 31, on the basis of the fundamental thermomechanical fluctuation noise limit (see Equation 1). We find that our device compares favorably to previous work on optically-interrogated, full-substrate proof mass supported by either SiN [1] or graphene [2]. Our device also compares favorably to optomechanical accelerometers with in-plane SiN-supported proof masses [3], but presently falls short relative to the performances of state-of-the-art capacitive silicon-on-insulator (SOI) accelerometers [14].

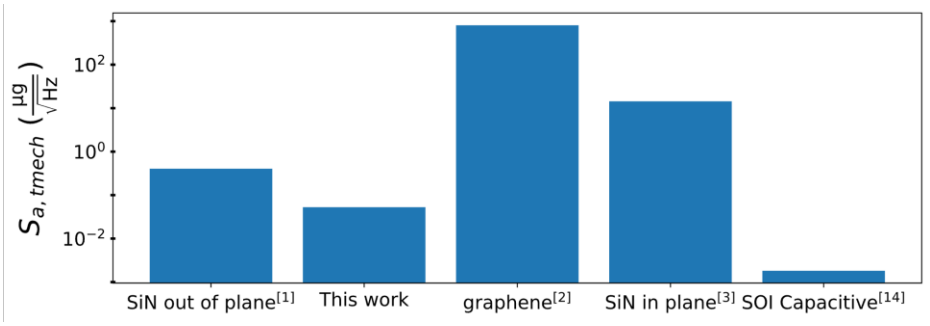

Figure 31: Thermomechanical noise limit of our device("This work") compared with other optically interrogated SiN based accelerometers[1], [3], graphene based accelerometer [2], state of the art capacitive accelerometer [14].

From Figure 31 we see that our device-inferred noise floor is currently limited by our optical readout noise ($S_{a,readout} \approx 1\ \frac{\mu g}{\sqrt{Hz}}$) rather than by thermomechanical fluctuations ($S_{a,tmech} \approx 0.1\ \frac{\mu g}{\sqrt{Hz}}$).

5. Numerical Simulations

To confirm whether the quality factor of our device is currently at its fundamental maximum or if it is limited by our mounting technique, we conduct a one-dimensional numerical simulation. This will predict the maximum mechanical Q-factor without considering the mounting technique. If this simulation matches with our experimental results of Figure 29 our devices have achieved their maximum performance. On the contrary, if the predicted Q-factor is much larger than our measurement, we can conclude that mounting improvements are still necessary. This simulation is aimed at identifying the fundamental mode shape of the device by approximating it to a one-dimensional problem of cross-section shown in Figure 30. This section consists of a SiN top tether supporting a Si proof mass, with the bottom membrane removed (see Figure 27 b). Once we have the mode shape, we can predict material dissipation, which is especially strong at the sharp-bending clamping points, as demonstrated for non-mass-loaded structures [44].

Utilizing a one-dimensional (1D) numerical simulation with a cross-sectional approximation, excluding the membrane on the bottom plane, offers a suitable representation for identifying the fundamental mode shape of the device and predicting its maximum mechanical quality factor (Q-factor). By simplifying the problem to a 1D cross section, we can focus on the crucial aspects that contribute to the overall device performance. This approach allows us to isolate and analyze specific regions of interest, such as the SiN top tether and Si proof mass, where mechanical dissipation, particularly at the sharp-bending clamping points, is known to be significant based on prior studies of non-mass-loaded structures. Additionally, it is worth noting that the damping mechanism in both membranes and strings is fundamentally the same [45]. By considering these factors, we can gain insights into the fundamental limitations of our device and determine if

further improvements are required to optimize its mounting technique and enhance its overall performance.

5.1 Mode Shape Simulation

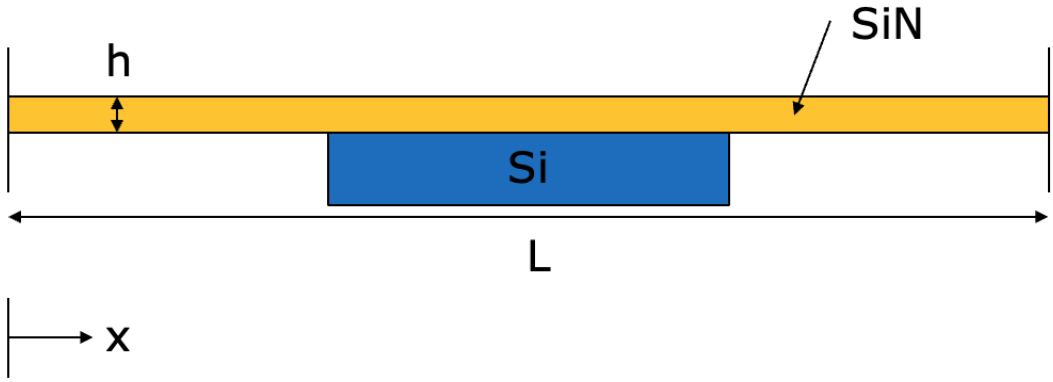

Figure 30: 1 Dimensional model of a SiN stressed string mass-loaded with a Si proof mass.

The SiN top layer in Figure 30 is a vibrating beam under tensile stress (string) that is clamped to a Si frame at both ends. By applying Newtons third law to a Euler-Bernoulli beam [45] whilst including a term for the tensile stress, the governing equation of a stressed string for an undamped bending vibration at small amplitudes is obtained:

$$\rho A \frac{\partial^2 u(x,t)}{\partial t^2} + EI_Z \frac{\partial^4 u(x,t)}{\partial x^4} - N \frac{\partial^2 u(x,t)}{\partial x^2} = 0$$

Equation 6

Where ρ is the density, A is the cross-sectional area, E is young's modulus, I_Z is the geometric moment of inertia and N is the tensile force of the string. Following separation of variables, the mode shape is $u(x,t)$ for the fundamental mode of the resonator can be expressed as:

$$u(x,t) = u(x)\, cos(\omega t)$$

Equation 7

u(x) being the mode shape as a function of *x*, *t* is the time and ω is the eigenmode frequency. By substituting Equation 7 into Equation 6 we can turn the partial differential equation into an ordinary differential equation that doesn't depend on time:

$$\omega^2 u(x) = \frac{Eh^2}{12\rho}\frac{\partial^4 u(x)}{\partial x^4} - \frac{\sigma}{\rho}\frac{\partial^2 u(x)}{\partial x^2}$$

Equation 8

where *h* is the thickness of the string and σ is its tensile stress. To discretize the equation, we apply the central finite difference method, which is a numerical technique that allows us to approximate the second and fourth derivative of the mode shape at a given point using a fixed number of points before and after. In this case we use four points before and after resulting in a 6th order accurate finite difference scheme for the fourth derivative and an eighth order accurate scheme for the 2nd derivative. Applying this method to Equation 8 we get:

$$\begin{aligned}\omega^2 u_i = \frac{\mathrm{Eh}^2}{12\rho\Delta x^4}\Big(&\frac{7}{240}u_{i-4} - \frac{2}{5}u_{i-3} + \frac{169}{60}u_{i-2} - \frac{122}{15}u_{i-1} + \frac{91}{8}u_i - \frac{122}{15}u_{i+1} + \frac{169}{60}u_{i+2} \\ &- \frac{2}{5}u_{i+3} + \frac{7}{240}u_{i+4}\Big) - \frac{\sigma}{\rho\Delta x^2}\Big(-\frac{1}{560}u_{i-4} + \frac{8}{315}u_{i-3} - \frac{1}{5}u_{i-2} + \frac{8}{5}u_{i-1} \\ &- \frac{205}{72}u_i + \frac{8}{5}u_{i+1} - \frac{1}{5}u_{i+2} + \frac{8}{315}u_{i+3} - \frac{1}{560}u_{i+4}\Big)\end{aligned}$$

Equation 9

Here u_i is the current point or node and Δx is the distance between the nodes. To transform Equation 9 into a matrix vector product, we consolidate like terms and expand it across a sequence of nodes on the string. Each row of the resulting matrix corresponds to a distinct node on the string:

$$\lambda I\vec{u} = G\vec{u}$$

Equation 10

With $\vec{u}$ being a vector from u_0 to u_n (the first and last nodes on the string), I is the identity matrix, $\lambda = \omega^2$ and G is a matrix containing all the collected terms for each node. A single coefficient of the matrix G can be represented by:

The diagonal terms ($G_{i,i}$, also identified as C_0) are:

$$G_{i,i} \equiv C_0 = \frac{Eh^2}{12\rho\Delta x^4}\left(\frac{91}{8}\right) - \frac{\sigma}{\rho\Delta x^2}\left(-\frac{205}{72}\right)$$

Equation 11

In turn, the off-diagonal (by one node) terms are:

$$G_{i,i\pm1} \equiv C_1 = \frac{Eh^2}{12\rho\Delta x^4}\left(\frac{122}{15}\right) - \frac{\sigma}{\rho\Delta x^2}\left(\frac{8}{5}\right)$$

Equation 12

The off diagonal by two terms are:

$$G_{i,i\pm2} \equiv C_2 = \frac{Eh^2}{12\rho\Delta x^4}\left(\frac{169}{60}\right) - \frac{\sigma}{\rho\Delta x^2}\left(-\frac{1}{5}\right)$$

Equation 13

The off diagonal by three terms are:

$$G_{i,i\pm3} \equiv C_3 = \frac{Eh^2}{12\rho\Delta x^4}\left(-\frac{2}{5}\right) - \frac{\sigma}{\rho\Delta x^2}\left(\frac{8}{315}\right)$$

Equation 14

The off diagonal by four terms are:

$$G_{i,i\pm4} \equiv C_4 = \frac{Eh^2}{12\rho\Delta x^4}\left(\frac{7}{240}\right) - \frac{\sigma}{\rho\Delta x^2}\left(-\frac{1}{560}\right)$$

Equation 15

After building the matrix, a key challenge is to enforce boundary conditions at the clamping points, and at the position where the mass is attached. At the ends, we enforce the boundary conditions of a double clamped resonator. That is at the clamping points of the frame and the string located at nodes u_0 and u_n. The boundary conditions are as follows:

$$u_0 = u_n = \frac{\partial u_0}{\partial \mathrm{x}} = \frac{\partial u_\mathrm{n}}{\partial \mathrm{x}} = 0$$

