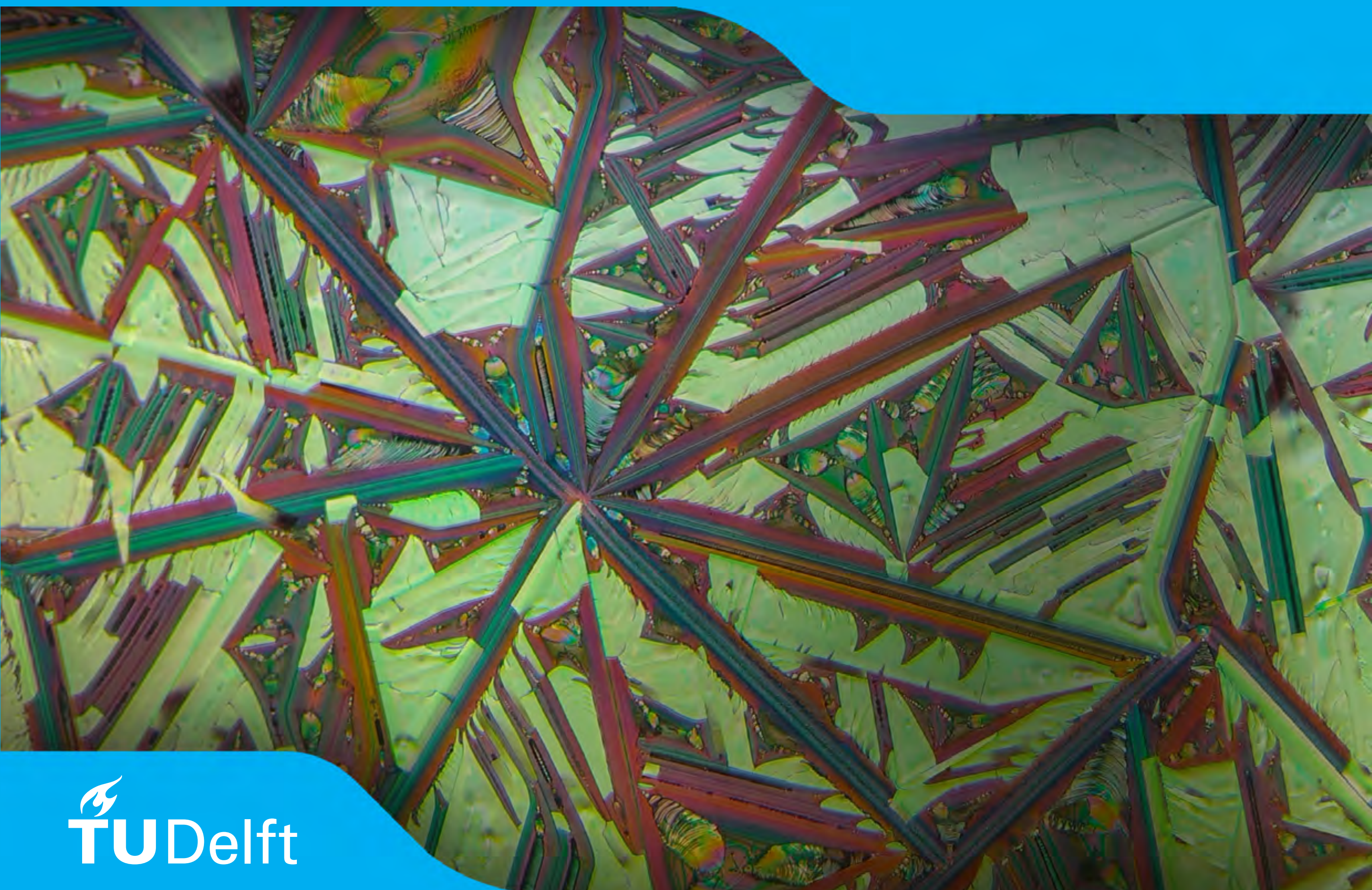
PASiC Ammonia Sensor Design

On Modelling the PASiC Layer, Electrode Designs and Device Fabrication

Jasper Rietveld



TU Delft

PASiC Ammonia Sensor Design

On Modelling the PASiC Layer, Electrode Designs and Device Fabrication

by

Jasper Rietveld

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on TBD

Student number:	4581881	
Project duration:	January, 2021 – September, 2022	
Supervisor:	Prof. Dr. Paddy French,	TU Delft
Thesis committee:	Prof. Dr. Paddy French,	TU Delft
	Dr. Ir. Sten Vollebregt,	TU Delft

An electronic version of this thesis is available at https://repository.tudelft.nl/.

TU Delft

Abstract

Capacitive, interdigital sensors in proximity of a PASiC thin film have been proposed as a cheap, robust alternative to existing ammonia sensing technologies. However, although prototype sensors have been manufactured, investigation into the working principles of this class of ammonia sensors is limited. This work aims to analyse the electrode and PASiC layer configuration to gain more insight into how design parameters influence the sensing capabilities of these devices.

To this end, finite element simulations and analytical simulations have been carried out to model the behaviour of the sensor configuration. The PASiC layer is modelled as a stack of layers with different permittivities, with the response to ammonia modelled as a change in the permittivity of the layers.

Using the simulation data, a lithography mask for a test-bench with varying electrode dimensions is designed to allow fabrication of devices for measuring the effects of different manufacturing parameters. A series of prototypes created using this mask is then used to devise an etching setup to allow electrochemical etching of single chips to create the porous layer.

The results of the manufacturing process are analysed and discussed.

Preface

This report is the thesis on the MSc thesis project proposal ”*[2020] Ammonia & humidity sensors*”, supervised by prof. Dr. Paddy French at the TU Delft. It documents the research done, including the results a series of simulations, the process of designing and fabricating a series of prototypes, and the results of measurements conducted using these prototypes.

Jasper Rietveld
Delft, September 2022

Contents

1 Introduction **1**

2 State of the art **3**
2.1 Applications of ammonia sensing . . . 3
2.2 Ammonia sensing methods . . . 3
2.2.1 Solid-state sensing methods . . . 3
2.2.2 Optical sensing methods . . . 3
2.2.3 Electrochemical sensing methods . . . 4
2.2.4 Porous Semiconductor sensing methods . . . 4
2.3 Porous Semiconductor based sensors . . . 5
2.3.1 Porous Silicon based sensors . . . 5
2.3.2 Porous Carbide based sensors . . . 5
2.3.3 PASiC fabrication . . . 5
2.4 Conclusions. . . . 6

3 Programme of requirements **7**
3.1 Mandatory requirements . . . 7
3.2 trade-off requirements . . . 7
3.2.1 accuracy . . . 7
3.2.2 response and recovery time . . . 7
3.2.3 Cross sensitivity . . . 8
3.2.4 Reliability . . . 8
3.2.5 Manufacturing process . . . 8
3.2.6 Safety . . . 8

4 Design of the Sensor Structure and Layout **9**
4.1 design approach . . . 9
4.2 Sensor structure . . . 9
4.3 Sensor electrode layout . . . 9
4.3.1 Sensing Capacitor Design . . . 9
4.3.2 Die design . . . 12
4.4 conclusion . . . 13

5 Simulation of the Sensor Structure **15**
5.1 Simulation approach . . . 15
5.1.1 Approach in modelling the multi-layered sensor structure . . . 15
5.1.2 Cross-sectional simulations . . . 16
5.1.3 Modelling of the PASiC layer. . . . 16
5.1.4 Modelling of the porosity distribution of the PASiC layer . . . 18
5.1.5 cross-sectional simulation in COMSOL . . . 21
5.1.6 cross-sectional analytical model using MATLAB . . . 22
5.1.7 3D device simulations in COMSOL . . . 25
5.2 Simulation results. . . . 26
5.2.1 Cross-sectional simulations using COMSOL . . . 26
5.2.2 Cross-sectional Simulations Using MATLAB . . . 29
5.2.3 Simulation of the integrated device . . . 30
5.3 Conclusions. . . . 32

6 Design of the lithography mask **33**
6.1 Capacitor design and variation. . . . 33
6.2 Capacitor placement on the Mask . . . 34
6.2.1 Automated Die Generation and Placement . . . 34

6.3 Mask overview . . . 35
6.4 Conclusions. . . . 35

7 Prototype Fabrication 37
7.1 Fabrication of the Capacitor Array . . . 37
7.1.1 Preparation of the wafers with Silicon Carbide substrate. . . . 37
7.1.2 Electrode Fabrication Using Lift-off Process . . . 37
7.2 Forming the PASiC layer . . . 38
7.2.1 Design of a current source for electrochemical etching. . . . 38
7.2.2 Design of a chip holder. . . . 39
7.2.3 Electrochemical etching setup and procedures . . . 41
7.3 Conclusions. . . . 41

8 Device Inspection and Measurement Results 43
8.1 Device inspections . . . 43
8.1.1 inspections during and following the lift-off . . . 43
8.1.2 Inspections after dicing. . . . 46
8.1.3 Inspection of the devices after etching . . . 49
8.2 Measurements . . . 53
8.3 Conclusions. . . . 54

9 Conclusions 55
9.1 Conclusions. . . . 55
9.2 Recommendations for future work. . . . 55

A Appendices 57
A.1 MATLAB Simulation Code . . . 57
A.1.1 Plot Layers . . . 57
A.1.2 Plot Layer Models . . . 59
A.1.3 Plot Configurations . . . 61
A.1.4 Plot Sensitivity . . . 62
A.1.5 Model Sensitivity . . . 65
A.1.6 Evaluate Capacitance . . . 68
A.1.7 Plot Precision Errors . . . 71
A.2 COMSOL Simulation Code . . . 72
A.2.1 Generate Pores. . . . 72
A.2.2 Porosity Model . . . 73
A.2.3 Porosity Model Expression. . . . 74
A.2.4 Apply Porosity Model. . . . 75
A.2.5 Remove Layer Data . . . 76
A.3 Mask Capacitor Dimensions . . . 77
A.4 Capacitor Reference Tables . . . 78
A.5 Current Source . . . 84
A.5.1 Current source measurements. . . . 84
A.5.2 Plot generation code . . . 84
A.6 Mask Generation Application Overview . . . 86
A.6.1 Application Usage . . . 86
A.6.2 Further remarks. . . . 92
A.7 Mask Generation Application COMSOL code. . . . 92
A.7.1 Generate Wafer Reference . . . 92
A.7.2 Generate Text. . . . 94
A.7.3 Generate Die Previews. . . . 97
A.7.4 Generate Capacitor . . . 99
A.7.5 Generate Cutmarks . . . 99
A.7.6 Generate Dies . . . 100
A.7.7 Generate Die . . . 108
A.7.8 Clear Geometry. . . . 109

A.8 Process flowcharts . . . 110
A.8.1 Preparation of wafers with SiC layer . . . 110
A.8.2 Fabrication of Au/Cr planar capacitors . . . 112
A.9 MATLAB Simulation Code . . . 114
A.9.1 Plot Measurements . . . 115
Bibliography **119**

1 Introduction

Ammonia (NH_3) sensors are in high demand because of their widespread use in multiple important industries. In these industries, ammonia is usually emitted as a byproduct of various processes. One of the primary sectors in which ammonia is used is the agricultural sector, as it is both a primary component of fertilisers and is emitted by livestock. [1] Ammonia is also present in many chemical industries. Some common applications include usage as a refrigerant in large cooling installations, the usage as a cleaning solvent, or water treatment. Another industry in which Ammonia is present is the automotive industry, where it is emitted as it is a byproduct in combustion engines, primarily ones fuelled by diesel. [2]

Monitoring of ammonia gas levels in these sectors is of importance for a number of reasons. Firstly, Exposure to ammonia gas can form a serious threat to human health. Ammonia is a rather corrosive substance. It dissolves readily in water to form vapours with high acidity. At lower concentrations, this can cause irritation of the respiratory tract, especially if the exposure time is long. [3] As sources of ammonia in many industries can cause increased ammonia gas concentrations within the workspace to which workers can be subjected for extended periods of time. In industrial applications exposure to very high concentrations is also possible, for instance as a result of gas leaks. Ammonia is often stored in pressurised containers. If a large leak of ammonia occurs, the released high concentrations pose a considerable threat. It can cause damage to the skin and eyes, or even fatal or severe respiratory system disorders. [3] Another risk relating to ammonia leaks is that at higher concentrations the gas can form explosive mixtures with air.[2] For these reasons, sensors are required in industrial sectors where workers could be exposed to ammonia in the air in order to prevent exposure. For this it's generally desirable to develop sensitive ammonia sensors capable of detecting low concentrations. A fast response is also necessary to ensure timely notification especially in the presence of high concentrations.

The relevance of the reactivity of Ammonia goes beyond the risk of human health. Ammonia gas can deteriorate sensors over time, which makes the reliability of sensors for ammonia gas a challenging aspect. Therefore, sensors are preferably made from especially stable materials to increase their lifetime in corrosive sensing environment. [4]

In the automotive industry, the concentration of ammonia gasses in the exhaust and engine can be an indicator of engine performance. [5] The injection of ammonia into engines can influence the emission of harmful nitrous oxides, which are one of the main pollutants emitted by diesel engines. [6] Ammonia is even investigated as a carbon-free fuel for compression engines. [7]

Furthermore, ammonia has a negative effect on the environment. Emissions can result in a decrease of the biodiversity, primarily through the processes of eutrophication and acidification.

Therefore, ammonia is considered a pollutant. As such, with the increasingly stricter governmental regulations concerning the emission of gasses, and the increasing concern of the public relating to the environment, there has been a growing demand for NH3 sensors to monitor these environmental emissions. This requires sensitive sensors that can be used reliably for longer times, although the response time can be slow.

There has also been interest in measuring the levels of NH3 for other medical purposes. One application is measuring the ammonia concentration in exhaled air, as these concentrations can be

used for the diagnosis of certain (respiratory) diseases. [8] For reliable diagnosis, a high sensitivity is required.

All in all, these applications require sensors that are able to accurately, quickly, selectively, reliably and cheaply measure concentrations of ammonia. [2]

This work aims to report on the analysis of ammonia sensors based on a planar interdigital capacitance with a layer of porous silicon carbide. A series of sensor devices was designed, modelled and in part fabricated to measure and characterise their accuracy and speed of Ammonia measurements. By varying the etching parameters and dimensional parameters of the capacitor geometry, an attempt is made at finding a relation between these parameters and the measurement performance, and an optimum is sought within these relations.

This report starts with an overview of the state of ammonia sensors and its applications in Chapter 2. Following this, the requirements for the design process are described in Chapter 3. Based on these requirements, the design of the sensors is reported on, including the design of the device layout based on the requirements in Chapter 4, simulations of the geometry in Chapter 5, and the mask design along with documentation of tools used in Chapter 6. The process of carrying out the fabrication of the design and conducting the measurements is described in Chapter 8. The results of the fabrication process and some measurements are then documented and interpreted in Chapter 7, and a summary of the findings is given in Chapter 9.

2
State of the art

Over the years, ammonia sensors have been continually improved and have been optimised in different aspects for the different fields where ammonia sensors are used. As such, many new sensor designs are being developed to further improve in these regards.