Equation 16

To ensure that our boundary conditions are met, we consider that nodes beyond the domain of the stressed string are clamped to the frame, and thus have a displacement of zero (e.g., un+1=0, un+2=0, etc.). Similarly, we enforce that the first and last nodes have a displacement of zero by setting all coefficients in the first and last row of the matrix *G* to zero. Enforced boundary conditions are represented in the following matrix, which shows the first 6 rows and 11 columns of Matrix *G*:

$$G = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ C_1 & C_0 & C_1 & C_2 & C_3 & C_4 & 0 & 0 & 0 & 0 & 0 & \cdots \\ C_2 & C_1 & C_0 & C_1 & C_2 & C_3 & C_4 & 0 & 0 & 0 & 0 & \cdots \\ C_3 & C_2 & C_1 & C_0 & C_1 & C_2 & C_3 & C_4 & 0 & 0 & 0 & \cdots \\ C_4 & C_3 & C_2 & C_1 & C_0 & C_1 & C_2 & C_3 & C_4 & 0 & 0 & \cdots \\ 0 & C_4 & C_3 & C_2 & C_1 & C_0 & C_1 & C_2 & C_3 & C_4 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

Equation 17

Now that we have established our model, we can move forward with verifying its accuracy by solving for the mode shape of a problem for which an analytical solution already exists: a stressed, non-mass-loaded resonator [45]. This is done prior to exploring the effects of mass loading. To achieve this, we solve for the smallest eigenvalue and corresponding eigenvector of the matrix *G* with boundary conditions of a doubly-clamped beam applied (Equation 16). We employed the eigenvalue solver available in SciPy to solve Equation 16, it

uses the mathematical technique of eigenvalue decomposition. The analytical solution to this problem is:

$$\phi(x) = sin\left(\frac{n\pi}{L}x\right) - \frac{n\pi}{L}\sqrt{\frac{EI_y}{\sigma A}}\left[cos\left(\frac{n\pi}{L}x\right) - e^{-\sqrt{\frac{\sigma A}{EI_y}}}\right]$$

Equation 18

where $\phi(x)$ is the normalized mode shape from 0 to $L/2$. Figure 31 (a) shows the numerical and analytical solutions [45] of the mode shape for a SiN resonator that is 100 nm thick and 8 mm long with an internal stress of 0.1 GPa using 10000 nodes (10000×10000 matrix). The mode shape and corresponding natural frequency of the analytical and numerical solutions are in good alignment. The accuracy of the method is measured by the relative error, which is the difference between the analytical and numerical solutions divided by the analytical solution. The relative error of the finite difference method generally decreases as the number of nodes used to approximate the derivative increases [46]. In this case, the scheme used has the lowest order of accuracy of 6, generally indicating that the relative error(Figure 31 d) should decreases by a factor of $2^6$for every doubling of the node count.

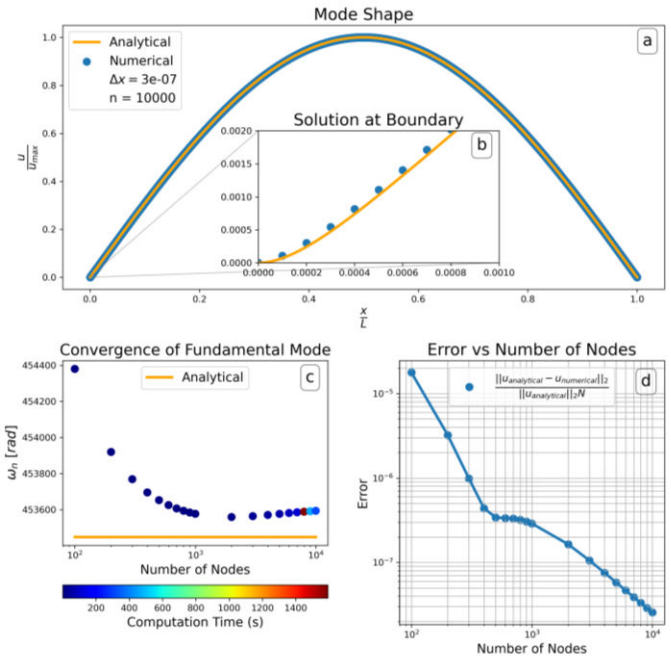

Figure 31: (a) non-mass-loaded SiN stressed string fundamental mode shape solved using 30000 nodes. (b) Boundary were the nano-string meets the frame. (c) Convergence of fundamental mode. (d) Relative error plot.

Figure 31 (b) demonstrates the resonator's bending near the clamping points with the frame, which is critical for predicting damping. Even though it occurs only within the first 0.06% of the beam length this sharp bending is typically the dominant dissipation path in these resonators [47]. It is a key result of our simulation that our bending matches the model. As a next step, in the mass loaded case, we will monitor whether the addition of a mass clamped to the center of the resonator excessively increases the sharp bending points.

To mass load our current model, we place the mass at the center of the stressed string going from ¼ to ¾ of its total length, as in Figure 30. Using the following, we force the clamping points of the mass to have no slope (for the fundamental mode) and no curvature:

$$\frac{\partial u_{n/4}}{\partial x} = \frac{\partial u_{3n/4}}{\partial x} = 0$$

Equation 19

To ensure compliance with the specified boundary conditions, we incorporate ghost nodes both before and after the clamping points where the string meets the mass. Ghost nodes are additional computational nodes introduced outside the physical domain of the problem or nodes inside the physical domain, that have a fixed value to facilitate the implementation of boundary conditions. Our implementation results in three distinct domains within the following structure: the left and right domains comprised of the SiN tethers, and the central domain occupied by the mass. The ghost nodes surrounding the mass ensure that it has a slope of 0 along its length, The corresponding matrix G for a small size of 36 × 36 nodes is shown in Figure 32. Figure 32 (a) shows the coefficients of the matrix G with the standard coefficients used in the non-mass loaded model in blue, the ghost nodes that are fixed to 0 are represented by an x and the boundary nodes that are substituted into the matrix are shown in red. Figure 32 (b) shows the same matrix but represented with a heat map.

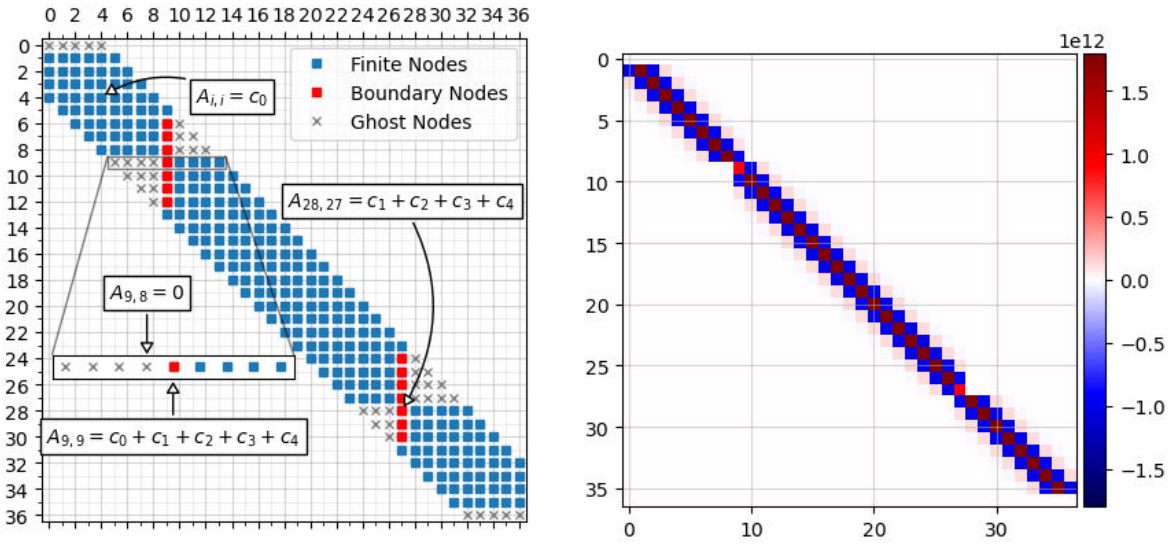

Figure 32: (a) Representation of matrix G for 36 nodes. (b) Heat-map of matrix G for 26 nodes.

It is important to note that this approach assumes that the mode shape is independent of the mass density and depends only on its length. As such, we are exclusively solving for the

stress string's shape with an attached mass and will need to integrate the mass density into subsequent calculations to obtain the eigenfrequency and quality factor. The mode shape of a 6 mm long and 100 nm thick SiN mass-loaded stressed string is presented in Figure 33 and has been calculated using 30000 nodes. The string is subject to 0.1 GPa of tensile stress and includes a mass with a length of 3 mm positioned at its center.

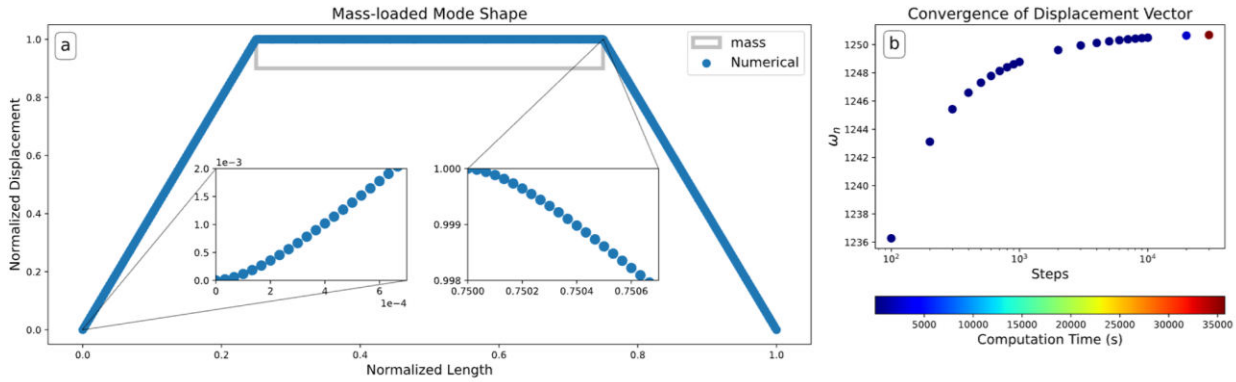

Figure 33: (a) Mode shape of Mass loaded SiN nano string. (b) Convergence of the natural frequency.

As can be seen in the insets of Figure 33 (a), the mode shape of the mass-loaded resonator exhibits a similar bending radius at both the clamping points between the string and the frame and the clamping points between the string and the mass. At these clamping points, the bending radius of the mass-loaded stress string is comparable to that of the non-mass-loaded stress string from Figure 31. The natural frequency (ω_n) converges towards around 1250.7 radians as visible in Figure 33 (b).

5.2 Analytical Mass-Loaded Model

An analytical approximation of a mass-loaded SiN resonator was created to compare with the results of our finite difference model to ensure its accuracy. This comparison allows us to assess the reliability of the finite difference model and somewhat validate its results. Figure 34

contains a schematic of the analytical model. The goal of the analytical model is to approximate the natural frequency to compare it with the numerically obtained natural frequency.

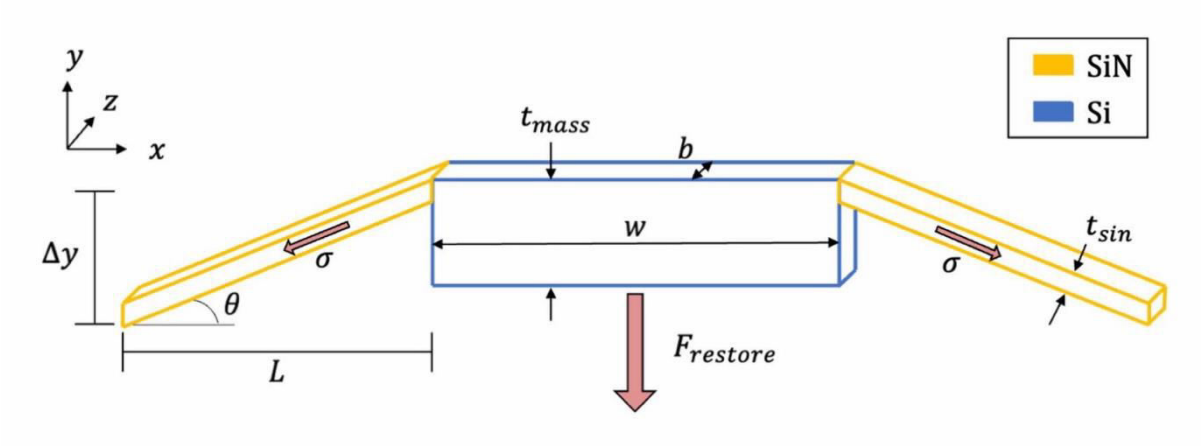

Figure 34: Analytical approximation of mass-loaded nano-string

In accordance with Figure 34, σ represents the internal tensile stress of the SiN, t_{mass} corresponds to the thickness of the mass, t_{SiN} is the thickness of the SiN tethers, L is the length of the tethers when the system is at rest, Δy denotes the amplitude of displacement, and b is the width of the mass and the tethers. The restoring force is given geometrically by:

$$F_{restore} = 2\sigma b t_{sin} \sin\theta$$

Equation 20

Since the amplitude of displacement (Δy) is orders of magnitude smaller than the length of the tethers (L), sinθ can be approximated as sinθ ≈Δy/L. As a result, the elongation of the tethers is considered negligible, and the tethers are assumed to be equal to their non-extended length. We can now calculate the spring constant:

$$k \approx \frac{\mathrm{F_{restore}}}{\Delta y}$$

Equation 21

The proof-mass of the analytical model (m) is given as follows:

$$m = \rho w b t_{mass}$$

Equation 22

Next, the natural frequency of the analytical model is expressed as:

$$\omega_{analytical} \approx \sqrt{\frac{k}{m}}$$

Equation 23

By substituting Equations 21-23 together, we can derive an equation for the analytical model's natural frequency:

$$\omega_{analytical} \approx \sqrt{\frac{2\sigma t_{\sin}}{\rho w L t_{mass}}}$$

Equation 24

The values of the parameters used in Equation 24 are provided in Table 4.

Table 4: Parameter values for equation 22, analytical approximation of natural frequency of a mass-loaded SiN nano string

Parameter	**Value**
$\boldsymbol{\sigma}$	0.1 Gpa
$\mathbf{t_{sin}}$	100 nm
$\boldsymbol{t_{mass}}$	500 μm
$\boldsymbol{L}$	3 mm
$\boldsymbol{w}$	3 mm
$\boldsymbol{\rho}$	3200 $^{kg}/_{m^3}$
$\boldsymbol{\omega_{analytical}}$	$\mathbf{1178\ rad}$

The natural frequency of the analytical approximation ($\omega_{analytical}$) was calculated to be 1178 rad, whereas the natural frequency obtained using the numerical finite difference method

(ω_n) was 1250 rad. Although there is a slight difference between the two values, with a percent error of approximately 5.7%, the fact that they are relatively close to each other still validates the numerical finite difference model.

5.3 Predicting Q-factor from the mode shape.

Having resolved the mass-loaded mode shape, we can now calculate the quality factor and eigenfrequency using the energy method [45]. This involves integrating over the mode shape to determine the energy in tension and bending. The behavior of a stressed string involves not only the bending energy resulting from its rigidity but also the tensile energy arising from its tensile stress. In other words, the string stores energy due to the work done against both bending and tensile stresses. These energies can be calculated using the following equations:

$$W_{Bending} = \frac{1}{2} E I_y \int_0^L (\frac{\partial^2 u}{\partial x^2})^2 dx$$

Equation 25

$$W_{Tension} = \frac{1}{2} \sigma A \int_0^L (\frac{\partial u}{\partial x})^2 dx$$

Equation 26

By utilizing these energies, we can compute the damping dilution factor. The damping dilution factor is defined as the ratio of energies in tension and bending, which is expressed as:

$$\alpha_{add} \approx \frac{W_{Tension}}{W_{bending}}$$

Equation 27

When a SiN nano-string is subjected to stress, the material is under-tension, leading to an increase in energy in tension compared to energy in bending. This leads to damping dilution factors that are greater than unity. Damping dilution refers to the reduction of damping caused by

a high ratio of energy in tension to energy in bending. Due to the damping dilution effect, the Q-factors of stressed SiN resonators are higher than those of un-stressed ones. These Q-factors can be calculated using:

$$Q \approx \alpha_{add} \cdot Q_{intrinsic}$$

Equation 28

where the $Q_{intrinsic}$ is the quality factor of non-stressed SiN from [48]. The Q-factor, energy in tension and energy in bending for the non-mass loaded and mass-loaded models are shown in Table 5.

Table 5: Energy's and Q-factor for mass-loaded and non-mass-loaded SiN nano-string

Model	Mass-loaded	Nano-string
Energy in Tension [$^{J}/_{m}$]	7.5×10^{-1}	6.1×10^{-1}
Energy in Bending [$^{J}/_{m}$]	4.3×10^{-4}	1.43×10^{-4}
Q-Factor	4.7×10^{6}	11×10^{6}

In summary, both the mass and non-mass loaded Q-factors were found to be within the same order of magnitude. The mass loaded Q-factor was significantly higher than the experimental value of 18,000. This suggests that there is potential for enhancing the mounting technique, as the lower-than-expected experimental Q-factor is not attributable to the added mass. This is supported by recent work done by another group in the field, that showed that by experimentally mass-loading a SiN resonator the quality-factor saturates in the limit of a large mass load [22]. By improving the mounting, we can further increase the Q-factor, leading to a lower thermomechanical noise limit and improved performance.

6. Conclusion and Future Work

7.1 Conclusion

In conclusion, this thesis successfully achieved its objectives by fabricating and characterizing the fundamental thermomechanical noise limit of mass-loaded silicon nitride resonators for traceable acceleration sensing. To the best of the author's knowledge, this is the mass-loaded SiN device with the lowest fundamental thermomechanical noise limit ($S_{a,tmech} \approx 1 \times 10^{-1}\ {}^{\mu g}/_{\sqrt{Hz}}$). This accomplishment was made possible through the implementation of a three-point kinematic magnetic mounting method on a device with a proof-mass ($m \approx 1.1 \times 10^{-6}$ Kg) and a natural frequency ($\omega_n = 526$ Hz) resulting in a Q-factor ($Q \approx 17{,}950$). While this mounting method resulted in a higher Q-factor than other methods tested in this thesis, numerical analysis suggests that the Q-factor could be further improved by exploring better mounting methods. Based on the simulation results presented in the previous section, it is evident that the Q-factor of our device is comparable to that of a SiN membrane of similar dimensions. However, the experimental Q-factor of our device falls two orders of magnitude below that of the membrane. This significant difference strongly suggests that further improvements in mounting techniques are required to minimize dissipation and enhance the Q-factor of our accelerometers. By doing so, we can effectively reduce the noise floor and enhance the overall performance of our device.This study represents a significant step forward in the development of traceable acceleration sensing using mass-loaded SiN resonators. Nonetheless, it is subject to certain limitations, which can be categorized as challenges in the detection system and the device itself.

Regarding the detection system, the fiber-interferometer has a higher noise floor than the thermomechanical limit of the device. To attain the fundamental limit of the device, a lower noise floor interferometer must be implemented. Furthermore, the interferometer has a linear region of only about 340 nm. By monitoring the signal from the accelerometer during oscillation, it can be observed that the accelerometer has a greater amplitude than the linear region of the interferometer. To measure the full amplitude of the accelerometer, a higher dynamic range interferometer is necessary. Lastly, the interferometer lacks directionality, which is necessary for identifying the direction of the acceleration input and further characterizing the device.

As for the limitations of the device, they are primarily related to the fabrication process and mounting. The success rate of the fabrication process can be significantly improved by incorporating more of the successful 6×6 mm accelerometers on the photomask, increasing the width of the border between the chips, and employing a new PTFE holder for KOH. Designing accelerometers with narrower tethers near the corners would result in sufficient under-etching for all chips. Finally, a better mounting method would result in a higher Q-factor that could potentially reduce the thermomechanical noise limit of the device to below that of the state-of-the art capacitive devices.

The implementation of these modifications is not only highly feasible but also strongly supported by my belief that implementing them would make our device suitable for acceleration sensing applications in metrology, with the goal of measuring the smallest acceleration possible with the utmost precision. While the goal of achieving the fundamental limit of acceleration sensing was not reached by this thesis, I firmly believe that it represents a significant step towards that direction for mass-loaded SiN resonators and demonstrates that they have the potential to reach an even smaller fundamental thermomechanical noise limit then other

comparable devices in the literature[1]–[3]. Exploring these devices at the frontier of science has the potential to generate new questions and discoveries.

7.2 Future Work

As previously mentioned, our readout system can be improved by implementing a new interferometer with a larger dynamic range, improved directionality, and reduced noise floor. To achieve this, we recommend using a Michelson interferometer with quadrature detection, similar to [49]. This interferometer utilizes laser modulation and demodulation (see Figure 37) to obtain multiple readout signals that can be combined in quadrature, effectively eliminating non-linear areas in the detection scheme, increasing the dynamic range, and enabling directional measurement.