This section will evaluate the state of the art in ammonia sensor design to give an overview of the sensing methods that have currently been developed.

2.1. Applications of ammonia sensing

2.2. Ammonia sensing methods

There exists a great variety of different ammonia sensing methods each with their own benefits and applications. Kwak et al. categorise the most prevalent of these methods into three categories, namely solid-state sensing methods, optical methods and other methods. [9].

2.2.1. Solid-state sensing methods

Solid-state sensing methods include methods without moving parts or liquids.

Within this category, one finds metal oxide-based sensors. This type of sensors use an oxide material of which the conductivity changes under the presence of ammonia. While n-type and p-type sensors are possible, n-type sensors are more commonly used as they have a higher sensitivity. With an n-type semiconducting metal oxide, the conductivity under baseline circumstances is low, as charged oxygen at the surface of the oxide traps electrons, forming a depletion layer. Reducing gases react with this oxygen at the surface, causing the conductivity to increase. The specific oxide used determines what gases the device is sensitive to. However, the selectivity is comparatively low. That being said, the device is process compatible, cost efficient a number of methods can be employed to obtain a high sensitivity to NH3.

This category also includes conducting polymer sensors. A variety of conducting polymers can be used, which in the presence of NH_3 have measurable changes in voltage over the material, current through the material, colour, or weight. Most commonly amperometric mode sensors are used, which change conductance through a redox reaction with the environmental ammonia. The change of the polymer can be detected in a multitude of configurations, though the chemiresistor is the most popular because of it's low detection limit and relatively high response time. Polymer sensors in general do not have a fast response time, and have low structural strength. Structure modifications of the polymers, or the use of dopants can enhance these parameters. Advantages of polymer sensors include that they have excellent stability, and are relatively easy to fabricate.

2.2.2. Optical sensing methods

It is also possible to detect ammonia gas using absorption based optical methods. Gasses absorb light in a narrow frequency band that is characteristic for a certain gas. In the case of NH_3 this peak lies in the infrared part of the spectrum. Using optical methods, the presence of Ammonia can be detected from the absorption of the spectrum near these frequencies.

A particularly well-developed technique within this category is Tunable Diode Laser Absorption Spectography (TDLAS). In this technique, light from a laser emitter, tuned to the specific wavelength required for the detection of Ammonia, is split and is directed into a gas chamber and a reference cell. The gas mixture to be tested flows through the gas chamber. To increase the length of the optical path, mirrors are placed on either side of the chamber. Both the reference cell and the gas chamber contain a photodetector like a photodiode to measure the incoming light. The measurement from the reference cell and the gas chamber are compared to determine the concentration of ammonia in the gas chamber.

This form of detection has numerous advantages. It has great selectivity and sensitivity, is simple to operate, and does not influence the gas. However, advanced laser emitters are required, which depending on the laser emitter can require cryogenic temperature control. Additionally a the optimal optical path length for the detection is large. This makes the setup relatively complex, and rather sizeable, which limits it's usage in applications which require mobile operation.

Another optical sensing method that reduces the required size relating to the path length is Cavity Ringdown Spectroscopy (CRDS). This method uses a single laser pulse sent into a gas filled spectral cavity between with highly reflective mirrors, as opposed to a continuous beam in the case of TDLAS. A photodetector measures the decay of the pulse to determine the composition of the analyte gas. This way, a large opStical path is attained with a much smaller chamber.

2.2.3. Electrochemical sensing methods

Other methods include electrochemical sensors using a liquid or solid electrolite, surface acoustic wave (SAW) sensors, and field effect transistor based sensors.

Electrochemical sensors measure the using electrodes places inside an electrolyte. Gas flows through a gas permeable membrane into the electrolyte. When ammonia gas diffuses through this membrane, it reacts with the electrolyte. This reaction results in a measurable change in the voltage between or the current through the electrodes. The choice of electrolyte largely determines the characteristics of the sensor. Sensors using a liquid electrolytes suffer from a low lifetime as the electrochemical reaction consumes the electrolyte. If a solid electrolyte is used, a high operating temperature is required, putting it at risk of electrolyte poisoning. These drawbacks are offset with a great sensitivity, a high selectivity, low manufacturing costs and good portability. The usage of Room Temperature Ionic Liquids (RTILs) have been investigated to limit the drawbacks. These liquids exhibit desirable properties, including working at a lower temperature with a high potential window, high polarity and high viscosity. A downside of RTILs, however, is that the viscosity also slows down the mass transport and diffusion. The choice of material for the substrate and electrodes also enhances the performance of these sensors, as these govern the reactions that take place at the electrode, which vary in sensitivity and selectivity. Usually noble and robust materials are chosen to increase the lifetime.

Surface acoustic wave (SAW) sensors excite a piezoelectric substrate, typically using Interdigitated Transducers (IDTs). On one end of the substrate, signal processing electronics use an IDT to create a standing wave inside the material, which travels across the substrate to the other side of the device. On the other side of the device, a second IDT senses the standing wave and converts it back into a signal which can then be processed. Between the two IDTs lies a delay line which can be used to control the parameters of the standing wave. The gasses present near the surface of the device change the properties of the standing wave, such as the amplitude and wavelength. The mass of the gas present determines the frequency shift that is detected. Since this is characteristic for certain gasses, a sensor that is selective to NH_3 can be fabricated.

2.2.4. Porous Semiconductor sensing methods

Another class of ammonia sensors is the porous semiconductor based gas sensor. While this could technically be classified as a solid state sensing method, it is considered separately here as it will be reviewed in more detail.

The fabrication of porous silicon was investigated as early as the 1950s. It was found that Silicon nanolayers would become porous when being subjected to certain electrochemical etching conditions. During this and the following decades, numerous studies were conducted to different techniques to fabricate similar porous materials as well as the theory behind these processes. [10]

Later studies revealed that the electrical properties of the porous material changed under the presence of certain gases. Initially, these experiments primarily involved humidity sensing. For instance,

Connoly et al. have created a humidity sensor based on an interdigital capacitor on top of a film of porous polysilicon. [11] Further research showed that other gases, including ammonia and various other polar molecules, would also influence the material properties. [12]

These discoveries have lead to the development of a range of different porous semiconductor based sensing methods, including photometric, conductometric and capacitive sensors. [10]

2.3. Porous Semiconductor based sensors

Different porous materials and sensing mechanisms can be employed within this category of sensors. This section will list a number of different solutions, along with relevant information on their attributes.

2.3.1. Porous Silicon based sensors

Porous Silicon (PS) has been one of the most common materials in this type of research. As it was the first material to be used in this type of sensors, it has been applied in numerous different devices.

The state-of-the-art includes a variety of sensor types. A major discovery has been the photoluminescent effects of PS. This has been used to create various photometric sensors.[10] Althought PS was initially primarily used for RH sensing, ammonia sensors have been developed using the technology.

Another method of ammonia sensing using porous silicon is reported in [13]. Silicon is made porous by using anodisation with 50% HF. By using two different current densities consecutively, two porous layers with a different porosity and thus different refractive index were manufactured, resulting in a waveguide like structure. This structure is then impregnated with Bromothymol blue, which changes colour under the presence of Ammonia. The resulting structure retains this optical property, and can as such be used to measure the ammonia concentration using optical methods.

Seals et al. report the use of a design of a conductiometric ammonia sensor based on PS. [12] A hybrid pore structure is used, with a nanoporous structure on top of a microporous layer. Two interdigitated electrodes are placed atop this structure, and the impedance between the electrodes is measured. The sensor shows a significant increase in impedance in the presence of gasses such as NH_3, NO and HCl. This change has a notably faster response than other PS sensors, which is reported to suggest an easily reversible surface reaction. Absorption of the molecules into the pores is attributed to either van der Waals interactions, dipole-dipole interactions and/or electron exchange with surface states, though no research is done to determine the contribution of each of these possibilities.

2.3.2. Porous Carbide based sensors

An alternative to Porous Silicon is the usage of Porous Silicon carbide. pSiC has been proposed as an alternative to PS for it's excellent stability. The material has great heat resistance and conductivity, good mechanical strength (to some extent depending on the pore structure), and chemical resistance. [14] [15]

A particular variant of the material uses amorphous silicon carbide as the starting material. This is then etched to form porous amorphous silicon carbide (PASiC). The material being amorphous allows for a cellular pore size that's comparatively easy to fabricate and control the pore dimensions of. Using electrochemical etching in HF, Connoly et al. have developed a PASiC based capacitive sensor which is sensitive to ammonia. [16]

2.3.3. PASiC fabrication

Controlling the structure of the pores in the material is a major aspect of sensor development using porous materials. Porous SiC comes in a variety of forms. Aside from the pore size, SiC comes in a variety of morphologies, some more suitable than other for the fabrication of ammonia sensors.[17] The carbon density of the material also affects it's mechanical and electrical properties. There is an extensive range of techniques to fabricate SiC, with varying practical applications and parameters to control the properties of the resulting porous SiC. This section will list a number of these methods.

For the creation of SiC layers on crystalline wafers, other methods are more commonly used.

Electrochemical etching of Silicon Carbide The first and most prominent method of PASiC fabrication electrochemical etching. The SiC or SiC thin-film on a wafer is used as the anode in an electrochemical etching setup, usually with a platinum cathode. The etching process predominantly uses HF as the main acid in the electrolyte, along with various additives to influence the pore characteristics

of the resulting thin-film. Dao et al. show how the pore size in this process depends on the HF concentration and etching current. It is also shown that the morphology of the pores is cellular if Triton X surfactant is used. [18]

Photochemical etching of Silicon carbide An alternative to electrochemical etching is metal assisted photochemical etching (MAPCE). Here, a thin film of a metal is first deposited on the SiC surface, before the material is etched in a solution containing HF and an oxidation agent. UV light is used to generates holes to serve as a starting point for the pore formation. The Oxidation agent then consume electrons that move from the SiC into the electrolyte. [19]

Leitgeb et al. characterises the process for a number of oxidising agents and using Pt as the noble metal. It is shown that the pore size increases linearly with time during this process, until the pores connect and the pore size stays relatively constant. The resulting modal pore area ranges from $200\,\mathrm{nm}^2$ to $1000\,\mathrm{nm}^2$. [19] Boukezzata et al. use Aluminium as the metal and $AgNO_3$ as the oxidising agent to create layer of PASiC. This is then used to prototype a resistive sensor for ammonia. [20]

Senthilnathan et al. selectively etch silicon from SiC bar to create carbon-rich mesoporous SiC layer by electrochemical etching using HF and acetonitrile. With a current density of 30 mA/cm2 a pore size in the range of $10\,\mathrm{nm}$ to $100\,\mathrm{nm}$ was obtained. These are tested as absorbents for the removal of volatile organic compounds from air. [21]

Other methods Other methods for the fabrication of Silicon Carbide suitable for sensors exist, though they are not as compatible to microelectronics technologies and on-wafer devices.

A popular method is the use of sintering. In this process, SiC powder is compressed under high pressure and subsequently sintered at high temperature to prodous a porous material. The morphology and size of the pores can be controlled with a range of parameters, including the temperature, pressure and powder grain size. [22]

Another method of fabricating the porous SiC layer is by employed by Li et al., which use freeze-casting to create microporous structures.[23] Here, a ceramic slurry is first frozen, so that the ceramic particles accumulate around the forming ice front. The solidified ice is then sublimated to leave only the SiC. Sintering is then used to strengthen the resulting structure. It is demonstrated that both lamellar and cellular porous SiC can be formed this way, with pore sizes of about $2.5\,\mu\mathrm{m}$.

2.4. Conclusions

An assortment of different sensor types for detecting Ammonia levels has been developed, making use of different physical mechanisms. Aside from limitations in these mechanisms, trade-offs are generally made between desirable sensing characteristics such as response time, system complexity and sensitivity. In this way, the different technologies may serve different applications.

Capacitive, PASiC based sensors may provide low-cost, sensitive devices with excellent stability while being subject to the corrosive ammonia sensing conditions. However, the devices have not been fully optimised as further characterisations of the technology are required.

In a review on gas sensors based on porous materials, Korotcenkov et al. conclude that although the technology has promising prospects, more characterisation is needed to determine what parameters of the porous material influence the gas sensing capabilities. [10] The diameter, length and surface density of the pores are proposed as variables that are likely the most influential in the working of these devices.

This work aims to work towards such a characterisation for a PASiC based capacitive sensor, by modelling and fabricating devices with varying electrode configurations and etching parameters.

3 Programme of requirements

Within the design of the sensor, a number of attributes can be considered to improve the suitability for applications. As requirements vary greatly between different specific applications, and this project involves research towards the optimisation of these parameters, strict requirements can not easily be formulated. Nevertheless, it is valuable to document these parameters so that they can serve as a guide in the design process. This section therefore enumerates these elements, remarking on their relevance in different fields. As such, it functions as a reference for later sections to refer to support choices made in design trade-offs.

The requirements are found following notes made during the literature study on discussed trade-offs, comparisons between previous work on ammonia sensing and more general design space exploration techniques.

3.1. Mandatory requirements

As mandatory functional requirements for the sensor, any sensing capability is firstly required. As the device is based on capacitive sensing, this can be concisely described as a requirement of a measurable change in capacitance relating to a change in the NH_3 concentration of the sensing environment. In order for this relation to be established the change needs to be repeatable given the same starting and environmental conditions.

3.2. trade-off requirements

On top of that there are a number of desirable attributes on which trade-offs are made between sensors, examples of which were seen in the introduction. For the purpose of this report, trade-offs are made to the benefit of characterising the influence of the electrode dimensions on the sensing characteristics.

3.2.1. accuracy

Firstly, the sensor should be accurate. In practice this means that it should be possible to infer the Ammonia concentration in the sensing environment from the measured capacitance with a low error. Therefore, a well defined relation between this capacitance and this concentration is to be established. This relation is heavily dependent on the electrode configuration and therefore of primary interest in the electrode design. Influence of the measurement setup and connecting electronics has to be limited or characterised. Further more, the influence from outside parameters, such as electrical interference have to be limited. The sensor's time stability, with parameters such as drift or other (low-frequency) noise can also be categorised as relating to the accuracy.

3.2.2. response and recovery time

Two other parameters of interest are the response time and the recovery time. These are usually defined as the time it takes for a sensor to settle to a response, and the time to return to the baseline value respectively, both with some margin. A fast response and recovery time makes the device more suitable for applications where the concentration is expected to change rapidly, or where consecutive

measurements in environments with different concentrations are desirable. For the capacitive sensor in this report the response time is reported as the time for the capacitive sensor to reach 90% of the of the maximum value the response settles to, and the recovery time as a return to 10% of this response. The recovery time is especially relevant for this technology, as it involves a porous structure. Gas needs to move in and out of these pores in order for the sensor to go back to the baseline value. If this movement is low, the response time will also be low.

3.2.3. Cross sensitivity

The cross-sensitivity is also exceptionally relevant. As discussed in , previous studies on this technology have also demonstrated a sensitivity to humidity. Sensitivity is also expected for other polar molecules. As many applications of ammonia sensing are in environments with a changing humidity, it is valuable to determine the cross-sensitivity to other gasses like these.

3.2.4. Reliability

Requirements can also be formulated for the reliability of the device, to increase the sensor's lifetime. Ideally the sensor is made from chemically stable materials, as Ammonia is a corrosive gas and depending on the application the sensing environment can be at a high temperature. There are many more factors that can be considered relating to reliability, but these apply more generally to electronics. Examples include ESD protection, mitigation of material migration, or prevention of material contamination with packaging.

3.2.5. Manufacturing process

As non-functional requirements, a number of attributes of the manufacturing process can give advantages to a sensor. For economic reasons, simpler processes with less or less complex steps are advantageous. This is also related to the cost of materials. Safety concerns relating to the materials used for the design . Usage of hazardous materials will increase the costs of the manufacturing process as extra safety measures are required to carry out the production. Environmental impact of the process is also to be taken into account when analysing the process. This includes aspects such as environmental contamination, but also water usage, which is a major aspect of the environmental impact of chip design. Trade-offs can be made between different materials and process steps to optimise the manufacturing process.

3.2.6. Safety

Lastly, other non-functional requirements are some more general safety concerns that can be considered. Materials that are or that degrade to be harmful to life near the sensing environment should be avoided.

4

Design of the Sensor Structure and Layout

Using the literature study results and the established requirements, the design of the device can be started. This chapter will outline the choices regarding the device structure, electrode configuration and the layout of the devices on the die.

4.1. design approach

To allow comparison and characterisation of the different device dimensions, a mask will be designed with the patterns for different devices. This chapter will focus on the design of the different types of devices that will be included.

4.2. Sensor structure

This works investigates the working of a PASiC based capacitive sensor. The working mechanic of this design is that the porous amorphous silicon carbide layer. The capacitance of the interdigital capacitance changes as ammonia diffuses into the pores.

Several different configurations are possible for this type of sensor, but as the device is manufactured using planar technologies, the most common configuration is a planar capacitance in proximity of a layer of Silicon Carbide.

Ideally, the electrodes would be positioned under the layer of SiC. This would protect them against the ammonia and the etchants, which would allow a large selection of materials to be used. As a starting point, however, the electrodes will be placed on top of the SiC. The wafers starting wafers already had a layer of SiC. For this reason, the electrodes will consist of a layer of gold on top of a layer of chromium. Both materials have excellent conductivity, and a low etch rate in HF. The SiC can then be made porous after deposition of the electrodes.

4.3. Sensor electrode layout

Following the design choices of the sensor structure, an electrode layout can be designed. This layout should enable optimal sensing performance in the measurement setup according to the requirements listed in 3.

4.3.1. Sensing Capacitor Design

Planar capacitances for use in sensing applications typically have a fairly simple layout. The interdigital electrodes lie parallel to each other, and are connected to a positive and a negative base electrode. These base electrodes are connected to a pad or on chip connection for measuring electronics. One common layout of such a capacitor is as seen in 4.1

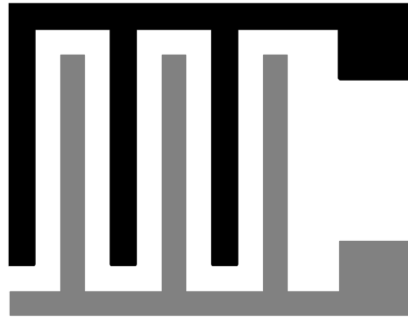
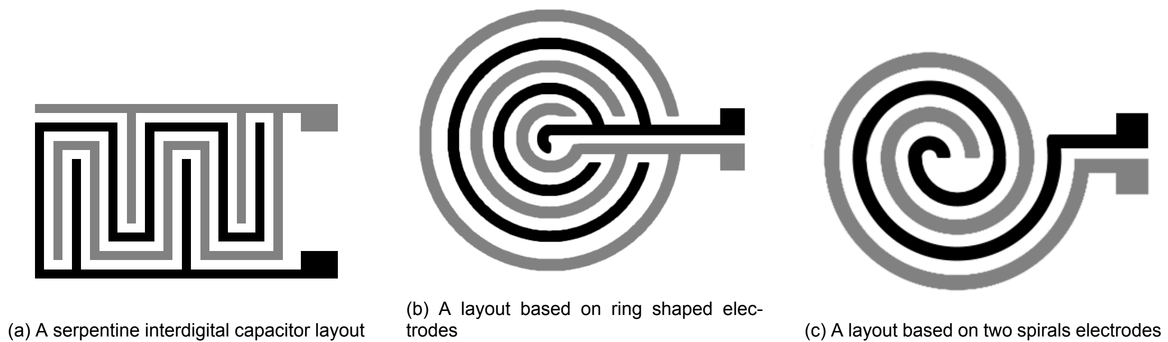


Figure 4.1: Geometry a simple interdigital capacitor

However, other configurations are possible. Rivadeneyra et al. show that by using a serpentine layout, an increased capacitance per area is obtained. [24] Figure 4.2a shows an example of this type of layout. Another configuration uses a circular layout, where the electrodes are layed out as annular rings or as a spiral, as shown in Figure 4.2b and Figure 4.2c. These devices have radial symmetry, which can be beneficial in applications where the angle of the sensor relative to the material under test influences the measurements. [25]



(a) A serpentine interdigital capacitor layout

(b) A layout based on ring shaped electrodes

(c) A layout based on two spirals electrodes

Figure 4.2: Various interdigital capacitor layouts

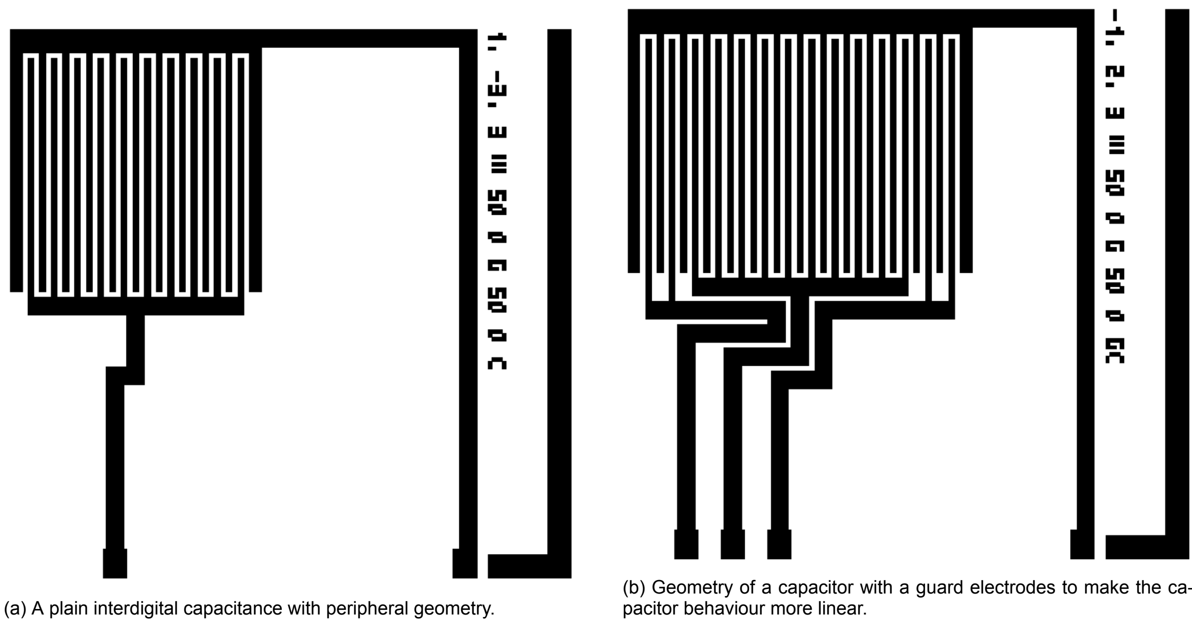
Although these are interesting benefits, these types of capacitors are less studied than the simpler type of interdigital capacitor. Equations regarding their expected capacitance are readily available. This information can be used to used to make appropriate simulations more easily. Furthermore, it becomes possible to use this established knowledge to relate measurement results to various parameters of the process, which can be fed back into the simulation to improve its accuracy.

For this reason, it was chosen to design the sensing capacitor as a simple interdigital capacitor, as shown in Figure 4.1.

guard electrodes At the edges of an interdigital planar capacitance, the field lines are not less constrained by following electrodes. This results in so called fringing field effects. While these effects increase the total capacitance of the device, they make the device more sensitive to disturbances in the fringing fields. Equations relating the capacitance to its environment in the fringing field are also considerably more complex than for the field near electrodes that are not towards the sensors edge.

The effect of fringing fields can be reduced by increasing the size of the capacitor. However, it is also possible to add guard electrodes. These are a set of electrodes which are kept at the same voltage level as the sensing electrode, but through which the current is not measured. A comparison of the capacitor layout with and without guard electrodes can be seen in Figure 4.3.

The sensing electrodes besides the guard electrode will then behave similar to the other electrodes. This way, the usage of guard electrodes therefore allows a reduction of the fringing effects using less space on the device.



(a) A plain interdigital capacitance with peripheral geometry.

(b) Geometry of a capacitor with a guard electrodes to make the capacitor behaviour more linear.

Figure 4.3: Illustration of the different modelling methods

Eliminating the effect of fringing fields simplifies calculations involving the capacitor greatly, and therefore makes it possible to infer information from the measurements and simulations. This can be especially useful when investigating to the permittivity of the gas layer and the PASiC layer using the measurement data.

The guard electrodes also reduce the capacitance between the connecting geometry of the ground electrode and the sensing electrode.

Meanders As an experiment, a geometry has also been designed that includes a meander in the ground electrode, as seen in Figure 4.4.

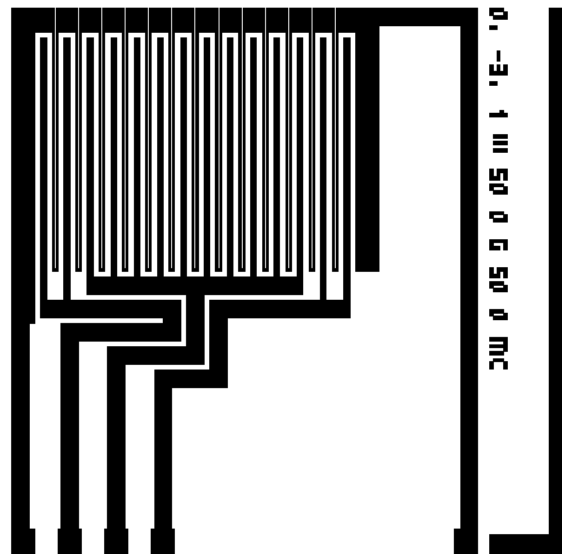


Figure 4.4: Geometry of a capacitor with a resistive heater in the ground.

While this meander does add considerable resistance to the ground electrode, it allows using the electrode as a heater by applying a voltage across it. The hypothesis is that when the electrode is used as a heater the gas will diffuse out of the sensor more quickly resulting in a lower recovery time of the

sensor. Note that these experiments are not part of the scope of this paper. They have been designed to be included in the same mask for future work.

4.3.2. Die design

The entire device includes both the sensing interdigital electrodes, and connecting geometry. This connecting geometry connects the capacitor to pads on the chip, from which connections can be made to external measurement circuitry. The layout of the connecting geometry depends on the position of the capacitors on the chips.

The size of the chips for this project depends not only on the geometry of the devices. To limit the current and amount of acid needed in the electrochemical etching current, the chips will be etched separately rather than etching the entire wafer. For this reason the wafer will be cut in chips of one square centimetre. This means that the current applied to the chip will conveniently correspond directly to the current density in Ampere per square centimetre.

On each of these chips, multiple capacitors can be placed on a single chip. It is known from previous work that when performing the electrochemical etching on the wafers there is a gradient of the porosity where areas near the edge of the die are less porous than the centre. As such, the capacitors were placed as close to the centre as possible.

To further reduce the variation in porosity between the capacitors they were placed with radial symmetry. Each die contains four capacitors, each in the corner of its quadrant as close to the centre as possible. This has the added advantage that the pads can be in the same position at the edge of the die regardless of the capacitor size, which means that a similar measurement setup can be used for all the capacitors. It also allows doing automated measurements of the capacitors before dicing, which can be useful to determine a reference capacitance of the devices and to test for defects.

This results in the placement of the capacitors on a die as seen in Figure 4.5

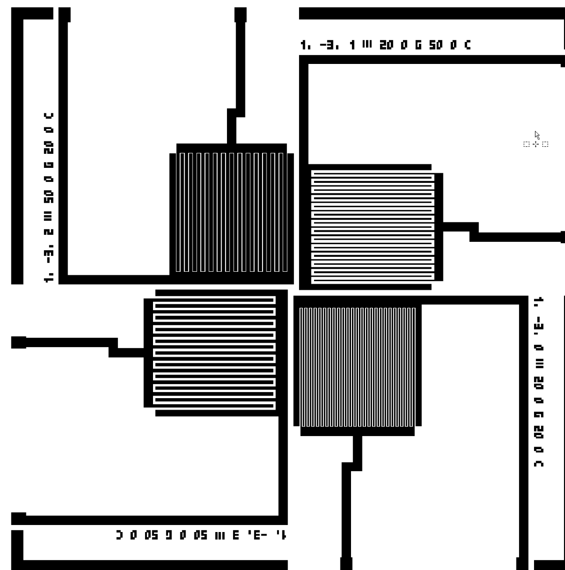


Figure 4.5: Geometry of a die with four capacitors with different parameters for the electrode geometry.

Because the purpose of the measurements is to determine the change of only the sensing capacitor, extra capacitances from the connecting geometry should be kept as small as possible. Additionally the resistance of the structures should be kept low to prevent power losses and filtering effects affecting the measurement. For this reason, the traces that lead to the pads are placed as far apart as possible. The lines should also not be too long to keep their resistance low. At the same time, a trade-off is made between the length of the electrodes and their width. The wider the lines are, the lower their resistance but the higher their capacitive coupling with other lines. For this reason, the pad for the driving electrode is placed on the opposite side of the die's edge as the sensing and guard electrodes. This allows the lines to be wider, reducing their resistance. The distance between the sensing electrode and the guard electrodes is less important, as they will both be connected to (virtual) ground during the measurements.

4.4. conclusion

The chosen device configuration of a planar capacitor on top of a PASiC layer is simple, but good enough to allow insights into the working of the device. By using gold electrodes, the silicon carbide layer can be etched with the electrodes deposited on the surface. The interdigital capacitors will be placed at the centre of the die where the porosity is expected to be even, and electrodes connecting the capacitor to pads at the edge of the chip have been separated while keeping their resistance low by choosing for wider tracks. By using guard electrodes, a better comparison of device measurements with the simulations will be possible, so that useful data about their working can be obtained.

5 Simulation of the Sensor Structure

The first manufactured wafer will contain a number of devices with different geometries. This will allow choosing geometries based on physical measurements. However, it is still useful to carry out a number of simulations in advance. Interesting simulations mainly involve simulating the magnitude of the capacitance and its sensitivity to ammonia. The purpose of these simulations is three-fold:

Firstly, the simulations serve as an estimate of the magnitude of the final capacitance as well as their sensitivity to ammonia. As such they can be used to determine a good geometry to use as a starting point when designing the mask for manufacturing the first devices.