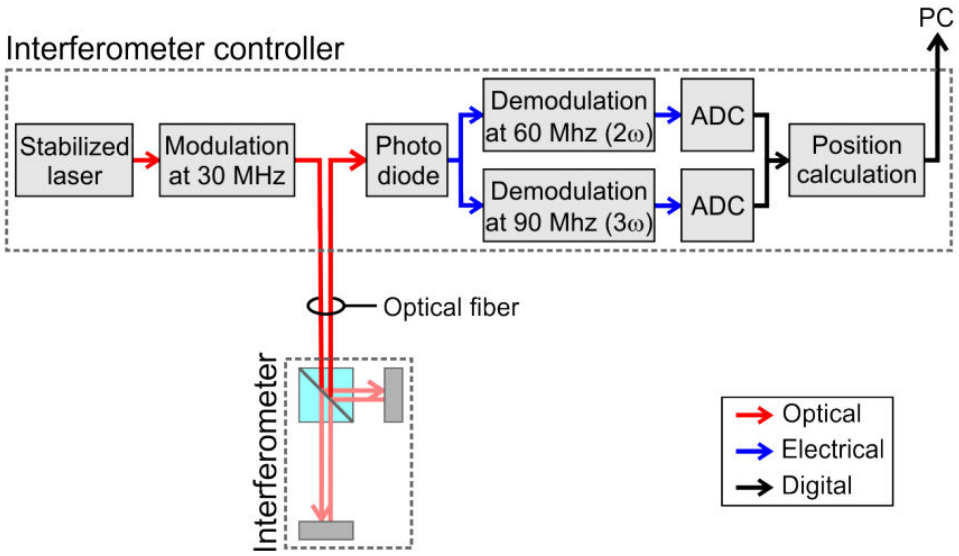

Figure 37: Modulated Michelson Interferometer. Taken from [50]

With our partners at the NRC, our next focus is on conducting shaker table tests to characterize the sensitivity of our accelerometer. We aim to determine the exact threshold of acceleration that our device can detect with the highest degree of accuracy. This test involves

subjecting the accelerometer to a known, small acceleration via a shaker and measuring the output signal generated by the device. By comparing the output signal with the known acceleration input, the sensitivity of the accelerometer can be determined. By conducting these tests, we can establish the performance parameters of our accelerometer, such as its dynamic range and resolution, which are critical to accurately measure and analyze the smallest accelerations possible.

To perform these tests at the NRC our team has developed a portable chamber that is held in vacuum by a molecular battery powered pump (see Figure 38). This portable chamber is small enough to fit on the shaker table at the NRC and can be used to characterize the sensitivity of our accelerometer.

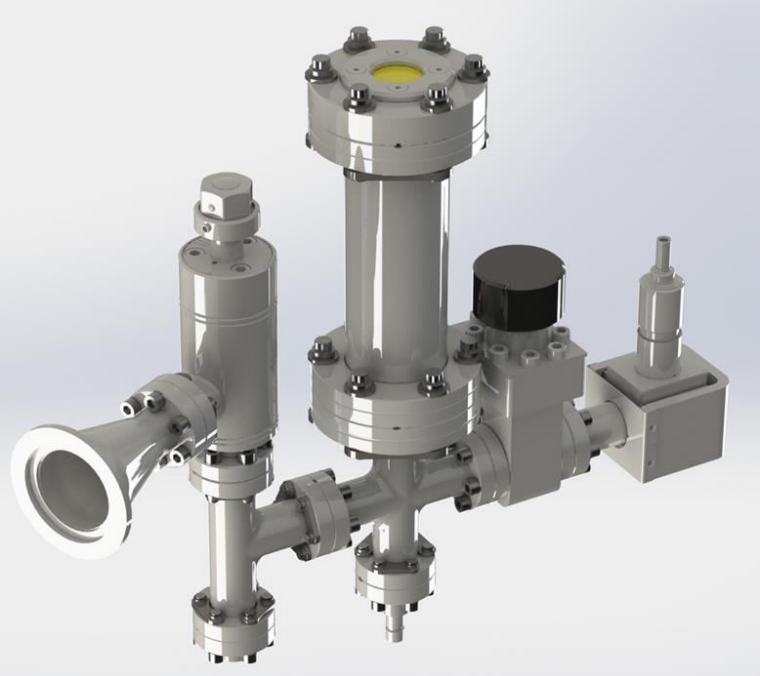

Figure 38: Portable Vacuum Chamber

As previously mentioned, there are various improvements that could be implemented in the fabrication process. One option is to design a new photomask that features $6 \times 6\, mm$

accelerometers with a 1mm border between chips, and tethers that connect to the frame with a width no wider than 700 μm. This modification could lead to more successful device fabrication. Another potential enhancement is to perform lithography on both sides of the wafer and remove the back membrane, replacing it with another trampoline structure. This alteration would decrease the overall stiffness and subsequently lower the natural frequency, resulting in a lower thermomechanical noise limit.

7. Bibliography

[1] A. G. Krause, M. Winger, T. D. Blasius, Q. Lin, and O. Painter, "A high-resolution microchip optomechanical accelerometer," *Nat. Photonics*, vol. 6, no. 11, Art. no. 11, Nov. 2012, doi: 10.1038/nphoton.2012.245.

[2] X. Fan *et al.*, "Manufacture and characterization of graphene membranes with suspended silicon proof masses for MEMS and NEMS applications," *Microsyst. Nanoeng.*, vol. 6, no. 1, Art. no. 1, Dec. 2020, doi: 10.1038/s41378-019-0128-4.

[3] F. Zhou, Y. Bao, R. Madugani, D. A. Long, J. J. Gorman, and T. W. LeBrun, "Broadband thermomechanically limited sensing with an optomechanical accelerometer," *Optica*, vol. 8, no. 3, p. 350, Mar. 2021, doi: 10.1364/OPTICA.413117.

[4] Y. Bidel *et al.*, "Absolute airborne gravimetry with a cold atom sensor," *J. Geod.*, vol. 94, no. 2, p. 20, 2020, doi: 10.1007/s00190-020-01350-2.

[5] C.-S. Jao, D. Wang, A. R. Parrish, and A. M. Shkel, "A Neural Network Approach to Mitigate Thermal-Induced Errors in ZUPT-aided INS," in *2022 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*, May 2022, pp. 1–4. doi: 10.1109/INERTIAL53425.2022.9787518.

[6] "The Comprehensive Nuclear-Test-Ban Treaty (CTBT) | CTBTO." https://www.ctbto.org/our-mission/the-treaty (accessed Oct. 26, 2022).

[7] N. R. C. Government of Canada, "IMS Seismic Network." https://can-ndc.nrcan.gc.ca/s_seismic-en.php (accessed Nov. 30, 2022).

[8] M. Cadoret, "PRECISION INERTIAL MEASUREMENTS USING COLD-ATOM INTERFEROMETRY," presented at the Inertial 2022, Avignon, France, May 08, 2022.

[9] O. J. Woodman, "An introduction to inertial navigation," p. 37.

[10] P. D. McFadden and J. D. Smith, "Vibration monitoring of rolling element bearings by the high-frequency resonance technique — a review," *Tribol. Int.*, vol. 17, no. 1, Art. no. 1, Feb. 1984, doi: 10.1016/0301-679X(84)90076-8.

[11] "ADXL354 Datasheet and Product Info | Analog Devices," Aug. 14, 2020. https://www.analog.com/en/products/adxl354.html (accessed Aug. 14, 2020).

[12] "Titan Accelerometer | Nanometrics," Aug. 14, 2020. https://www.nanometrics.ca/products/accelerometers/titan-accelerometer (accessed Aug. 14, 2020).

[13] C. Reinhardt, T. Müller, A. Bourassa, and J. C. Sankey, "Ultralow-Noise SiN Trampoline Resonators for Sensing and Optomechanics," *Phys. Rev. X*, vol. 6, no. 2, p. 021001, Apr. 2016, doi: 10.1103/PhysRevX.6.021001.

[14] J. Laine and D. Mougenot, "A high-sensitivity MEMS-based accelerometer," *Lead. Edge*, vol. 33, no. 11, Art. no. 11, Nov. 2014, doi: 10.1190/tle33111234.1.

[15] B. Malayappan, U. P. Lakshmi, B. V. V. S. N. P. Rao, K. Ramaswamy, and P. K. Pattnaik, "Sensing Techniques and Interrogation Methods in Optical MEMS Accelerometers: A Review," *IEEE Sens. J.*, vol. 22, no. 7, pp. 6232–6246, Apr. 2022, doi: 10.1109/JSEN.2022.3149662.

[16] "A low noise capacitive MEMS accelerometer with anti-spring structure | Elsevier Enhanced Reader." https://reader.elsevier.com/reader/sd/pii/S092442471832130 7?token=8CBBA8423D3FCABBC11EE0AA1D54F241D483DC87F19F8371A0E900272C21FFBCAF1941BF451D13DBA

BBC7F920402381A&originRegion=us-east-1&originCreation=20230105173017 (accessed Jan. 05, 2023).
[17] Z. Qu, H. Liu, H. Ouyang, C. Hu, and L. Tu, "A High-sensitivity Optical MEMS Accelerometer based on SOI Double-side Micromachining," in *2020 IEEE SENSORS*, Oct. 2020, pp. 1–4. doi: 10.1109/SENSORS47125.2020.9278761.
[18] M. Pandit *et al.*, "An Ultra-High Resolution Resonant MEMS Accelerometer," in *2019 IEEE 32nd International Conference on Micro Electro Mechanical Systems (MEMS)*, Jan. 2019, pp. 664–667. doi: 10.1109/MEMSYS.2019.8870734.
[19] X. Zou, P. Thiruvenkatanathan, and A. A. Seshia, "A Seismic-Grade Resonant MEMS Accelerometer," *J. Microelectromechanical Syst.*, vol. 23, no. 4, pp. 768–770, Aug. 2014, doi: 10.1109/JMEMS.2014.2319196.
[20] B. M. Zwickl *et al.*, "High quality mechanical and optical properties of commercial silicon nitride membranes," *Appl. Phys. Lett.*, vol. 92, no. 10, p. 103125, Mar. 2008, doi: 10.1063/1.2884191.
[21] M. W. Puckett *et al.*, "422 Million intrinsic quality factor planar integrated all-waveguide resonator with sub-MHz linewidth," *Nat. Commun.*, vol. 12, no. 1, Art. no. 1, Feb. 2021, doi: 10.1038/s41467-021-21205-4.
[22] R. Shaniv, S. K. Keshava, C. Reetz, and C. A. Regal, "Understanding the Quality Factor of Mass-Loaded Tensioned Resonators." arXiv, Sep. 03, 2022. Accessed: Jan. 18, 2023. [Online]. Available: http://arxiv.org/abs/2209.01488
[23] C. Zhang, M. Giroux, T. A. Nour, and R. St-Gelais, "Radiative Heat Transfer in Freestanding Silicon Nitride Membranes," *Phys. Rev. Appl.*, vol. 14, no. 2, p. 024072, Aug. 2020, doi: 10.1103/PhysRevApplied.14.024072.
[24] N. Sharma, M. Hooda, and S. K. Sharma, "Synthesis and Characterization of LPCVD Polysilicon and Silicon Nitride Thin Films for MEMS Applications," *J. Mater.*, vol. 2014, p. e954618, Apr. 2014, doi: 10.1155/2014/954618.
[25] C. Iliescu, F. E. H. Tay, and J. Wei, "Low stress PECVD—SiNx layers at high deposition rates using high power and high frequency for MEMS applications," *J. Micromechanics Microengineering*, vol. 16, no. 4, p. 869, Mar. 2006, doi: 10.1088/0960-1317/16/4/025.
[26] A. L. Koh, D. W. McComb, S. A. Maier, H. Y. Low, and J. K. W. Yang, "Sub-10 nm patterning of gold nanostructures on silicon-nitride membranes for plasmon mapping with electron energy-loss spectroscopy," *J. Vac. Sci. Technol. B*, vol. 28, no. 6, p. C6O45-C6O49, Nov. 2010, doi: 10.1116/1.3501351.
[27] H. D. Tong *et al.*, "Silicon Nitride Nanosieve Membrane," *Nano Lett.*, vol. 4, no. 2, pp. 283–287, Feb. 2004, doi: 10.1021/nl0350175.
[28] J. S. Kim *et al.*, "Fabrication of high-speed polyimide-based humidity sensor using anisotropic and isotropic etching with ICP," *Thin Solid Films*, vol. 517, no. 14, pp. 3879–3882, May 2009, doi: 10.1016/j.tsf.2009.01.129.
[29] W. Jiang, D. Xu, S. Yao, B. Xiong, and Y. Wang, "Effect of hyperthermal annealing on LPCVD silicon nitride," *Mater. Sci. Semicond. Process.*, vol. 43, pp. 222–229, Mar. 2016, doi: 10.1016/j.mssp.2015.12.020.
[30] T. Arakawa *et al.*, "Radiation stability of SiO2-antireflective film coated SiN and SiC x-ray mask membranes," *J. Vac. Sci. Technol. B Microelectron. Nanometer Struct. Process. Meas. Phenom.*, vol. 12, no. 4, pp. 2372–2375, Jul. 1994, doi: 10.1116/1.587766.