Secondly, the simulations serve as a reference point. Comparative analysis between the manufactured devices and the results of the theoretical simulations can be used to gain insight into both the effectiveness of the devices as well as the accuracy of the model.

Lastly, the simulations will provide tools for future work. The scripts and applications developed are all made to use variable input parameters wherever possible. This allows for iterative design, as information resulting from measurements can be used as input in the tools when creating a new design.

At a later stage further simulations pertaining to reliability may also be relevant.

5.1. Simulation approach

Various simulations are carried out

5.1.1. Approach in modelling the multi-layered sensor structure

In order to simulate the capacitive properties of the sensor, an accurate but not overly complex model is to be made of the geometrical structure. The model can be divided into the following different parts:

- The gas layer above the wafer structure
- The interdigital electrode array, consisting of only the parallel strips of alternating voltages
- Further connecting geometry of the electrodes that connects the interdigital electrodes to pads where voltage is supplied.
- The layer of sensitive PASiC near the electrodes.
- The wafer bulk.

When modelling this structure, a few observations can be made to choose simulations that reduce the complexity of the simulations while retaining as much useful information as possible.

Firstly, it can be observed that the change in permittivity of the gas layer above the structure is expected to be orders of magnitude smaller than the change in permittivity of the PASiC layer. The permittivity of ammonia gas and that of air lie very close together. The change in the capacitance has been reported to be much greater with the porous layer. It is also the effect of this layer that is of particular interest. For these reasons it stands to reason to model the gas layer as a layer of air with no change in permittivity.

Secondly, the sensor is to be designed so that the interdigital electrode array is much more sensitive to changes of the PASiC layer than the connecting geometry to allow accurate measurements. In order to design the geometry of this part of the electrodes it is sufficient to simulate the interdigitated electrodes without including the connecting geometry in the model. This allows for a number of extra simplifications, as will be discussed in the next section.

5.1.2. Cross-sectional simulations

Simulating only the interdigital electrode array allows making a few useful simplifications to speed up simulations.

If the fingers are much longer than they are wide, the influence of fringing effects near the connecting geometry becomes small. It then suffices to model just a cross section of the finger electrodes near the centre of their length. Such a cross-sectional simulation is considerably faster than a 3D simulation. For the total capacitance one can multiply the resulting capacitance per unit length by the total finger length.

Furthermore, if the number of electrodes is sufficiently large the influence of fringing effects on the electrodes near the centre will be negligible. Then the geometry can be seen as a periodical structure with repeating electrodes alternating between two polarities. This periodical structure can then be modelled by simulating only the capacitance between between n electrode pairs and setting the boundary conditions such that the boundary at the left is equal to that of the right. The capacitance for all N electrode pairs of the sensing area can be calculated by multiplying the capacitance resulting from the simulation with n electrode pairs by N/n.

The resulting model geometry then becomes as is displayed in Figure 5.1.

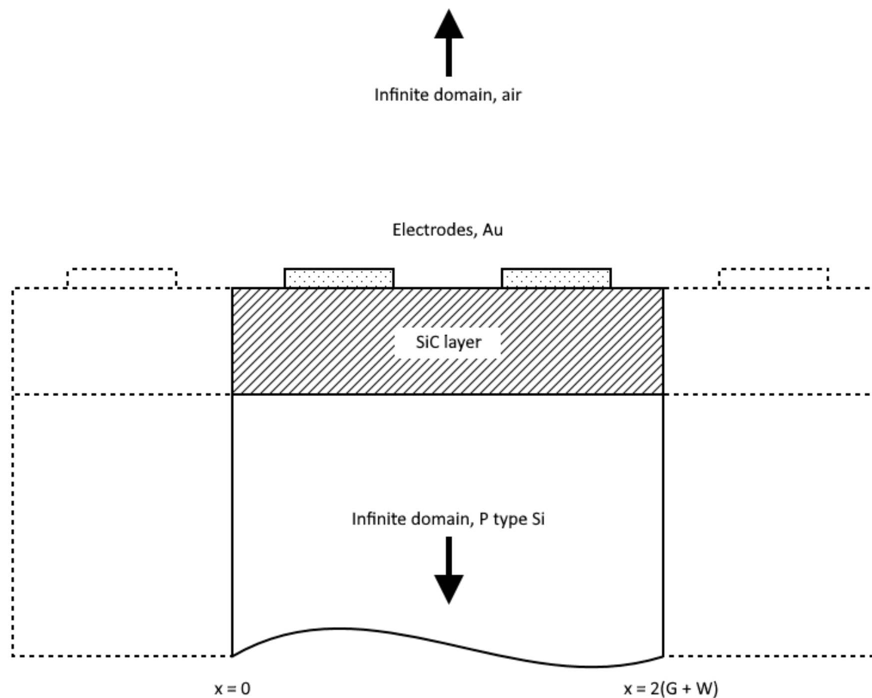


Figure 5.1: Example cross section of the periodic simulation domain with two electrodes. To model the periodicity, the boundary conditions should be set such that $E(x) = E\,(x + 2(G + W))$

5.1.3. Modelling of the PASiC layer

When modelling the PASiC layer, the sensitivity of this layer to NH3 gas has to be taken into account, which requires some extra steps.

As discussed previously in **??**, it has been observed in previous work that the capacitance seen by the interdigital electrode structure increases if the concentration of certain gasses in the environment

increases. This effect was reported to occur if the silicon carbide layer is made porous. It can then be concluded the physical processes that causes the increase in capacitance occur primarily in this layer. The presence of the gas in the pores increases the charge holding capabilities of the material by some mechanism. This can be modelled by increasing the permittivity of the material.

The sensitivity of the material here is thus described as the change in permittivity following a change in environmental ammonia concentration and should not be confused with the sensitivity of the change in capacitance resulting from the change in permittivity.

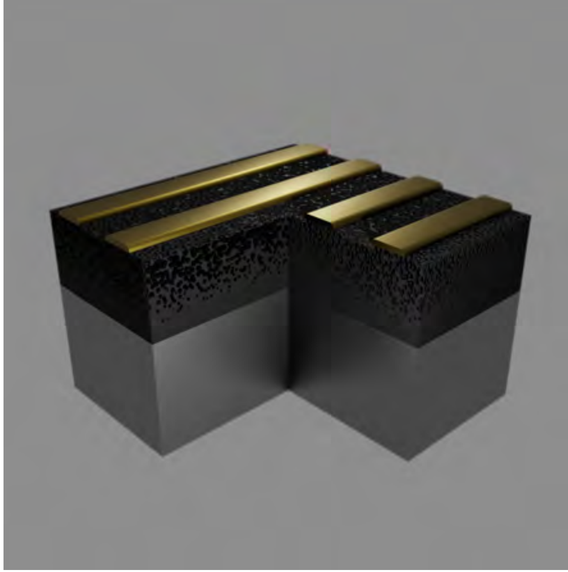
Another attribute of the porous layer that can be taken into account is that the porosity and therefore sensitivity of the material changes with the depth from the surface. This is because the etching process starts at the surface, and deeper material can only be etched after material close to the surface has been etched. Material near the surface will have a longer effective etching time and as such a greater porosity, so that the porosity of the SiC layer will decrease with the depth. Since the porosity influences the sensitivity of the material, this sensitivity then also changes with depth. Diffusion of ammonia into and out of the pores also happens more rapidly at the surface, but this mainly influences the time of the response and not the magnitude. Therefore, two methods have been used to model the changes in permittivity based on this porosity distribution.

geometric model The first method involves geometrically modelling the pores, as illustrated in Figure 5.2a. This is done by randomly subtracting circular or spherical sections from an a-SiC layer. The depth and size of the placed pores is chosen at random with a weighted probability according to a certain porosity distribution. To model the sensitivity, the permittivity of the gas inside the pores would be changed. The permittivity of the gas layer above the pores remains close to that of air. This method provides good insights into the behaviour of the field lines around the pores. However, while it can therefore give an accurate representation of the situation when no NH3 is present, it is not entirely accurate for modelling the sensitivity of the material. This is because the change in permittivity does not follow from the change in permittivity of the gas in the pores alone, but rather from a secondary physical process within the pores, although what specific process this is is not yet fully understood. Therefore, modelling the change of the permittivity as just a change of permittivity of the material in the pores may not be accurate.

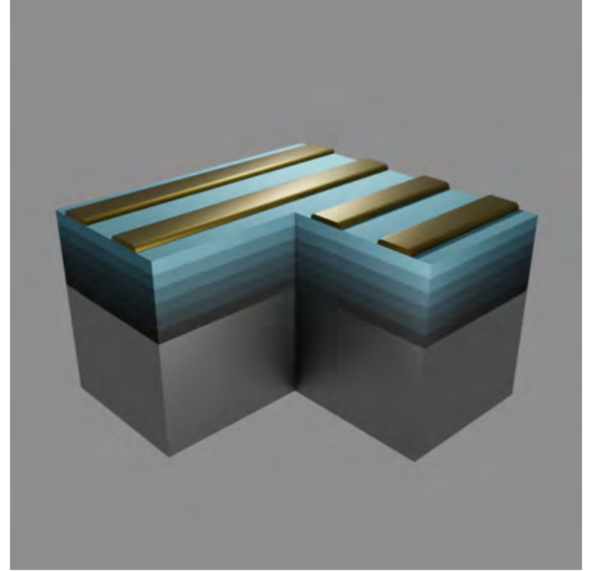
Furthermore, this model gives limited control in modelling the relation of the sensitivity with depth. With this model the porosity will change with the depth, because the most pores are placed near the surface, and therefore this area will have the greatest change in permittivity. This may be inaccurate as beyond a certain porosity the sensitivity should decrease again. In order to model this, the change in permittivity of the material in the pores has to decrease with the distance from the surface or with the pore size. Neither of these options is easily achieved as the pores overlap.

The model is also fairly complex in terms of geometry, which will increase simulation times and memory usage. Additionally, because the placement of the pores is probabilistic, there will be variation in the simulation results which may be difficult or computationally costly to quantise.

layered model In the second method the PASiC layer is divided into a number of discrete layers, each with a different permittivity based on the porosity at the depth of the layer according to the porosity distribution. This is illustrated in Figure 5.2b. The sensitivity of the different layers can then be modelled as an increase of the permittivity based on the depth of the layer. This simplification of the situation has a number advantages with respect to the physical model. Firstly, since the base permittivity and the permittivity under the presence of environmental NH_3 can be set for each layer, it is easy to control these parameters. Because of the simplicity of the geometry and because the simulation is deterministic, the simulations can also be considerably faster.



(a) modelling the pores by removing geometry. Areas with more pores will have a higher sensitivity.



(b) modelling the PASiC layer by dividing it in layers. Brighter layers near the surface are modelled with a higher sensitivity.

Figure 5.2: Illustration of the different modelling methods

5.1.4. Modelling of the porosity distribution of the PASiC layer

Modelling the distribution of the porosity and sensitivity of the PASiC with depth poses its own challenges. Little information is present regarding the porosity of the SiC layer formed following from certain processing steps, especially how this changes with the depth of the layer. Furthermore, quantitative information on how the change in relative permittivity of the porous silicon carbide changes depending on gas concentration in the pores is also unknown.

It is therefore not possible to make quantitative simulations of the PASiC layer. It is, however, still possible to get qualitative insights by making a observations regarding these properties. Therefor, a model is designed and described in this subsection that takes into account the different properties of the PASiC layer.

The model is based on a few relations between the different parameters relevant to the material sensitivity. Firstly, the porosity is related to the depth as a result from the etching process. This porosity is then related to the permittivity to the material, as the permittivity of the material will approach that of air the more porous it is. Lastly, the change in permittivity depends on the porosity. These three relations are combined to model the capacity of the material under different Ammonia concentrations.

Porosity with depth Because the layers are made porous by electrochemical etching of the SiC which starts from the surface, material near the surface will be exposed for a longer duration. One therefore expects the porosity to decrease monotonically with depth. Furthermore, depending on the etching time the porosity will reach only to a certain depth below the surface. This leaves a certain a-SiC buffer layer between the PASiC and the substrate. As a starting point an equation with linear falloff can be used, such as

$$\phi(d) = \begin{cases} 0 & d < 0 \\ \phi_{max}\left(1 - \frac{d}{d_{\text{buffer}}}\right) & d_{\text{buffer}} \leq d \leq d_{\text{SiC}} \\ 0 & d > d_{\text{buffer}} \end{cases}$$

Where $\phi(d)$ is the porosity at depth d and ϕ_{max} is the maximum porosity at the SiC surface at $d = 0$. Alternatively, a negative-exponential falloff can be used, described by

$$\phi(d) = \phi_{\text{max}} e^{-dc_{\text{falloff}}}$$

or, taking into account the a-SiC buffer

$$\phi(d) = \begin{cases} 0 & d < d_{\text{buffer}} \\ \phi_{\text{max}} e^{-dc_{\text{falloff}}} & d_{\text{buffer}} \leq d \leq d_{\text{SiC}} \\ 0 & d > d_{\text{SiC}} \end{cases}$$

Here $c_{falloff}$ is a constant controlling how fast the porosity decreases with depth.

Permittivity with porosity Even if no NH3 is present, the permittivity of the material will also depend on the porosity. If the material is not porous at all it will have the permittivity of the unetched SiC, whereas if the material is entirely porous it will have the permittivity of air. To model this, a linear interpolation between the permittivity is used. In actuality the permittivity will have a more complex relationship. Models exist for porous silicon, which show varying types of falloff with porosity. Sarafis and Nassiopoulou compare various models which include a steeper falloff with porosity, being closer to that of air than to that of silicon at moderate porosities. [26] However, such models are not available for PASiC. Therefore, using a linear equation here makes it easier to gain qualitative insights into the effect of the change in permittivity following from the porosity as its influence will be greater. The linear interpolation can be written as :

$$\epsilon_{\text{SiC, air}}(\phi) = (1-\phi)\epsilon_{\text{a-SiC}} + \phi \cdot \epsilon_{\text{air}}$$

Sensitivity with porosity When ammonia is present in the environment, the permittivity of the material changes. This contribution following from the sensitivity is linearly to the equation. The resulting relation between the permittivity and the porosity can be written as:

$$\epsilon_{\text{SiC}}(\phi, [\text{NH}_3]) = (1-\phi)\epsilon_{\text{a-SiC}} + \phi \cdot \epsilon_{\text{air}} + \epsilon_{\text{NH3}}(\phi, [\text{NH}_3])$$

Though the contribution is linearly added, the factor does depend on porosity. The material will have nearly no sensitivity if it is not porous or if it is entirely porous. Somewhere in between there will be at least one peak in sensitivity where the permittivity is maximal. The most simple equation that satisfies this condition is:

$$s(\phi) = \begin{cases} \frac{\phi}{\phi_{\text{smax}}} s_{\text{max}} & 0 \leq \phi < \phi_{\text{smax}} \\ \frac{1-\phi}{1-\phi_{\text{smax}}} s_{\text{max}} & \phi_{\text{smax}} \leq \phi \leq 1 \end{cases}$$

Where $s(\phi)$ is the sensitivity at porosity ϕ, ϕ_{smax} is the porosity at which the sensitivity is maximal and $s_{\text{max}} = s(\phi_{\text{smax}})$ is the maximum sensitivity.