[31] S. Tuomikoski and S. Franssila, “Wafer-Level Bonding of MEMS Structures with SU-8 Epoxy Photoresist,” *Phys. Scr.*, vol. 2004, no. T114, p. 223, Jan. 2004, doi: 10.1088/0031-8949/2004/T114/056.
[32] S. Tanaka, “Wafer-level hermetic MEMS packaging by anodic bonding and its reliability issues,” *Microelectron. Reliab.*, vol. 54, no. 5, pp. 875–881, May 2014, doi: 10.1016/j.microrel.2014.02.001.
[33] J. Jellison, “Effect of Surface Contamination on the Thermocompression Bondability of Gold,” *IEEE Trans. Parts Hybrids Packag.*, vol. 11, no. 3, pp. 206–211, Sep. 1975, doi: 10.1109/TPHP.1975.1135065.
[34] A. Singh, D. A. Horsley, M. B. Cohn, A. P. Pisano, and R. T. Howe, “Batch transfer of microstructures using flip-chip solder bonding,” *J. Microelectromechanical Syst.*, vol. 8, no. 1, pp. 27–33, Mar. 1999, doi: 10.1109/84.749399.
[35] R. C. Benson, D. Farrar, and J. A. Miragliotta, “Polymer Adhesives and Encapsulants for Microelectronic Applications,” *Johns Hopkins APL Tech. Dig.*, vol. 28, no. 1, 2008.
[36] R. S. Pai and K. M. Walsh, “The viability of anisotropic conductive film as a flip chip interconnect technology for MEMS devices,” *J. Micromechanics Microengineering*, vol. 15, no. 6, p. 1131, Apr. 2005, doi: 10.1088/0960-1317/15/6/003.
[37] G. Mu, “SiN Drum Resonator Fabrication and Integrated Actuation Using Substrate Capacitors.”
[38] I. Bajwa, “KOH etching of (100) Si wafer, No 1,” *Protoc. Rep.*, Mar. 2016, [Online]. Available: https://repository.upenn.edu/scn_protocols/18
[39] “datasheets_S1800.pdf.” Accessed: Jan. 18, 2023. [Online]. Available: https://amolf.nl/wp-content/uploads/2016/09/datasheets_S1800.pdf
[40] Mohammad. J. Bereyhi *et al.*, “Clamp-Tapering Increases the Quality Factor of Stressed Nanobeams,” *Nano Lett.*, vol. 19, no. 4, pp. 2329–2333, Apr. 2019, doi: 10.1021/acs.nanolett.8b04942.
[41] D. Rugar, H. J. Mamin, and P. Guethner, “Improved fiber-optic interferometer for atomic force microscopy,” *Appl. Phys. Lett.*, vol. 55, no. 25, pp. 2588–2590, Dec. 1989, doi: 10.1063/1.101987.
[42] S. Schmid, K. D. Jensen, K. H. Nielsen, and A. Boisen, “Damping mechanisms in high-Q micro and nanomechanical string resonators,” *Phys. Rev. B*, vol. 84, no. 16, Art. no. 16, Oct. 2011, doi: 10.1103/PhysRevB.84.165307.
[43] C. Zhang, M. Giroux, T. A. Nour, and R. St-Gelais, “Radiative Heat Transfer in Free-Standing Silicon Nitride Membranes,” *ArXiv200209017 Phys.*, Feb. 2020, Accessed: Apr. 07, 2020. [Online]. Available: http://arxiv.org/abs/2002.09017
[44] L. Sementilli, E. Romero, and W. P. Bowen, “Nanomechanical Dissipation and Strain Engineering,” *Adv. Funct. Mater.*, vol. 32, no. 3, p. 2105247, 2022, doi: 10.1002/adfm.202105247.
[45] S. Schmid, L. G. Villanueva, and M. L. Roukes, *Fundamentals of Nanomechanical Resonators*. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-28691-4.
[46] A. Iserles, *A first course in the numerical analysis of differential equations*. Cambridge ; New York : Cambridge University Press, 2009. Accessed: Mar. 15, 2023. [Online]. Available: http://archive.org/details/firstcoursenumer00iser

[47] Q. P. Unterreithmeier, T. Faust, and J. P. Kotthaus, “Damping of Nanomechanical Resonators,” *Phys. Rev. Lett.*, vol. 105, no. 2, p. 027205, Jul. 2010, doi: 10.1103/PhysRevLett.105.027205.
[48] R. Nikbakht, X. Xie, A. Weck, and R. St-Gelais, “High Q-Factor Silicon Nitride Nanomechanical Resonators Fabricated by Maskless Femtosecond Laser Micro-machining.” arXiv, Oct. 01, 2022. doi: 10.48550/arXiv.2210.00290.
[49] A. Dandridge, A. B. Tveten, and T. G. Giallorenzi, “Homodyne Demodulation Scheme for Fiber Optic Sensors Using Phase Generated Carrier,” *IEEE Trans. Microw. Theory Tech.*, vol. 30, no. 10, pp. 1635–1641, Oct. 1982, doi: 10.1109/TMTT.1982.1131302.
[50] “Application Notes & Documentation of the Vibrometer - SmarAct.” https://www.smaract.com/en/metrology-applications (accessed Mar. 15, 2023).