This sensitivity is related to the permittivity by noting that there is no change in permittivity if no NH3 is present and the change will be maximum at an NH3 concentration of 100%. The simplest way to model this relationship would be to take a linear increase as:

$$\epsilon_{\text{NH3}}(\phi, [\text{NH}_3]) = \frac{[\text{NH}_3]}{100} s(\phi)$$

$[\text{NH}_3]$ is the ammonia gas concentration. Although the relationship between the NH3 concentration and the change in permittivity will likely follow some saturation relationship, this function is practically evaluated at $[\text{NH}_3] = 0$ and $[\text{NH}_3] = 100$ so that the relationship is less important. This linear equation has the added advantage that the slope of the change in permittivity directly relates to the slope of the concentration.

Permittivity with depth The equations are combined to get the permittivity at a certain depth as :

$$\epsilon(d) = \epsilon(\phi(d), [\text{NH}_3])$$

Figure 5.3 displays an example of the porosity distribution resulting from the model and Figure 5.4 shows the permittivity with depth. The parameters used are $d_{\text{SiC}} = 0.5\,\mu\text{m}$, $d_{\text{buffer}} = 0.1\,\mu\text{m}$,$\phi_{\text{max}} = 0.5$, $phi_{\text{smax}} = 0.5$ and $s_{\text{max}} = 0.2$.

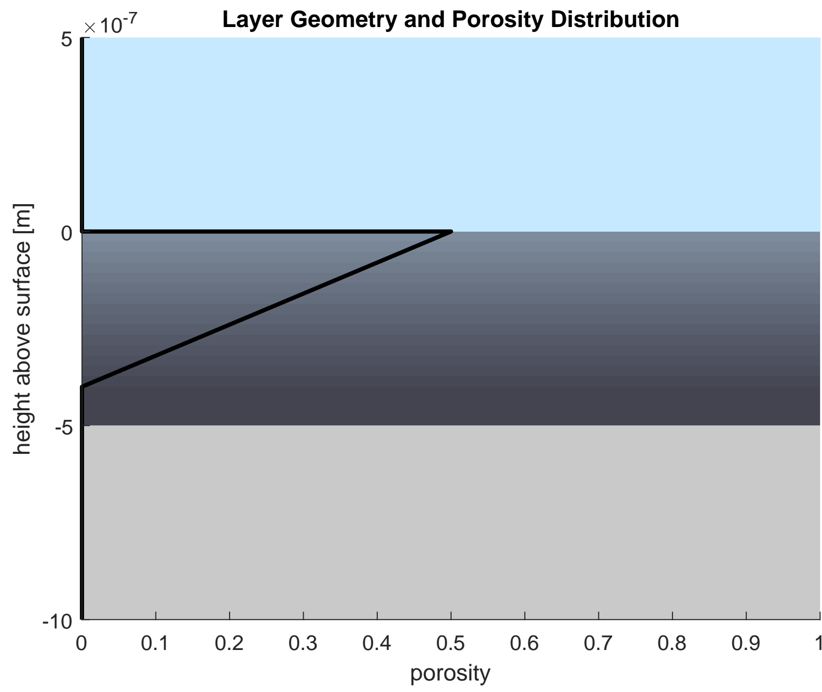


Figure 5.3: Distribution of porosity with depth within the SiC layer following from the porosity model.

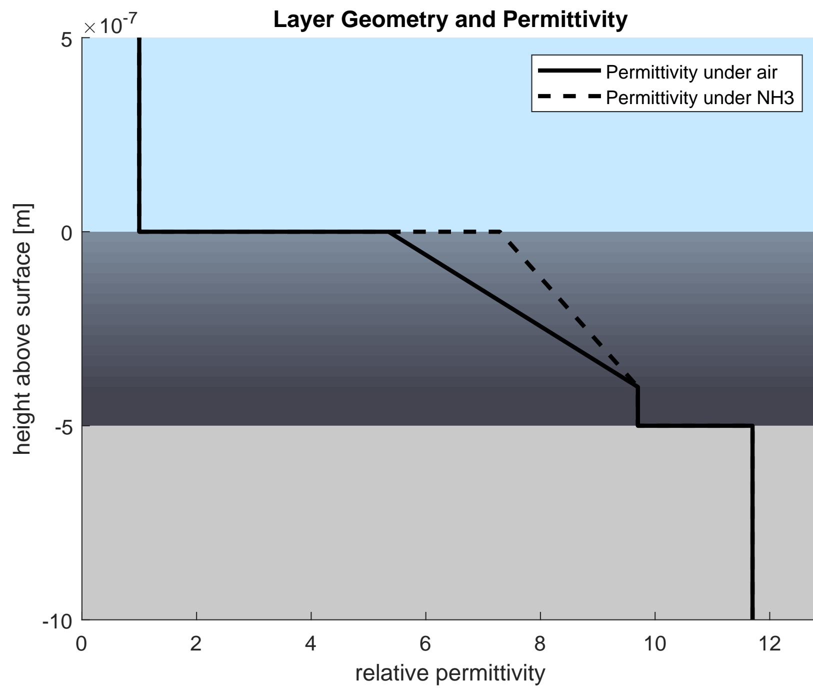


Figure 5.4: Permittivity distribution with depth with and without the presence of NH3 gas following from the porosity model.

The figures were generated using a function written in MATLAB which can be found in Appendix A.1.2. One can see how with this model the permittivity can vary with depth. The porous material at the surface has a lower permittivity. At the same sensitivity is maximal at the surface where $\phi_{\max} = \phi_{\text{smax}} = 0.5$, which can be seen as the increase in permittivity here is the greatest.

It should be noted that while these partly arbitrary equations have been chosen as a starting point, the simulations are set up such that if information on the PASiC is obtained, this information can be used in future simulations. The simpler equations should still suffice to give an estimate of qualitative properties of the electrode configuration as long as they are properly interpreted.

The wafer will contain a number of PASiC structures with different dimensions. Because the porosity under these structures will be made by the same process, this means that the wafer will contain different electrode dimensions over fairly similar porous layers. Furthermore, because the electrode structures with different dimensions will have a different sensitivity to the permittivity at different depths, it may be possible to estimate the permittivity at different depths when the PASiC is exposed to different concentrations of ammonia gas. This could then replace this model so that more accurate simulations could be performed. However, this is beyond the scope of this project and as such are proposed as a subject for future work.

5.1.5. cross-sectional simulation in COMSOL

Using the sensitivity model, it is possible to set up the 2D simulations. Using COMSOL software package, various finite element simulations were carried out.

geometric model Some preliminary experiments were done with modelling the pores geometrically. A COMSOL application was written which would procedurally subtract circular pores from the side view, the code for which can be found in Appendix A.2.1. This results in a geometry as seen in Figure 5.5.

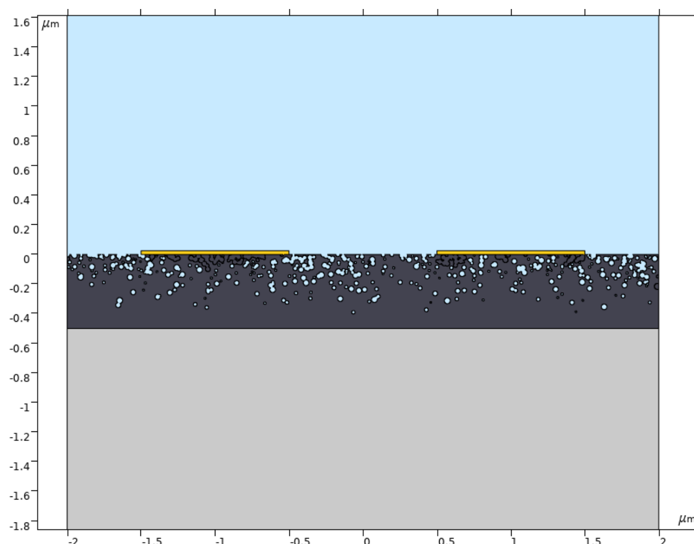


Figure 5.5: Geometry of a COMSOL model with geometrical pores.

These simulations could be used for some insights into how the field lines could behave around the pores. However, they provide limited information. Firstly, because this is a 2D simulation it's equivalent to cylindrical pores when extruded rather than the cellular pores structure that is desired. Furthermore, it is not easily possible to control the change in permittivity with depth, as the pores overlap.

layered model For the majority of the capacitance estimations a 2D simulation in COMSOL with the layered model was used. A model was made with parametric geometry which generates the geometry for a certain number of layers. A method would then assign to each of these layers a material with the permittivity according to an implementation of the model derived in 5.1.4. The code for the corresponding methods are found in Appendix **??**. The geometry of the model then becomes as seen in Figure 5.6.

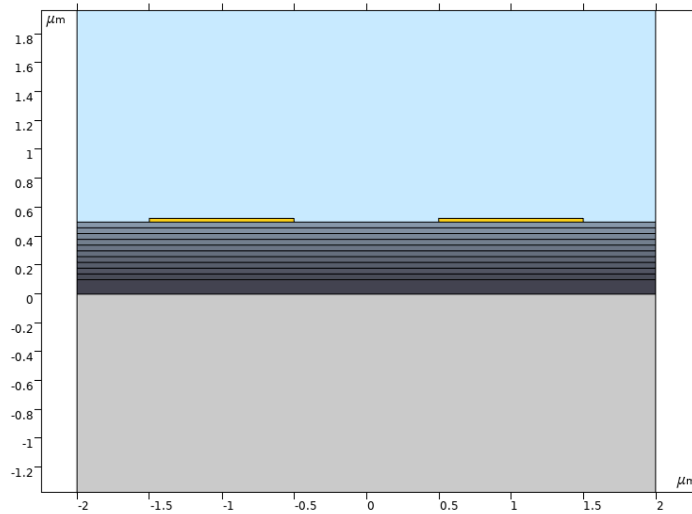


Figure 5.6: Geometry of a COMSOL model with the PASiC modelled as layers with different permittivities.

The different coloured layers below the electrodes are the various SiC layers with different sensitivities. The layers have their permittivity property set to some form of:

$$\epsilon_{\text{base}} + [\text{NH}_3] * \epsilon_{\text{base}}$$

Where ϵ_{base} and $\epsilon_{\text{increase}}$ are constants given to the material by the script according to the model, and $[\text{NH}_3]$ is a parameter that is included in the simulation to control the effect of the sensitivity.

Because the design is parametric, a sweep can be ran to calculate the capacitance for a range of different electrode dimensions and ammonia concentrations. To compare the results of the different simulations, the capacitance per unit area has to be used since the simulations will have different widths. The capacitance follows from the capacitance determined in the 2D simulation. The width of a simulation using n electrode pairs will be:

$$2 * n * (G + W)$$

For convenience, area per cm^2 is calculated. Since G and W are specified in COMSOL in μm, the length in centimetres can be written as:

$$2 * n * (G + W) * 10^{-6}/10^{-2} = 2 * n * (G + W) * 10^{-4}$$

The capacitance per unit length for a centimetre wide strip would therefore become:

$$\frac{C_{sim}}{2 * n * (G + W) * 10^{-4}}$$

This can be multiplied by the depth to get the capacitance per area as:

$$C_{\square} = \frac{10^{-2} \cdot C_{sim}}{2 * n * (G + W) * 10^{-4}} = \frac{C}{2 * n * (G + W)} * 10^{2}\text{F cm}^{-2}$$

One relevant detail in these simulations is that the permittivity increase also occurs underneath the electrodes. In reality, the electrodes will protect the material underneath them from being etched, which will influence the sensitivity. If the devices are eventually fabricated with the electrodes under the SiC, this problem becomes irrelevant.

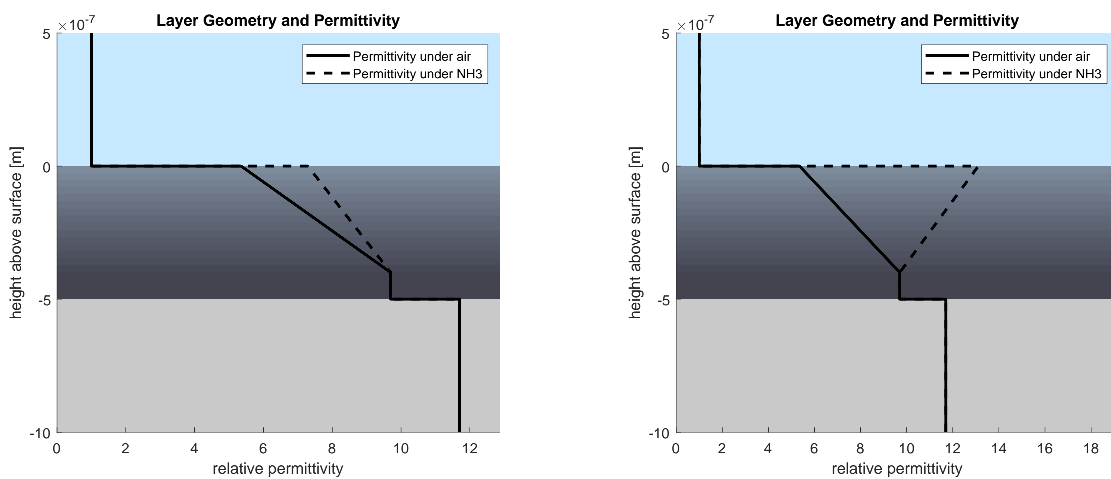
5.1.6. cross-sectional analytical model using MATLAB

Although the cross-sectional simulations in COMSOL are fairly accurate, they take a considerable time to be performed. as such, iterating over a large number of combinations of W and G will be time-consuming.

Igreja and Dias present an analytical method to estimate the capacitance of a multilayered structure, which estimates the capacitance of the multilayered structure by calculating the contribution of the different layers of the capacitor using conformal mapping techniques and partial capacitance. [27] An implementation of the model was written in MATLAB, of which the code is presented in Appendix A.1.6. Using the analytical model, many geometries could theoretically be calculated in a reasonable amount of time. However, this model does come with some limitations which make it's application troublesome.

limitations on layer configurations The primary limitation is that both domains must be monotonically increasing or decreasing in order for the model to be accurate. That is, the permittivity of the layers must either be lower with every layer further away from the electrodes or the permittivity must be higher with every layer.

This is problematic, as with the chosen PASiC model, the permittivity of the layers changes such that this condition isn't always met. Consider the two situations depicted in Figure 5.7, generated using the code in Appendix A.1.3.



(a) Example of the permittivity change of a material with low sensitivity of $s_{max} = 0.2$

(b) Example of the permittivity change of a material with high sensitivity of $s_{max} = 0.8$

Figure 5.7: Permitivity of different sensing situations

Figure 5.7a shows an example situation where the permittivity changes only slightly under NH3 presence. Both with and without Ammonia gas in the environment, the permittivity in the SiC decreases monotonically with depth from the surface. Because the permittivity of the Silicon is higher than that of the SiC, the condition on the lower domain holds for the entire domain and the analytical model can be applied. If the the electrodes are below the SiC, as the upper domain with the SiC is monotonically increasing. The model can then be applied as well.