8. Appendix

9.1 Finite difference Simulation Code

```
# Import Library's
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import math
import scipy.linalg as la
import pandas as pd

# Define Parameters
params = dict(start=500, end=10000, length=0.008, thickness=100*10**-9, young=225*10**9, sigma=0.10*10**9, row=3200)


# Launch main
def main(param):
    # Create output
    output = dict(omega=np.empty(0), steps=np.empty(0), displacement=np.empty(0))
    # Initialize n
    n = param["start"]
    # Loop through sol while doubling n
    while n <= param["end"]:
        # build FD matrix
        A = build(n, h=param["length"] / n, D=(param["young"] * param["thickness"] ** 2) / (12 * param["row"]),
                  T=param["sigma"] / (param["row"]))
        # Solve shape
        output = sol_shape(A, output, n)
        del A
        # Solve freq
        output = sol_freq(output, n, param, h=param["length"] / n)
        print("n = ", n, " omega = ", output["omega"][-1])
        n = n + 500
    n = int(n-500)
    # Graph results
    graph(output, n, param, h=param["length"] / n)

    return


def build(n, h, D, T):
    # Finite Central Difference matrix
    A = np.zeros(((n + 1), (n + 1)))

    # Build SiN Sol first quarter
    for i in range(4, int(n / 4)):
        A[i, i - 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0
        A[i, i - 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
        A[i, i - 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
        A[i, i - 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
        A[i, i] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
        A[i, i + 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
        A[i, i + 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
        A[i, i + 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
        A[i, i + 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    # Build SiN Sol last quarter
    for i in range(int(3 * n / 4), int(n - 3)):
        A[i, i - 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0
        A[i, i - 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
        A[i, i - 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
        A[i, i - 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
        A[i, i] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
        A[i, i + 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
        A[i, i + 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
        A[i, i + 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
        A[i, i + 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    # Build SiN boarder first quarter
    A[int(n / 4) - 1, :] = 0
    A[int(n / 4) - 1, int(n / 4)] = (D / (h ** 4)) * (-122 / 15.0 + 169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (
                T / (h ** 2)) * (8 / 5.0 - 1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(n / 4) - 1, int(n / 4) - 1] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(n / 4) - 1, int(n / 4) - 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) - 1, int(n / 4) - 3] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) - 1, int(n / 4) - 4] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(n / 4) - 1, int(n / 4) - 5] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(n / 4) - 2, :] = 0
    A[int(n / 4) - 2, int(n / 4) - 0] = (D / (h ** 4)) * (169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (T / (h ** 2)) * (
                -1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(n / 4) - 2, int(n / 4) - 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) - 2, int(n / 4) - 2] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(n / 4) - 2, int(n / 4) - 3] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) - 2, int(n / 4) - 4] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) - 2, int(n / 4) - 5] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(n / 4) - 2, int(n / 4) - 6] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0
```

```
    A[int(n / 4) - 3, :] = 0
    A[int(n / 4) - 3, int(n / 4) - 0] = (D / (h ** 4)) * (-2 / 5.0 + 7 / 240.0) - (T / (h ** 2)) * (
                8 / 315.0 - 1 / 560.0)
    A[int(n / 4) - 3, int(n / 4) - 1] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) - 3, int(n / 4) - 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) - 3, int(n / 4) - 3] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(n / 4) - 3, int(n / 4) - 4] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) - 3, int(n / 4) - 5] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) - 3, int(n / 4) - 6] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(n / 4) - 3, int(n / 4) - 7] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    ## Build Boarder left Si
    A[int(n / 4), :] = 0
    A[int(n / 4), int(n / 4)] = (D / (h ** 4)) * (91 / 8.0 - 122 / 15.0 + 169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (
                T / (h ** 2)) * (-205 / 72.0 + 8 / 5.0 - 1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(n / 4), int(n / 4) + 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4), int(n / 4) + 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4), int(n / 4) + 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(n / 4), int(n / 4) + 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(n / 4) + 1, :] = 0
    A[int(n / 4) + 1, int(n / 4)] = (D / (h ** 4)) * (-122 / 15.0 + 169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (
                T / (h ** 2)) * (8 / 5.0 - 1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(n / 4) + 1, int(n / 4) + 1] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(n / 4) + 1, int(n / 4) + 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) + 1, int(n / 4) + 3] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) + 1, int(n / 4) + 4] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(n / 4) + 1, int(n / 4) + 5] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(n / 4) + 2, :] = 0
    A[int(n / 4) + 2, int(n / 4) + 0] = (D / (h ** 4)) * (169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (T / (h ** 2)) * (
                -1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(n / 4) + 2, int(n / 4) + 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) + 2, int(n / 4) + 2] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(n / 4) + 2, int(n / 4) + 3] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) + 2, int(n / 4) + 4] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) + 2, int(n / 4) + 5] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(n / 4) + 2, int(n / 4) + 6] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(n / 4) + 3, :] = 0
    A[int(n / 4) + 3, int(n / 4) + 0] = (D / (h ** 4)) * (-2 / 5.0 + 7 / 240.0) - (T / (h ** 2)) * (
                8 / 315.0 - 1 / 560.0)
    A[int(n / 4) + 3, int(n / 4) + 1] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) + 3, int(n / 4) + 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) + 3, int(n / 4) + 3] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(n / 4) + 3, int(n / 4) + 4] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(n / 4) + 3, int(n / 4) + 5] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(n / 4) + 3, int(n / 4) + 6] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(n / 4) + 3, int(n / 4) + 7] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    ## Build Sol Si
    for i in range(int(n / 4 + 4), int(3 * n / 4 - 3)):
        A[i, i - 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0
        A[i, i - 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
        A[i, i - 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
        A[i, i - 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
        A[i, i] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
        A[i, i + 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
        A[i, i + 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
        A[i, i + 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
        A[i, i + 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    ## Build Boarder RIGHT Si
    A[int(3 * n / 4), :] = 0
    A[int(3 * n / 4), int(3 * n / 4)] = (D / (h ** 4)) * (
                91 / 8.0 - 122 / 15.0 + 169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (T / (h ** 2)) * (
                                                    -205 / 72.0 + 8 / 5.0 - 1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(3 * n / 4), int(3 * n / 4) - 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4), int(3 * n / 4) - 2] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4), int(3 * n / 4) - 3] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(3 * n / 4), int(3 * n / 4) - 4] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(3 * n / 4) - 1, :] = 0
    A[int(3 * n / 4) - 1, int(3 * n / 4)] = (D / (h ** 4)) * (-122 / 15.0 + 169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (
                T / (h ** 2)) * (8 / 5.0 - 1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(3 * n / 4) - 1, int(3 * n / 4) - 1] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(3 * n / 4) - 1, int(3 * n / 4) - 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) - 1, int(3 * n / 4) - 3] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) - 1, int(3 * n / 4) - 4] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(3 * n / 4) - 1, int(3 * n / 4) - 5] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(3 * n / 4) - 2, :] = 0
    A[int(3 * n / 4) - 2, int(3 * n / 4) - 0] = (D / (h ** 4)) * (169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (
                T / (h ** 2)) * (-1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(3 * n / 4) - 2, int(3 * n / 4) - 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) - 2, int(3 * n / 4) - 2] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(3 * n / 4) - 2, int(3 * n / 4) - 3] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) - 2, int(3 * n / 4) - 4] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) - 2, int(3 * n / 4) - 5] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(3 * n / 4) - 2, int(3 * n / 4) - 6] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(3 * n / 4) - 3, :] = 0
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 0] = (D / (h ** 4)) * (-2 / 5.0 + 7 / 240.0) - (T / (h ** 2)) * (
                8 / 315.0 - 1 / 560.0)
```

```
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 1] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 3] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 4] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 5] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 6] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(3 * n / 4) - 3, int(3 * n / 4) - 7] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    # Build SiN boarder last quarter
    A[int(3 * n / 4) + 1, :] = 0
    A[int(3 * n / 4) + 1, int(3 * n / 4)] = (D / (h ** 4)) * (-122 / 15.0 + 169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (
                T / (h ** 2)) * (8 / 5.0 - 1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(3 * n / 4) + 1, int(3 * n / 4) + 1] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(3 * n / 4) + 1, int(3 * n / 4) + 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) + 1, int(3 * n / 4) + 3] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) + 1, int(3 * n / 4) + 4] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(3 * n / 4) + 1, int(3 * n / 4) + 5] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(3 * n / 4) + 2, :] = 0