However, the situation changes if the sensitivity is greater, as shown in Figure 5.7b. When NH_3 is present, the permittivity increases so much that the domain which includes the SiC layer will no longer decrease or increase monotonically regardless of the electrode position. It would only be possible to apply the analytical model to the situation without ammonia present. This is troublesome, as it will later be shown that the increase in permittivity has to be large in order for the capacitance to be significantly influenced.

Because of these limits in the application of the model, complete calculation of the sensitivity according to the sensitivity model of 5.1.4 is not possible. For the situation with the electrodes on top, under NH_3, and with a large sensitivity, it may be possible to clamp the permittivity of the SiC layers such that it is always at least as great as that of the Si layer. Combined with the otherwise monotonously decreasing domain, this would allow calculating a comparison to the situation without Ammonia presence. If the sensitivity is very large, the slight increase in permittivity for the porous material has less influence. If such an approach is chosen, comparison to the COMSOL models is limited unless the same alterations are made.

precision limitations During the application of the model, a second limitation of the implementation became apparent. The model expressed the electrode dimensions using the wavelength $\lambda = 2 * (W + G)$, the metallisation ratio $\eta = W/(W+G)$. The height of the layers relative to the electrode dimensions is then expressed using the adimensional layer thickness $r = h/\lambda$, where h is the height of the top of the specific layer from the electrodes.

For each of the layers in the model, a factor is calculated based on the values of r and θ, which is then multiplied by the difference in permittivity between the layer and the adjacent layer to determine the contribution to the total capacitance. If $r << 1$, that is, the wavelength of the device is much greater than the layer height, precision errors occur in the calculation of this factor. When these precision errors occur, the factor can no longer be determined. Figure 5.8 shows how the factor changes for different values of r and η.

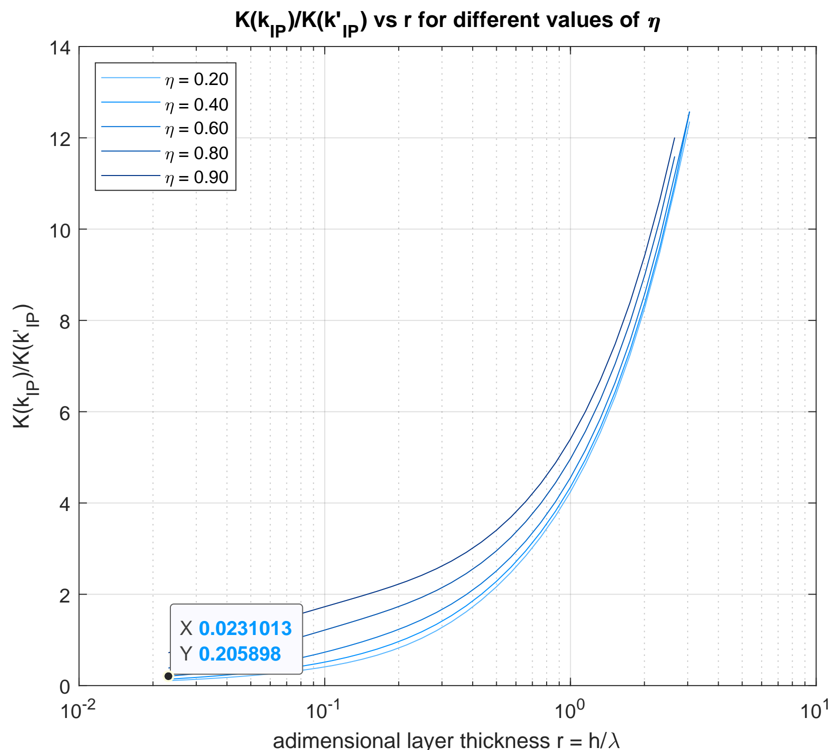


Figure 5.8: Geometry of a COMSOL model with the PASiC modelled as layers with different permittivities.

The code to generate the figure is found in A.1.7 It can be seen that the calculation of the factor becomes impossible at a certain point. The value at which calculation of the factor results in errors is $r = 0.0231$.

Because the errors occur because the layers are very thin compared to the electrodes, one might suggest to simply discard the influence of the layer, assuming that it is not significant. However, this may not be the case.

At the point where the precision errors occur, the factor depends on η, but is generally still large enough to have a notable influence relative to the other layers. Therefore discarding the layers causes significant deviations in the calculated capacitance. Moreover, the factor is multiplied by a factor based on the permittivity of the layers. Even though the layer is thin, it may have a large permittivity and should therefore still contribute significantly to the total capacitance.

Precision errors also occur if $r >> 1$. At this point, the size of the layer relative to the electrodes is large enough for the factor to become infinite where it should become large but finite. However, this issue was less relevant at the dimensions of the capacitors and layers that were investigated.

5.1.7. 3D device simulations in COMSOL

While the cross-sectional models do provide insights into the relative magnitudes of the capacitance resulting from different geometrical dimensions of the electrodes and the relative magnitudes of their sensitivities, there are a number of shortcomings that will cause significant differences between the modelled capacitance.

One main shortcoming of the cross-sectional models is that they consider only the fingers of the electrodes. As such, fringing capacitances and parasitic capacitances between the electrodes resulting from geometry outside of the finger electrodes themselves are not included. The practical capacitance will then be significantly larger. It should be noted however that parts of this capacitance may also be sensitive to changes in the gas layer as they may also be situated above the PASiC layer.

To get a more accurate estimate of the added capacitance of the connecting geometry, 3D simulations can be carried out. These 3D simulations can also be used in design considerations to choose geometrical features to reduce the losses in sensitivity with respect to an ideal interdigital capacitance that follow from the added parasitics.

These simulations can again be done with finite-element simulations using the COMSOL software package. The geometry of a complete die was modelled and extruded in COMSOL on top of a SiC layer. Infinite domains of air, SiC, and silicon were added on the boundaries of the simulation. This results in geometry such as that displayed in Figure 5.9

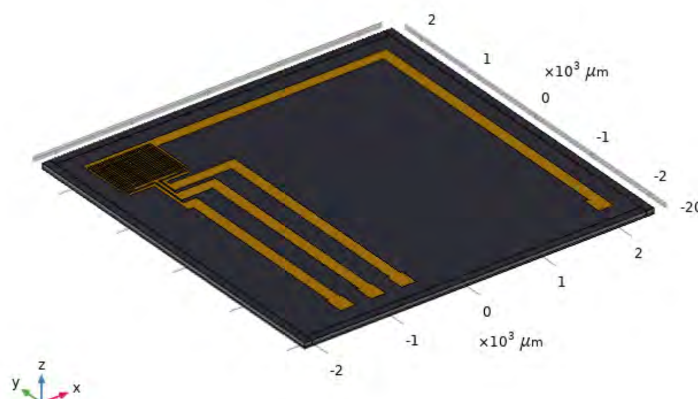


Figure 5.9: 3D geometry of a complete device for modelling in COMSOL

This geometry can then be used in electrostatic simulations to estimate the influence of the peripheral electrodes on the sensing capacitor. The domain of air above the device is rendered transparently.

5.2. Simulation results

5.2.1. Cross-sectional simulations using COMSOL

geometric model Although not extensively used, some electrostatic simulations were done using the geometric model. The result of this is seen in Figure 5.10.

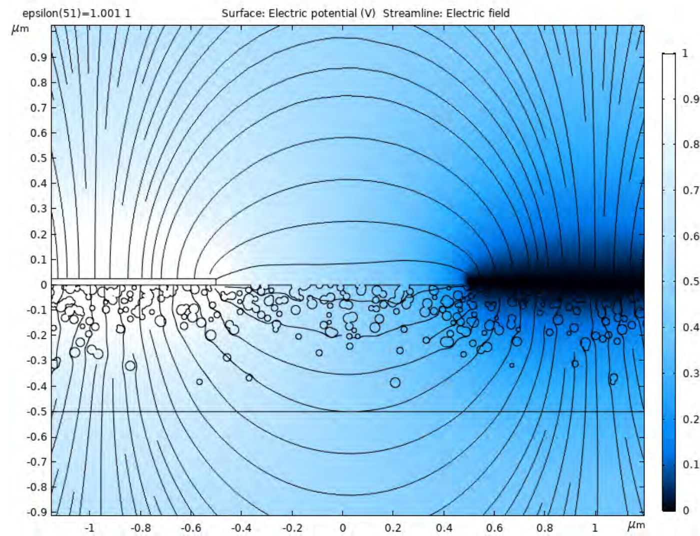


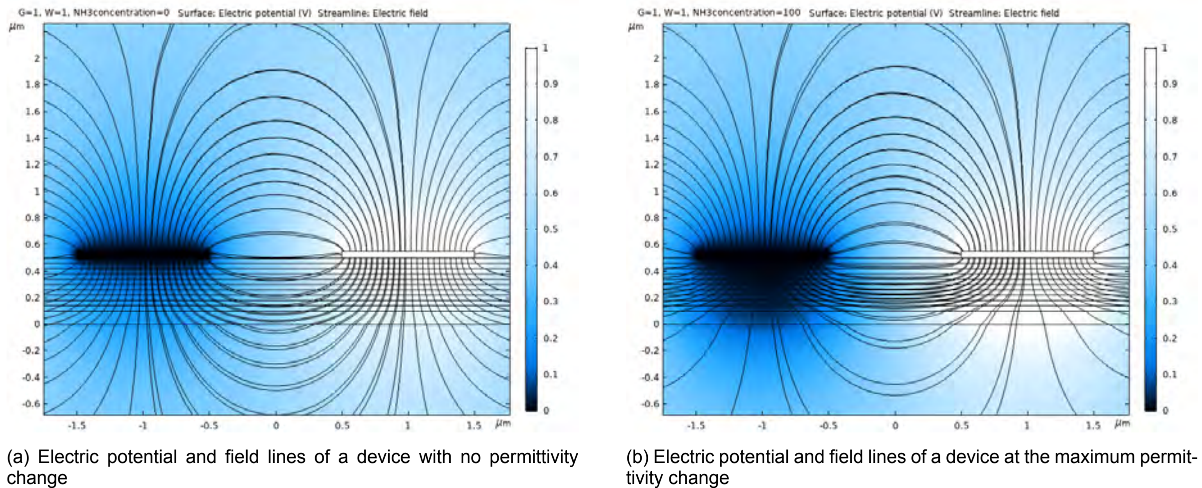
Figure 5.10: Simulation showing the electric field in a 2D simulation with geometric pores

While not much quantitative information can be inferred from the simulation, it is interesting to see how the field behaves near the pores. The field lines go predominantly around the pores. Since the pores are filled with air, they have a lower permittivity than the surrounding SiC. Therefore, the field strength will be greater around the pores than inside them.

layered model The layered model provides more useful information in designing the electrode dimensions. Here the sensitivity of the PASiC layer can be modelled by an increase of the permittivity of the layers.

At first, little seemed to happen based on a change in concentration. Even with a 10 fold increase (as the maximum sensitivity at the surface) in permittivity, little change was apparent in the capacitance. Only at larger values a change was apparent. For this reason, large increases have been used for some of the visualisations.

With this increase in permittivity, a considerable change in permittivity occurs, especially for devices with small wavelengths. The field strength becomes much greater in the PASiC layer, as is visible in Figure 5.11 which shows how the field lines change under the different circumstances.



(a) Electric potential and field lines of a device with no permittivity change

(b) Electric potential and field lines of a device at the maximum permittivity change

Figure 5.11: Results of electrostatic simulations of the layered 2D model with devices with $W = G = 10\,\mu\mathrm{m}$ under a 15 fold permittivity increase

A parameter sweep was done with these simulations to create a. Gap and electrode widths of 1, 2, 5, 10, 20, and 50 $\mu\mathrm{m}$ were modelled with a 10 fold maximum permittivity increase. The resulting capacitances were then processed in MATLAB to calculate the sensitivity of the devices, as a percentage increase in capacitance per cm^2. This results in the plot seen in Figure 5.12.

Sensitivity for Different Electrode Dimensions

Electrode Gap \ Electrode Width	1	2	5	10	20	50
1	454%	316%	190%	133%	97.2%	67.9%
2	336%	234%	143%	100%	72.9%	50.9%
5	231%	159%	96.3%	67.4%	48.7%	33.6%
10	180%	121%	72.4%	50.2%	35.9%	24.3%
20	144%	96%	55.8%	37.9%	26.7%	17.6%
50	113%	73.5%	41.2%	27.2%	18.6%	11.8%

Figure 5.12: Figure showing the simulated sensitivity for varying electrode width and distance at a ten-fold maximum permittivity increase

The MATLAB code to process the simulation data can be found in Appendix A.1.4. The plot clearly shows that, as was expected, the smaller devices have a higher. What is less straightforwards is that the sensitivity of devices with a greater electrode width is generally larger than devices with the same wavelength that have narrower electrodes. An explanation for this is that the wider electrodes create an electric field that penetrate less deep into the surface, and therefore is stronger near the sensitive surface.

The same simulations were repeated with the electrodes under the carbide rather than on top of it. The processed results of this sweep are seen in Figure 5.13.

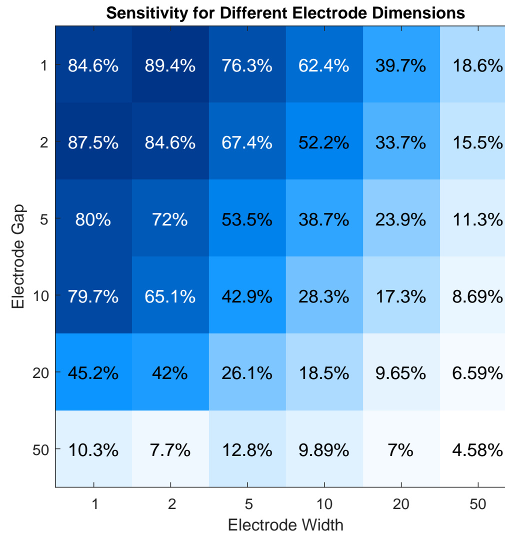


Figure 5.13: Figure showing the simulated sensitivity for varying electrode width and distance at a ten-fold maximum permittivity increase with the electrodes beneath the silicon carbide

The results of this simulation are especially interesting. The change in permittivity is considerably lower. This is to be expected, as the distance to the sensitive surface is increased. What's more notable is that the sensitivity has a maximum at a device with a wavelength greater than the smallest device included in the sweep. This suggests that smaller electrodes do not sense deep enough to reach the top layer.

Lastly, a part of the simulation was ran again with electrodes with a five times larger thickness to evaluate the effect on the sensitivities. A base thickness of $125\,\mathrm{nm}$ was used rather than $25\,\mathrm{nm}$ for electrode dimensions $W = 1\,\mu\mathrm{m}$, $G = 1\,\mu\mathrm{m}$. The resulting sensitivities are shown in Figure 5.14.