    A[int(3 * n / 4) + 2, int(3 * n / 4) + 0] = (D / (h ** 4)) * (169 / 60.0 - 2 / 5.0 + 7 / 240.0) - (
                T / (h ** 2)) * (-1 / 5.0 + 8 / 315.0 - 1 / 560.0)
    A[int(3 * n / 4) + 2, int(3 * n / 4) + 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) + 2, int(3 * n / 4) + 2] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(3 * n / 4) + 2, int(3 * n / 4) + 3] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) + 2, int(3 * n / 4) + 4] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) + 2, int(3 * n / 4) + 5] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(3 * n / 4) + 2, int(3 * n / 4) + 6] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[int(3 * n / 4) + 3, :] = 0
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 0] = (D / (h ** 4)) * (-2 / 5.0 + 7 / 240.0) - (T / (h ** 2)) * (
                8 / 315.0 - 1 / 560.0)
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 1] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 3] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 4] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 5] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 6] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[int(3 * n / 4) + 3, int(3 * n / 4) + 7] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    ## Build Left Boarder
    A[1, 0] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[1, 1] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[1, 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[1, 3] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[1, 4] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[1, 5] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[2, 0] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[2, 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[2, 2] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[2, 3] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[2, 4] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[2, 5] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[2, 6] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[3, 0] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[3, 1] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[3, 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[3, 3] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[3, 4] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[3, 5] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[3, 6] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[3, 7] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    ## Build right Boarder
    A[n - 1, n - 0] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[n - 1, n - 1] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[n - 1, n - 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[n - 1, n - 3] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[n - 1, n - 4] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[n - 1, n - 5] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[n - 2, n - 0] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[n - 2, n - 1] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[n - 2, n - 2] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[n - 2, n - 3] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[n - 2, n - 4] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[n - 2, n - 5] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[n - 2, n - 6] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    A[n - 3, n - 0] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[n - 3, n - 1] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[n - 3, n - 2] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[n - 3, n - 3] = (D / (h ** 4)) * 91 / 8.0 - (T / (h ** 2)) * -205 / 72.0
    A[n - 3, n - 4] = (D / (h ** 4)) * -122 / 15.0 - (T / (h ** 2)) * 8 / 5.0
    A[n - 3, n - 5] = (D / (h ** 4)) * 169 / 60.0 - (T / (h ** 2)) * -1 / 5.0
    A[n - 3, n - 6] = (D / (h ** 4)) * -2 / 5.0 - (T / (h ** 2)) * 8 / 315.0
    A[n - 3, n - 7] = (D / (h ** 4)) * 7 / 240.0 - (T / (h ** 2)) * -1 / 560.0

    return A


def sol_shape(A, out, n):

    A = np.matrix(A)
```

```
    li = la.eig(A)
    eigenvalues, eigenvectors = li[0], li[1]
    eigenvalues = eigenvalues.real
    eigenvalues = np.trim_zeros(eigenvalues)
    lamb = np.where(eigenvalues == eigenvalues.min())
    out["displacement"] = np.abs(np.reshape(eigenvectors[:, lamb], (n + 1)))
    out["steps"] = np.append(out["steps"], n)

    return out


def sol_freq(out, n, param, h):

    uprime = derive(out["displacement"], n, h)
    double_uprime = double_derive(out["displacement"], n, h)

    thickness = np.ones(n + 1) * param["thickness"]
    thickness[int(n / 4):int(3 * n / 4 + 1)] = 5000 * param["thickness"]
    # Use quadtradture
    # Scipy integrate
    Wten = (1 / 2.0) * param["sigma"] * param["thickness"] * np.trapz(uprime ** 2, x=None, dx=h, axis=- 1)
    Wbend = (1 / 24.0) * param["young"] * np.trapz((param["thickness"] ** 3) * double_uprime ** 2, x=None, dx=h, axis=- 1)
    Wkin = (1 / 2.0) * param["row"] * np.trapz((thickness * out["displacement"] ** 2), x=None, dx=h, axis=- 1)
    out["omega"] = np.append(out["omega"], math.sqrt((Wbend + Wten) / Wkin))

    data = [Wten, Wbend, Wkin]
    data = pd.DataFrame(data, index=["Tension Energy", "Bending Energy", "Kinetic Energy"])
    # display table
    print(data)

    return out


def derive(u, n, h):

    uprime = np.zeros(n + 1)

    uprime[0] = (4.0 / 5 * u[0 + 1] - 1 / 5.0 * u[0 + 2] + 4 / 105.0 * u[0 + 3] - 1 / 280.0 * u[0 + 4]) / h

    uprime[1] = (-4.0 / 5 * u[1 - 1] + 4.0 / 5 * u[1 + 1] - 1 / 5.0 * u[1 + 2] + 4 / 105.0 * u[1 + 3] - 1 / 280.0 * u[
        1 + 4]) / h
    uprime[2] = (1 / 5.0 * u[2 - 2] - 4.0 / 5 * u[2 - 1] + 4.0 / 5 * u[2 + 1] - 1 / 5.0 * u[2 + 2] + 4 / 105.0 * u[
        2 + 3] - 1 / 280.0 * u[2 + 4]) / h
    uprime[3] = (-4 / 105.0 * u[3 - 3] + 1 / 5.0 * u[3 - 2] - 4.0 / 5 * u[3 - 1] + 4.0 / 5 * u[3 + 1] - 1 / 5.0 * u[
        3 + 2] + 4 / 105.0 * u[3 + 3] - 1 / 280.0 * u[3 + 4]) / h

    uprime[n - 3] = (1 / 280.0 * u[n - 3 - 4] - 4 / 105.0 * u[n - 3 - 3] + 1 / 5.0 * u[n - 3 - 2] - 4.0 / 5 * u[
        n - 3 - 1] + 4.0 / 5 * u[n - 3 + 1] - 1 / 5.0 * u[n - 3 + 2] + 4 / 105.0 * u[n - 3 + 3]) / h
    uprime[n - 2] = (1 / 280.0 * u[n - 2 - 4] - 4 / 105.0 * u[n - 2 - 3] + 1 / 5.0 * u[n - 2 - 2] - 4.0 / 5 * u[
        n - 2 - 1] + 4.0 / 5 * u[n - 2 + 1] - 1 / 5.0 * u[n - 2 + 2]) / h
    uprime[n - 1] = (1 / 280.0 * u[n - 1 - 4] - 4 / 105.0 * u[n - 1 - 3] + 1 / 5.0 * u[n - 1 - 2] - 4.0 / 5 * u[
        n - 1 - 1] + 4.0 / 5 * u[n - 1 + 1]) / h
    uprime[n] = (1 / 280.0 * u[n - 4] - 4 / 105.0 * u[n - 3] + 1 / 5.0 * u[n - 2] - 4.0 / 5 * u[n - 1]) / h

    for i in range(4, n - 3):
        uprime[i] = (1 / 280.0 * u[i - 4] - 4 / 105.0 * u[i - 3] + 1 / 5.0 * u[i - 2] - 4.0 / 5 * u[i - 1] + 4.0 / 5 *
                     u[i + 1] - 1 / 5.0 * u[i + 2] + 4 / 105.0 * u[i + 3] - 1 / 280.0 * u[i + 4]) / h

    return uprime


def double_derive(u, n, h):

    double_uprime = np.zeros(n + 1)

    double_uprime[0] = (- 205 / 72.0 * u[0] + 8.0 / 5 * u[0 + 1] - 1 / 5.0 * u[0 + 2] + 8 / 315.0 * u[0 + 3] - 1 / 560.0 * u[
        0 + 4]) / h ** 2
    double_uprime[1] = (8.0 / 5 * u[1 - 1] - 205 / 72.0 * u[1] + 8.0 / 5 * u[1 + 1] - 1 / 5.0 * u[1 + 2] + 8 / 315.0 * u[
        1 + 3] - 1 / 560.0 * u[1 + 4]) / h ** 2
    double_uprime[2] = (- 1 / 5.0 * u[2 - 2] + 8.0 / 5 * u[2 - 1] - 205 / 72.0 * u[2] + 8.0 / 5 * u[2 + 1] - 1 / 5.0 * u[
        2 + 2] + 8 / 315.0 * u[2 + 3] - 1 / 560.0 * u[2 + 4]) / h ** 2
    double_uprime[3] = (+ 8 / 315.0 * u[3 - 3] - 1 / 5.0 * u[3 - 2] + 8.0 / 5 * u[3 - 1] - 205 / 72.0 * u[3] + 8.0 / 5 * u[
        3 + 1] - 1 / 5.0 * u[3 + 2] + 8 / 315.0 * u[3 + 3] - 1 / 560.0 * u[3 + 4]) / h ** 2

    double_uprime[n - 3] = (-1 / 560.0 * u[n - 3 - 4] + 8 / 315.0 * u[n - 3 - 3] - 1 / 5.0 * u[n - 3 - 2] + 8.0 / 5 * u[
        n - 3 - 1] - 205 / 72.0 * u[n - 3] + 8.0 / 5 * u[n - 3 + 1] - 1 / 5.0 * u[n - 3 + 2] + 8 / 315.0 * u[
                           n - 3 + 3]) / h ** 2
    double_uprime[n - 2] = (-1 / 560.0 * u[n - 2 - 4] + 8 / 315.0 * u[n - 2 - 3] - 1 / 5.0 * u[n - 2 - 2] + 8.0 / 5 * u[
        n - 2 - 1] - 205 / 72.0 * u[n - 2] + 8.0 / 5 * u[n - 2 + 1] - 1 / 5.0 * u[n - 2 + 2]) / h ** 2
    double_uprime[n - 1] = (-1 / 560.0 * u[n - 1 - 4] + 8 / 315.0 * u[n - 1 - 3] - 1 / 5.0 * u[n - 1 - 2] + 8.0 / 5 * u[
        n - 1 - 1] - 205 / 72.0 * u[n - 1] + 8.0 / 5 * u[n - 1 + 1]) / h ** 2
    double_uprime[n] = (-1 / 560.0 * u[n - 4] + 8 / 315.0 * u[n - 3] - 1 / 5.0 * u[n - 2] + 8.0 / 5 * u[n - 1] - 205 / 72.0 *
                u[n]) / h ** 2

    for i in range(4, n - 3):
        double_uprime[i] = (-1 / 560.0 * u[i - 4] + 8 / 315.0 * u[i - 3] - 1 / 5.0 * u[i - 2] + 8.0 / 5 * u[
            i - 1] - 205 / 72.0 * u[i] + 8.0 / 5 * u[i + 1] - 1 / 5.0 * u[i + 2] + 8 / 315.0 * u[i + 3] - 1 / 560.0 * u[
                           i + 4]) / h ** 2

    return double_uprime


def graph(out, n, param, h):

    r = (out["omega"]-out["omega"][-1])**2
```

```
    print(r)
    plt.scatter(out["steps"], r)
    plt.title("Convergence of Natural Frequency")
    plt.ylabel(r'${\omega} \ [{rad}^{2}]$')
    plt.xlabel("number of pts")
    plt.xscale("log")
    plt.yscale("log")
    plt.text(out["steps"][0], out["omega"][-1], r'${\omega}$' + " = " + str(out["omega"][-1]))
    plt.show()

    x = np.linspace(0, 1, n + 1)
    fig, ax1 = plt.subplots(figsize=(20, 5))
    plt.title("Mode Shape")
    ax2 = fig.add_axes([0.3, 0.2, 0.2, 0.2])
    ax3 = fig.add_axes([0.55, 0.2, 0.2, 0.2])
    ax1.plot(x, out["displacement"] / out["displacement"].max(), label="Numerical")
    ax2.set_title("Left Boundary")
    ax3.set_title("Mass Boundary")
    ax2.set_xlim(0, 0.001)
    ax2.set_ylim(0, 0.002)
    ax2.scatter(x, out["displacement"] / out["displacement"].max(), linestyle="dashed", label="Numerical")
    ax1.legend()
    ax3.scatter(x, out["displacement"] / out["displacement"].max(), label="Numerical")
    ax3.set_xlim(0.249, 0.25)
    ax3.set_ylim(0.998, 1)
    ax1.add_patch(Rectangle((1 / 4.0, 1.0), 2 / 4.0, -0.1, edgecolor='silver', facecolor='none'))
    plt.show()

    uprime = derive(out["displacement"], n, h)
    double_uprime = double_derive(out["displacement"], n, h)
    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(30, 6))
    ax1.scatter(x, uprime / np.max(uprime), label="Numerical")
    ax1.set_xlabel(r'$\frac{x}{L}$', fontsize=18)
    ax2.set_xlabel(r'$\frac{x}{L}$', fontsize=18)
    f.suptitle("Derivatives of mode shape")
    ax1.set_ylabel(r'$\frac{u}{{u}_{max}}$', fontsize=18)
    ax2.set_ylabel(r'$\frac{u}{{u}_{max}}$', fontsize=18)
    ax1.set_title(r'$\frac{\partial{u}}{\partial{x}}$', fontsize=20)
    ax2.set_title(r'$\frac{\partial^{2}{u}}{\partial{x}^{2}}$', fontsize=20)
    ax2.plot(x, double_uprime / np.max(double_uprime), linestyle="dashed", label="Numerical")
    plt.show()

    fig = plt.figure(figsize=(30, 10))
    ax = fig.gca()

    ax.add_patch(Rectangle((1 / 4.0, 1.0), 2 / 4.0, -0.1, edgecolor='silver', facecolor='none',linewidth=7))
    plt.plot(x, out["displacement"] / out["displacement"].max(), label="Numerical", linewidth=7)

    plt.show()
    np.savetxt("dis.csv", out["displacement"], delimiter=",")


if __name__ == '__main__':
    main(params)
```