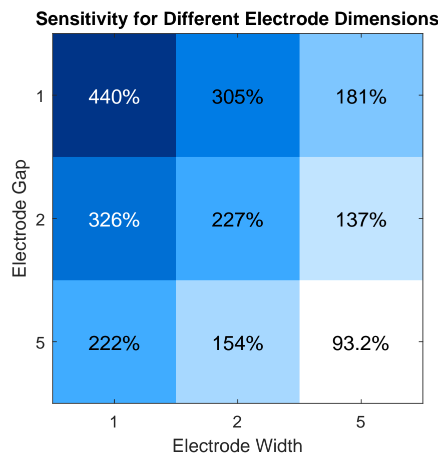


Figure 5.14: Figure showing the simulated sensitivity for varying electrode width and distance at a ten-fold maximum permittivity increase with thick electrodes

It can be seen that the sensitivity is reduced, but the effects are limited considering the increase in electrode thickness.

5.2.2. Cross-sectional Simulations Using MATLAB

Despite the previously mentioned limitations, the analytical model implemented in MATLAB has also been used. A large, 10-fold increase of permittivity at the top layer was used for comparison to the COMSOL models, which also use this increase. As a result, the . In practice this only means that the lowest buffer layer has a permittivity of $\epsilon_r = 11.7$ instead of $\epsilon_r = 9.7$. Precision errors are in this case dealt with by discarding the layers. The MATLAB code for the execution of the model is found in Appendix A.1.5, and the implementation of the analytical model is in Appendix A.1.6.

To investigate the influence the precision errors have on this result, an upper bound for the error is calculated. If r is too low for a certain layer, the contribution of the layer is calculated by using the smallest value of r where errors do not occur. This value is added to an error value. The resulting errors can be seen in Table 5.1.

Table 5.1: Analytical model error compared to capacitor value

G [μm]	W [μm]		
	1	2	5
1	57.9 nF 2.13 nF (27.2)	19.2 nF 1.15 nF (16.7)	3.21 nF 0.454 nF (7.08)
2	35.8 nF 2.01 nF (17.8)	16.5 nF 1.21 nF (13.7)	3.9 nF 0.517 nF (7.55)
5	13.2 nF 1.54 nF (8.55)	8.85 nF 1.07 nF (8.29)	3.61 nF 0.544 nF (6.64)

(a) The maximal error of the analytical model under air compared to the total capacitance (in green) as a fraction (in purple)

G [μm]	W [μm]		
	1	2	5
1	6.26 nF 70.9 nF (0.0883)	4.78 nF 24.5 nF (0.195)	3.86 nF 3.95 nF (0.977)
2	9.1 nF 45.7 nF (0.199)	6.26 nF 17.3 nF (0.362)	4.47 nF 2.9 nF (1.54)
5	17 nF 16.7 nF (1.02)	10.5 nF 6.57 nF (1.59)	6.25 nF 0.562 nF (11.1)

(b) The maximal error of the analytical model under ammonia presence compared to the total capacitance (in green) as a fraction (in purple)

The value of the errors appear to be quite large, but it should be emphasised that this error is an upper bound, the actual error will be considerably lower, especially for the larger devices. The value of r for the small layers can get as small as $r = 0.002$, but the value used to calculate the error bound is $r = 0.0231$ as this is the lowest possible value for which errors do not occur.

The error is generally smaller under ammonia presence as the permittivity of the layers is smaller, and therefore discarding the layers has a smaller impact on the capacitance.

When comparing the ratio between the error and the total capacitance, it may be noted that in the case where ammonia is present the upper bound of the error increases with the wavelength, whereas for the case under air the upper bound decreases. This has to do with the difference in configuration of the layer permittivities for the two cases. In the case of ammonia presence, the domain with the SiC is monotonically decreasing in permittivity, whereas in the case of air it is monotonically increasing. The model estimates the contribution of the layers differently in these cases, so that in the case under air where the permittivity increases away from the electrodes, the influence of the discarded layers will be smaller.

The calculated capacitance values for different electrode geometries with the small layers discarded have been displayed in Table 5.2.

Table 5.2: Analytical model capacitor values

G [μm]	W [μm]		
	1	2	5
1	2.13 nF 1.92 nF (1.11)	1.15 nF 1.07 nF (1.08)	0.454 nF 0.426 nF (1.06)
2	2.01 nF 1.75 nF (1.14)	1.21 nF 1.09 nF (1.11)	0.517 nF 0.481 nF (1.08)
5	1.54 nF 1.29 nF (1.2)	1.07 nF 0.923 nF (1.16)	0.544 nF 0.492 nF (1.11)

(a) Comparison of the capacitance under air resulting from the analytical model with the COMSOL simulations (in green) as well as the ratio between them (in purple)

G [μm]	W [μm]		
	1	2	5
1	70.9 nF 10.6 nF (6.67)	24.5 nF 4.44 nF (5.52)	3.95 nF 1.24 nF (3.2)
2	45.7 nF 7.66 nF (5.96)	17.3 nF 3.64 nF (4.76)	2.9 nF 1.17 nF (2.48)
5	16.7 nF 4.27 nF (3.91)	6.57 nF 2.39 nF (2.75)	0.562 nF 0.966 nF (0.582)

(b) Comparison of the capacitance under ammonia presence resulting from the analytical model with the COMSOL simulations (in green) as well as the ratio between them (in purple)

In all cases, the capacitance in the analytical model is larger than that of the COMSOL simulations. For the case under air, the values found by the analytical model and the simulations are relatively close. However, the capacitance found under NH3 is for small devices is much larger for the analytical model, which is somewhat surprising considering that the most sensitive layers have been discarded for the larger devices. The removal of these layers can still explain that the increase in sensitivity is lower for larger wavelengths.

Lastly, the sensitivity has been calculated for both for the analytical model and the COMSOL simulations in Table 5.3.

Table 5.3: Sensitivity of the analytical model

G [μm]	W [μm]		
	1	2	5
1	3330% 550% (6.02)	2120% 420% (5.11)	870% 290% (3)
2	2270% 440% (5.21)	1430% 330% (4.29)	560% 240% (2.31)
5	1080% 330% (3.27)	610% 260% (2.38)	100% 200% (0.526)

Table 5.4: Comparison of the sensitivity of the analytical model and the simulations (in green). The ratio between the two is also shown (in purple).

The calculated sensitivity of the analytical model is much higher than that of the COMSOL simulations. This follows from the same reasons causing the capacitance under ammonia to be high.

5.2.3. Simulation of the integrated device

3D simulations were done of an integrated device to determine the influence of the peripheral electrodes.

A very fine mesh is used for the simulation, but even still quite thick electrodes have to be used to prevent errors in the simulation. This will add to the capacitance considerably. For the following plots, an electrode thickness of $500\,\mathrm{nm}$ has been used. This is as thick as the A $500\,\mathrm{nm}$ layer of SiC. The electrode width used is $20\,\mu\mathrm{m}$ and the gap width is $10\,\mu\mathrm{m}$.

Figure 5.15 shows the electric potential across the entire device.

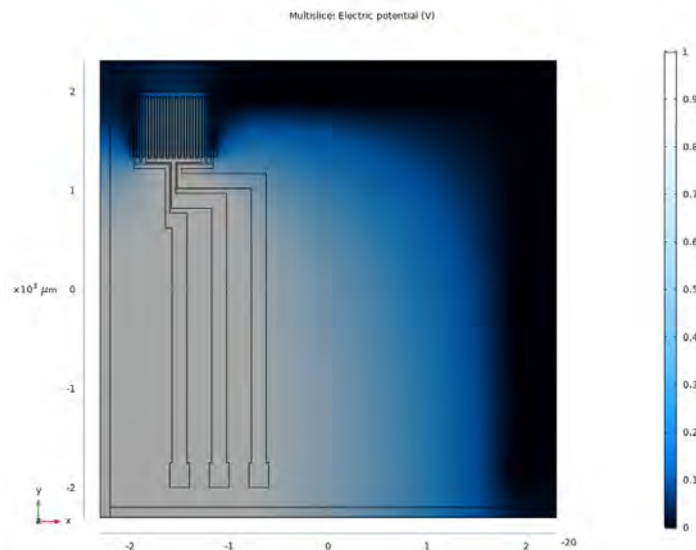


Figure 5.15: Top view of the result of an electrostatic simulation of a complete device in COMSOL

It can be seen that the field strength is the greatest near the sensing area. Because the guard electrodes are floating it is visible how the field behaves near the edge of the ground electrode.

Figure 5.16 shows a slice of the simulation near the sensing surface.

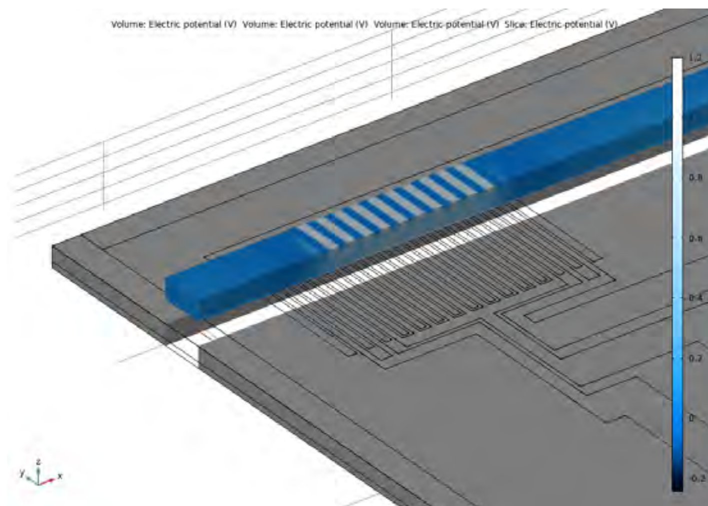


Figure 5.16: Slice of the 3D simulation near the sensing electrodes

On the cross-section of the slice, the electric potential is displayed. It is seen how the field reaches into the device, and how the electric field is much stronger near the electrodes than anywhere else as a result of the electrode layout.

The capacitance resulting from the evaluation is about $3.17\,\mathrm{pF}$. The sensing area is $600\,\mu\mathrm{m}$ by $600\,\mu\mathrm{m}$, so that the capacitance per area can be calculated as $879\,\mathrm{pF\,cm^{-2}}$. For comparison, the 2D simulation found a value of $226\,\mathrm{pF\,cm^{-2}}$. Considering the extra thickness of the capacitors and fringe effects in the simulations that contribute to a further increased capacitance, this is a reasonably small

increase.

5.3. Conclusions

From the simulations it has become clear that, as expected, the devices should be made as small as possible. Further details

It also appears that the change in permittivity of the layer has to be quite large for the change in capacitance as is seen in the literature.

3D simulations suggest that the peripheral electrodes add some capacitance to the device, but the accuracy of the simulations is limited. Working devices remain feasible with the design.

6
Design of the lithography mask

This chapter documents the choices made in the process of designing a mask containing a variety of different capacitors.

6.1. Capacitor design and variation

Because a single mask is first made to reduce costs, a number of different designs is included on the mask to allow the possibility of obtaining as much useful information out of measurements on the created structures as possible.

Guard electrodes are included wherever possible to allow for more accurate measurements For insight into the effect of the guard electrodes and to determine whether the fringing effects have influence on the sensitivity of the capacitor, there is also a series of capacitors included that do not have guard electrodes.

size variation Firstly, the capacitors have varying dimensions of the interdigital electrodes. Both the gap width and electrode width are varied. Because it followed from the simulations that the capacity and sensitivity decreases if the difference between the electrode width and gap width increases, it is expected that the sensitivity is optimal when the gap width and electrode width are equal. For this reason, more capacitors are included that have equal gap and electrode widths compared to capacitors with a difference between the two parameters. Instead, variation is primarily chosen in the number of electrodes and the total sensing area.

To do a comparison of the effect of only the dimensions of the electrodes, firstly a series of capacitors is chosen that all have the same number of electrodes. Furthermore, the sensing area and the connecting electrodes close to the sensing area also scaled with the sum of the gap width and electrode width.

This does, however, limit the size of the capacitors considerably as a capacitor with the same number of electrodes but with a larger width takes up a much greater area. Therefore, the bulk of the capacitors uses a constant size of the peripheral electrodes and the sensing area while the interdigital electrode dimensions and number varies to fit the area. This does increase the effect of the peripheral electrodes and the parasitic resistances of the geometry, but allows for easier comparison of effects of the etching parameters.

repetition To allow testing different etching parameters, multiple copies of certain capacitances are included on the wafer. This also builds in some redundancy against variations in the quality of the fabrication process across the wafer. The amount of copies of a capacitor depends on the purpose and the geometry. Extra capacitors are included for the sizes intended to be tested with different etching parameters. As mentioned earlier, more capacitors are also included with equal gap and electrode width.

6.2. Capacitor placement on the Mask

6.2.1. Automated Die Generation and Placement

In order to allow a comparison of geometries with different dimensions, a large number of different geometries is to be tested. Designing and maintaining all these different geometries by hand is cumbersome and rather error prone. For this reason it was instead chosen to write an application that manages the placement of the capacitors on the wafer.

Because the geometry had already been designed and built in COMSOL in order to use it in simulations and because COMSOL has a fairly well documented application programming interface, COMSOL was chosen as the application of choice. COMSOL can export geometry in DXF file format so that it can be imported in mask design programs such as L-Edit.

This section documents in short the working and usage of the application. For a full documentation refer to Appendix A.6.

The program is divided into four separate sections. Each of these sections includes a tab with parameters and a button to execute the related code. The result of each section is displayed in a geometry window.

The first section allows the user to describe parameters relating the wafer. This includes parameters about the wafer dimensions, but also information about margins where wafers should not be placed. With the ”Preview wafer dimensions” button the user is shown an outline of the sections of the wafer where chips will not be placed.

In the same section the user can choose parameters for the dimension of the chips to be placed, as well as the distance between them and a grid offset. When the ”Preview die placement” button is clicked, squares will be displayed on the wafer preview where the dies will be placed.

In the second section the user can enter all the values for the base capacitor dimension parameters. Because COMSOL interprets the values from the fields as expressions, one can aside from providing numeric values also enter expressions using the other variables. A preview of the capacitor can be generated by using the ”Generate capacitor preview” button.

The last section allows the user to control the actual placement. Firstly the dies on which the capacitors are to be placed is selected. One can either select a quadrant or enter a list with positions.

describe the variation for one or multiple parameters. A parameter name is entered along with parameters for the variation range or a list.

When the ”generate dies” button is pressed the program starts generating the capacitors and exporting the dies one by one, as generation of geometry in COMSOL slows down if all dies are kept within the geometry. The program will for each of the selected dies generate capacitors with the specified parameters, while varying the parameters for which ranges are specified.