9.2 Photomask Code

```
import gdspy
import math

# The GDSII file is called a library, which contains multiple cells.
lib = gdspy.GdsLibrary()
ld_photomask = {"layer": 0}
ld_wafer = {"layer": 2}
wafer = lib.new_cell("Wafer")


def roman num(i):
    i = i + 1

    if i == 1:
        sting = "I"
    elif i == 2:
        sting = "II"
    elif i == 3:
        sting = "III"
    elif i == 4:
        sting = "IV"
    elif i == 5:
        sting = "V"
    elif i == 6:
        sting = "VI"
    elif i == 7:
        sting = "VII"
    elif i == 8:
        sting = "IIX"
    elif i == 9:
        sting = "IX"
    elif i == 10:
        sting = "X"

    return (sting)


def makechipsA(s_min, s_max, theta_min, theta_max, r, T, R, mem, c, mass, margin):
    for i in range(r):

        # Inputs
        w = (s_max - s_min) / (r - 1)

        E = (theta_max - theta_min) / (r - 1)
        # Tether width
        s1 = s_min + w * i  # um

        k = s1 / math.sqrt(2)

        # Use complex numbers to facilitate writing polar coordinates
        c3 = gdspy.Curve(-mass + k, 0).C((-theta_min - i * E) + k, -2000, (theta_min + i * E) - k, -2000, mass - k, 0)
        # Elliptical arcs have syntax similar to gdspy.Round

        c0 = gdspy.Polygon(c3.get_points(), **ld_photomask)

        c0.fillet(700)

        a1 = mass
        # Create a cell with a component that is used repeatedly
        cutout = lib.new_cell("CUTOUTA" + str(i))

        cutout.add(c0)

        # Create a cell with the complete device
        membrane = lib.new_cell("MEMBRANEA" + str(i))

        # Add 2 references to the component changing size and orientation
        ref1 = gdspy.CellReference(cutout, (0, a1))
        ref2 = gdspy.CellReference(cutout, (a1, 0), rotation=-90)
        ref3 = gdspy.CellReference(cutout, (0, -a1), rotation=180)
        ref4 = gdspy.CellReference(cutout, (-a1, 0), rotation=90)

        membrane.add([ref1, ref2, ref3, ref4])

        # Create a cell with a component that is used repeatedly

        outs_x = c / 2

        outs_y = c / 2 * 1.5

        ins_x = outs_x - margin

        ins_y = outs_y - margin
```

```
        chip = lib.new_cell("CHIPA" + str(i))

        parameter = gdspy.Polygon(
            [(-outs_x, -outs_y), (outs_x, -outs_y), (outs_x, outs_y), (-outs_x, outs_y), (-outs_x, -outs_y),
             (-ins_x, -ins_y), (-ins_x, ins_y), (ins_x, ins_y), (ins_x, -ins_y), (-ins_x, -ins_y)])

        y_off = -4000

        ref5 = gdspy.CellReference(membrane, (0, 3325))

        # Horizontal text with height 2.25
        htext = gdspy.Text(roman_num(i), 1000, (-1000, -8000))

        offset = 3787.5

        c1 = gdspy.Polygon([(0 + offset, y_off), (-604 + offset, -604 + y_off), (-604 + offset, 604 + y_off),
                            (604 + offset, 604 + y off), (604 + offset, -604 + y off), (-604 + offset, -604 + y off)])

        c2 = gdspy.Polygon([(0 - offset, 0 + y_off), (-604 - offset, -604 + y_off), (-604 - offset, 604 + y_off),
                            (604 - offset, 604 + y_off), (604 - offset, -604 + y_off), (-604 - offset, -604 + y_off)])
        chip.add(c1)
        chip.add(c2)

        chip.add([parameter, ref5])

        chip.add(htext)

        if i / 5 == 1:
            T = -52725
            R = R + (c - margin)

        if i / 6 == 1:
            T = 17575

        T = T + (1.5 * c - margin)

        if i / 4 == 1 or i / 6 == 1:
            ref6 = gdspy.CellReference(chip, (R, T), 180)
            ref7 = gdspy.CellReference(chip, (-R, T), 180)
        else:
            ref6 = gdspy.CellReference(chip, (R, T))
            ref7 = gdspy.CellReference(chip, (-R, T))

        wafer.add(ref6)
        wafer.add(ref7)


def makechipsB(s_min, s_max, theta_min, theta_max, r, T, R, mem, c, mass, margin):
    T = T - (c - margin)

    for i in range(r):

        # Inputs
        w = (s_max - s_min) / (r - 1)

        E = (theta_max - theta_min) / (r - 1)
        # Tether width
        s1 = s_min + w * i  # um

        k = s1 / math.sqrt(2)

        # Use complex numbers to facilitate writing polar coordinates
        c3 = gdspy.Curve(-mass + k, 0).C((-theta_min - i * E) + k, -1000, (theta_min + i * E) - k, -1000, mass - k, 0)
        # Elliptical arcs have syntax similar to gdspy.Round

        c0 = gdspy.Polygon(c3.get_points(), **ld_photomask)

        c0.fillet(700)

        a1 = mass
        # Create a cell with a component that is used repeatedly
        cutout = lib.new cell("CUTOUTB" + str(i))

        cutout.add(c0)

        # Create a cell with the complete device
        membrane = lib.new_cell("MEMBRANEB" + str(i))

        # Add 2 references to the component changing size and orientation
        ref1 = gdspy.CellReference(cutout, (0, a1))
        ref2 = gdspy.CellReference(cutout, (a1, 0), rotation=-90)
        ref3 = gdspy.CellReference(cutout, (0, -a1), rotation=180)
        ref4 = gdspy.CellReference(cutout, (-a1, 0), rotation=90)

        membrane.add([ref1, ref2, ref3, ref4])

        # Create a cell with a component that is used repeatedly
```

```
        outs_x = c / 2

        outs y = c / 2

        ins_x = outs_x - margin

        ins_y = outs_y - margin

        chip = lib.new_cell("CHIPB" + str(i))

        parameter = gdspy.Polygon(
            [(-outs_x, -outs_y), (outs_x, -outs_y), (outs_x, outs_y), (-outs_x, outs_y), (-outs_x, -outs_y),
             (-ins_x, -ins_y), (-ins_x, ins_y), (ins_x, ins_y), (ins_x, -ins_y), (-ins_x, -ins_y)])

        offset = 3787.5
        y_off = -2500
        ref5 = gdspy.CellReference(membrane, (0, 1787.5))
        c1 = gdspy.Polygon([(0 + offset, y_off), (-604 + offset, -604 + y_off), (-604 + offset, 604 + y_off),
                            (604 + offset, 604 + y_off), (604 + offset, -604 + y_off), (-604 + offset, -604 + y_off)])

        c2 = gdspy.Polygon([(0 - offset, 0 + y_off), (-604 - offset, -604 + y_off), (-604 - offset, 604 + y_off),
                            (604 - offset, 604 + y_off), (604 - offset, -604 + y_off), (-604 - offset, -604 + y_off)])
        chip.add(c1)
        chip.add(c2)

        chip.add([parameter, ref5])

        # Horizontal text with height 2.25
        htext = gdspy.Text(roman_num(i), 750, (-750, -5000))
        chip.add(htext)

        if i / 2 == 1:
            T = -32150

        if i / 4 == 1:
            R = R + (c - margin)
            T = -40725

        T = T + (c - margin)

        if i / 9 == 1:
            ref6 = gdspy.CellReference(chip, (R, T), 180)
            ref7 = gdspy.CellReference(chip, (-R, T), 180)
        else:
            ref6 = gdspy.CellReference(chip, (R, T))
            ref7 = gdspy.CellReference(chip, (-R, T))

        wafer.add(ref6)
        wafer.add(ref7)

    print(R + (9000 / 2 + c / 2 - margin))


def makechipsC(s_min, s_max, theta_min, theta_max, r, T, R, mem, c, mass, margin):
    for i in range(r):
        # Inputs
        w = (s_max - s_min) / (r - 1)

        E = (theta_max - theta_min) / (r - 1)
        # Tether width
        s1 = s min + w * i  # um

        k = s1 / math.sqrt(2)

        # Use complex numbers to facilitate writing polar coordinates
        c3 = gdspy.Curve(-mass + k, 0).C((-theta_min - i * E) + k, -1000, (theta_min + i * E) - k, -1000, mass - k, 0)
        # Elliptical arcs have syntax similar to gdspy.Round

        c0 = gdspy.Polygon(c3.get_points(), **ld_photomask)

        c0.fillet(700)

        a1 = mass
        # Create a cell with a component that is used repeatedly
        cutout = lib.new cell("CUTOUTC" + str(i))

        cutout.add(c0)

        # Create a cell with the complete device
        membrane = lib.new_cell("MEMBRANEC" + str(i))

        # Add 2 references to the component changing size and orientation
        ref1 = gdspy.CellReference(cutout, (0, a1))
        ref2 = gdspy.CellReference(cutout, (a1, 0), rotation=-90)
        ref3 = gdspy.CellReference(cutout, (0, -a1), rotation=180)
        ref4 = gdspy.CellReference(cutout, (-a1, 0), rotation=90)
```

```
        membrane.add([ref1, ref2, ref3, ref4])

        # Create a cell with a component that is used repeatedly

        outs_x = c / 2

        outs y = c / 2

        ins_x = outs_x - margin

        ins_y = outs_y - margin

        chip = lib.new_cell("CHIPC" + str(i))

        Border = lib.new cell("Border" + str(i))

        points = [(-outs_x, -outs_y), (outs_x, -outs_y), (outs_x, outs_y), (-outs_x, outs_y), (-outs_x, -outs_y),
                  (-ins x, -ins y), (-ins x, ins y), (ins x, ins y), (ins x, -ins y), (-ins x, -ins y)]

        poly = gdspy.Polygon(points)

        Border.add(poly)

        ref14 = gdspy.CellReference(Border, (0, 0))

        chip.add(ref14)

        htext = gdspy.Text(roman_num(i), 500, (-300, -3500))
        chip.add(htext)

        offset = 2500
        y_off = -2000
        ref5 = gdspy.CellReference(membrane, (0, 1787.5))
        c1 = gdspy.Polygon([(0 + offset, y_off), (-604 + offset, -604 + y_off), (-604 + offset, 604 + y_off),
                            (604 + offset, 604 + y_off), (604 + offset, -604 + y_off), (-604 + offset, -604 + y_off)])

        c2 = gdspy.Polygon([(0 - offset, 0 + y off), (-604 - offset, -604 + y off), (-604 - offset, 604 + y off),
                            (604 - offset, 604 + y_off), (604 - offset, -604 + y_off), (-604 - offset, -604 + y_off)])
        chip.add(c1)
        chip.add(c2)

        T = T + (c - margin)

        chip.add(ref5)
        ref6 = gdspy.CellReference(chip, (R, T))
        ref7 = gdspy.CellReference(chip, (-R, T))

        wafer.add(ref6)
        wafer.add(ref7)


##makechips(s_min,s_max,theta_min,theta_max,r,T,R,mem,c,mass,margin)
makechipsA(300, 1500, 1000, 2000, 7, -52725, 5787.5, 6000, 12000, 3000, 425);
makechipsB(200, 750, 500, 1000, 10, 9000, 17362.5, 4000, 12000, 2000, 425);
makechipsC(200, 750, 500, 1000, 5, -26425, 39012.5, 3000, 9000, 1500, 425);

c1 = gdspy.Polygon([(-18000, -45400), (18000, -45400), (18000, -45800), (-18000, -45800), (-18000, -45400)],
                   **ld_photomask)
c2 = gdspy.Polygon([(-18000, -46400), (18000, -46400), (18000, -46800), (-18000, -46800), (-18000, -46400)],
                   **ld_photomask)
c3 = gdspy.Polygon([(-18000, -47400), (18000, -47400), (18000, -47800), (-18000, -47800), (-18000, -47400)],
                   **ld photomask)
c4 = gdspy.Polygon([(-18000, -48400), (18000, -48400), (18000, -48800), (-18000, -48800), (-18000, -48400)],
                   **ld_photomask)

wafer.add(c1)
wafer.add(c2)
wafer.add(c3)
wafer.add(c4)

allignment = lib.new_cell("ALLIGNMENT")
cross = gdspy.Path(200, (0, 0))

# Add a segment to the path goin in the '+y' direction
cross.segment(1000, "+y")
cross.segment(2000, "-y")
cross.segment(1000, "+y")
cross.segment(1000, "+x")
cross.segment(2000, "-x")

allignment.add(cross)

ref12 = gdspy.CellReference(allignment, (19000, 0))
ref13 = gdspy.CellReference(allignment, (-19000, 0))

wafer.add([ref12, ref13])

nallignment = lib.new_cell("NALLIGNMENT")
```

```
points = [(-100, 100), (-1000, 100), (-1000, 1000), (-100, 1000), (-100, 100)]
points1 = [(100, 100), (1000, 100), (1000, 1000), (100, 1000), (100, 100)]
points2 = [(-100, -100), (-1000, -100), (-1000, -1000), (-100, -1000), (-100, -100)]
points3 = [(100, -100), (1000, -100), (1000, -1000), (100, -1000), (100, -100)]

poly = gdspy.Polygon(points)
nallignment.add(poly)
poly1 = gdspy.Polygon(points1)
nallignment.add(poly1)
poly2 = gdspy.Polygon(points2)
nallignment.add(poly2)
poly3 = gdspy.Polygon(points3)
nallignment.add(poly3)

ref14 = gdspy.CellReference(nallignment, (17000, 0))
ref15 = gdspy.CellReference(nallignment, (-17000, 0))

htext = gdspy.Text("RSG_LAB/NRC UOTTAWA 09/2020", 1000, (-9575, 45000))
wafer.add(htext)

wafer.add([ref14, ref15])
lib.write_gds('Photomask.gds')
wafer.write_svg('Photomask.svg')
lib.write_gds('PHOTOMASK.gds')
```