6.3. Mask overview

Using the exported dies, a mask can be assembled. The resulting mask is shown in Figure 6.1

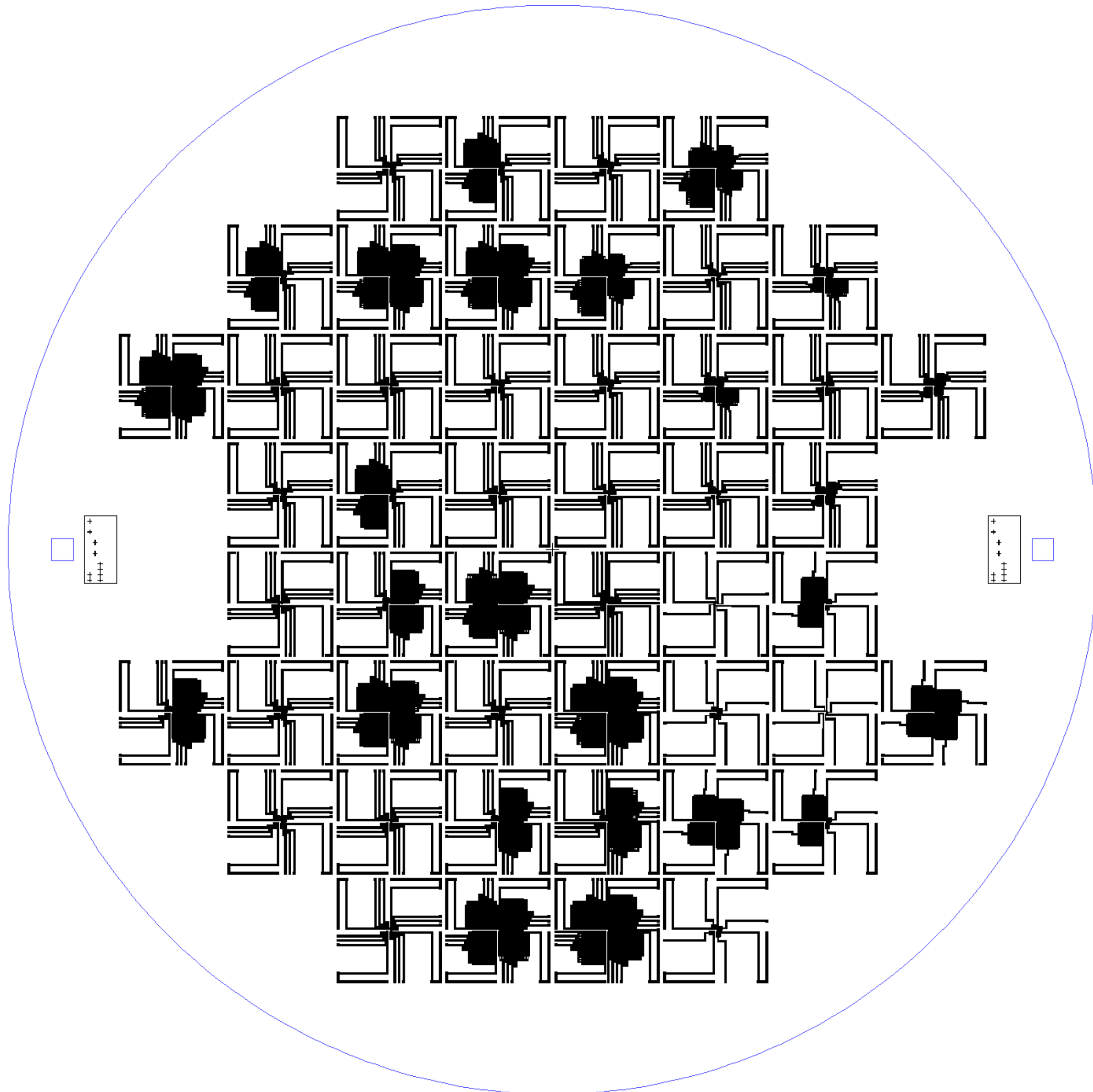


Figure 6.1: Overview of a complete mask for the wafer.

The dimensions chosen for the different capacitances follow from the simulations. Appendix A.4 provides reference tables with the different capacitor sizes that have been included, as well as where on the wafer they are located. It can be seen that the devices of different sizes are spread across the wafer.

6.4. Conclusions

The mask generator provided a flexible tool for creating variation in the device shapes and placement, and allowed for the design of a lithography mask to be used for the device fabrication. A sufficient variety of device types and dimensions is included on the resulting mask to allow comparisons between devices for both electrode dimensions, configurations and etching parameters.

7 Prototype Fabrication

This section will describe the process used to fabricate the physical devices before measurements. The flowcharts that lay out the processing steps and systems used in detail can be found in Appendix A.8.2. The measurement setup will also be described in detail.

7.1. Fabrication of the Capacitor Array

The fabrication of the capacitor array can roughly be divided in two steps. Firstly, wafers are prepared with a layer of amorphous SiC. A lift-off process is then carried out to create the electrodes on top of this SiC layer. An overview of the processing steps is shown in Figure 7.1.

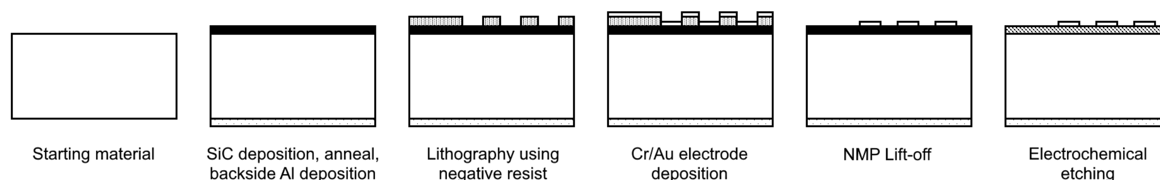


Figure 7.1: Cross sectional overview of the manufacturing process.

7.1.1. Preparation of the wafers with Silicon Carbide substrate

This project used the wafers from a previous project which already had a layer of amorphous SiC formed on the front, and a layer of Aluminium on the backside. A flowchart adapted from these previous steps can be found in Appendix A.8.1.

In this process, the wafer was first implanted with B^+ Ions. After this, a $500\,\mathrm{nm}$ layer of amorphous Silicon Carbide is formed on the wafer, after which an annealing step is done. Following the formation of the SiC layer, the wafer was then cleaned, and $1.4\,\mathrm{\mu m}$ of Al/Si was deposited on the backside.

7.1.2. Electrode Fabrication Using Lift-off Process

Using the pre-processed wafers, the processing starts with cleaning and drying steps to prepare the wafers for lithography. A negative resist was applied to the wafers using a spin-coater. Following this step, the resist was exposed using a mask aligner. The masked with the designed pattern was ordered from Compugraphics and arrived in good quality. The resist was consecutively developed by performing an X-link bake followed by a development bake. To verify the quality of the lithography, the edges and flats were inspected under an optical microscope.

The wafers were next exposed to an O_2 plasma flash to prepare for the gold deposition. After this, a layer of $15\,\mathrm{nm}$ Cr and $185\,\mathrm{nm}$ Au was deposited on the wafers using electron-beam physical vapour deposition.

Then, the lift-off process was performed using heated NMP inside an ultrasonic bath. After this the wafers were again inspected to verify the quality of the lift-off.

7.2. Forming the PASiC layer

With the electrodes fabricated on top of the SiC surface, the devices are ready to be etched to make the SiC layer porous.

7.2.1. Design of a current source for electrochemical etching

In the electrochemical etching step, a constant current is applied to the sample to be etched. Based on currents used in previous work, it was chosen to do etching tests using set currents of $1\,\mathrm{mA}$, $2\,\mathrm{mA}$ and $5\,\mathrm{mA}$. To supply this current a simple constant current was designed to supply these currents to to a load of variable resistance. Any constant current source could have been used, but using a custom built source had some advantages, as its usage wouldn't depend on the availability of material in the intended workspace. Furthermore, because the required currents are low, the circuit could be powered from a battery, making it possible to use the device in fumehoods without power sockets.

The current source was designed as seen in the schematic in Figure 7.2.

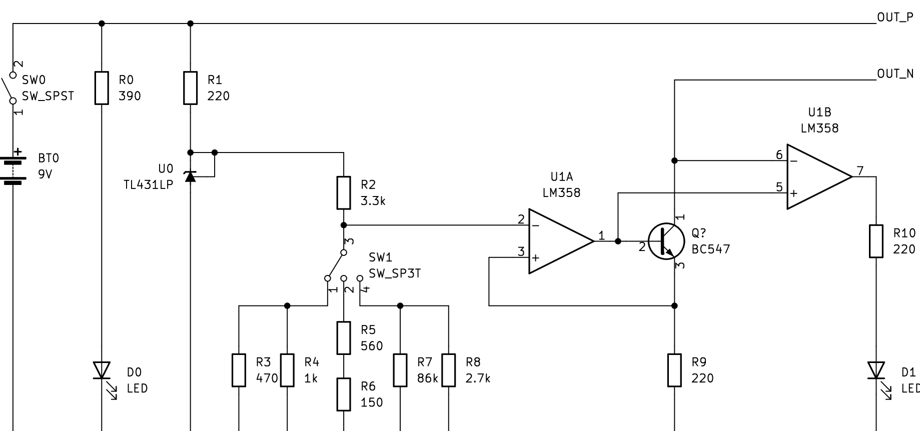


Figure 7.2: Schematic for the used current source.

A $2.5\,\mathrm{V}$ reference voltage is generated with a TL431LP. This reference voltage is then used as the input to a resistive voltage divider. An SP3T switch in the divider can be used to select a different resistive network to select the desired output current. An op-amp and a transistor in negative feedback ensure that the voltage on the sense resistor R9 is equal to the output of the voltage divider. With the chosen resistor values, the emitter current of the NPN transistor becomes $1\,\mathrm{mA}$, $2\,\mathrm{mA}$ or $5\,\mathrm{mA}$ depending on the state of the SP3T switch in the resistive divider.

The circuit is able to provide these currents as long as the load is small enough for the transistor not to enter saturation. Figure 7.3 shows the output of the current source for the different switch positions for different values of the load resistance.

The vertical lines show the resistances at which the transistor goes into saturation. It can be seen that the required current can be supplied to loads of up to $1.4\,\mathrm{k\Omega}$. Both the simulation and the measurements used a battery of $9\,\mathrm{V}$ as the voltage source. The circuit also works with a higher voltage, such as two batteries in series, provided that the current limiting resistors R1, R0 and R10 are replaced. This would allow supplying larger loads if required.

The load consists of the copper wire from the current source to the die to be etched, the die itself, the electrolyte, the platinum electrode, and the copper wire back to the current source. The resistance of the wires and the platinum electrode is expected to be small compared to that of the die and the electrolyte. Furthermore, the resistivity of the Si wafer is known to be low at $5\,\Omega\,\mathrm{cm}^{-1}$ The specific

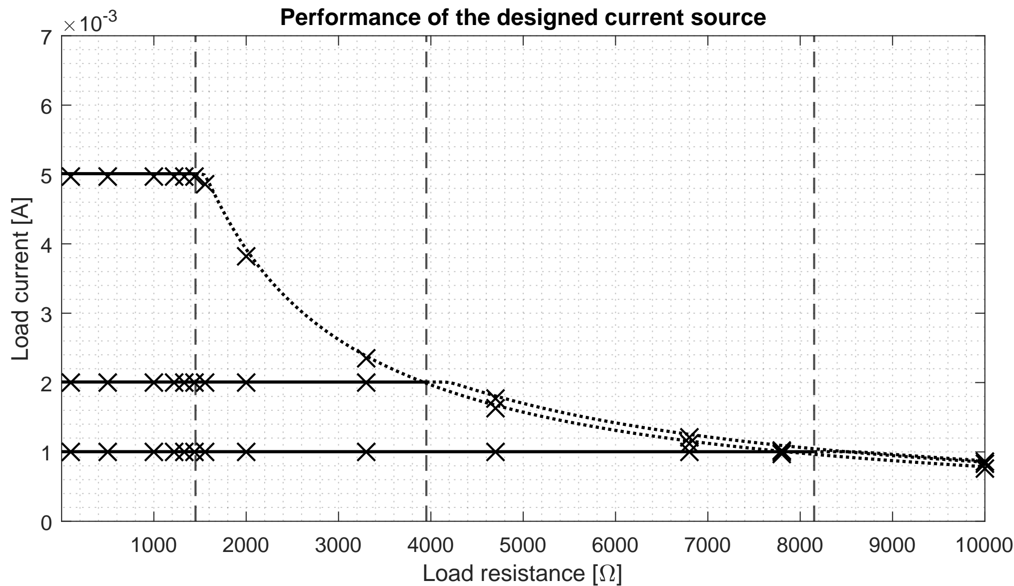


Figure 7.3: Simulated and measured output of the current source.

conductivity of 70% Hydrofluoric Acid is above $\kappa = 0.6\,\mathrm{S\,m^{-1}}$. [28] With the $1\,\mathrm{cm}^2$ die to be etched and at a $4\,\mathrm{cm}$ distance between the electrode and the sample, this would result in a resistance of

$$R_{\text{electrolyte}} = \frac{l}{A\kappa} = \frac{.05}{.01^2 \cdot 0.6} \approx 833\,\Omega$$

As such, it is expected that the load will be below the $1.4\,\mathrm{k\Omega}$ the current source can supply.

As an extra practical measure, the second op-amp of the LM358 package is used as a comparator which compares the base and source of the emitter. This comparator drives a warning LED so that the LED turns on if the load is too high and the transitor goes into saturation. The X markings in Figure 7.3 are the values measured on the actual device. A table with these measurements can be found in Appendix A.5, and the MATLAB code used to generate the plots can be found in **??**.

Ultimately, a high precision SMU was used rather than the designed current source as it happened to be available at the time the etching was done. Using a voltage compliance of $20\,\mathrm{V}$, it was possible to provide the $1\,\mathrm{mA}$, $2\,\mathrm{mA}$, and $5\,\mathrm{mA}$ currents. As such it's likely that the battery powered source would also have had enough compliance with two $9\,\mathrm{V}$ batteries placed in series.

7.2.2. Design of a chip holder

During the electrochemical etching process, the die is kept within the acid, while there is an electrical connection from the back side of the die to the current source. After this process, connections must be made from on top of the die to a measurement setup.

To facilitate both processes, a PCB was designed onto which the die could be mounted. A cross-sectional overview of the PCB is seen in 7.4.

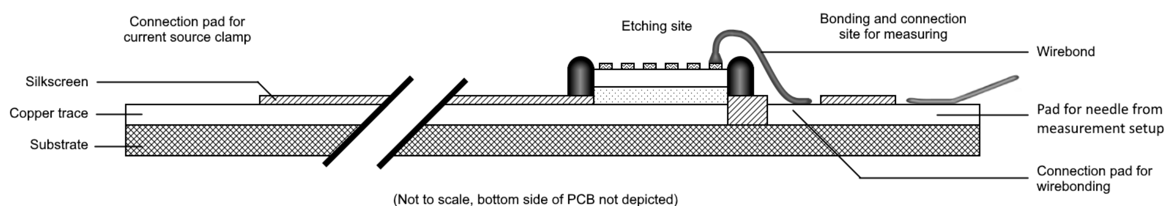


Figure 7.4: Schematic view of the PCB with the chip attached for testing.

The die is attached to a pad on the PCB using a conductive silver epoxy. This is applied on the PCB

using a small synthetic brush. The die is then placed onto the PCB and the epoxy is left to dry at room temperature over night.

A copper trace goes from the pad with the die to a pad on the other side of the PCB. This pad can then be used as a contact for the current source. Additionally, pads are placed near the PCB with copper traces leading to the edge of the PCB. Using wirebonds, connections can be made from geometry on the die to these pads at the PCB edge, which can then be used for measurements.

In order to prevent moisture from getting into the PCB, which could interfere with the measurements, it is manufactured out of aluminium. This is sufficiently resistant against electrochemical etching in the high concentration HF.

To protect sensitive parts of the PCB, such as the copper traces and the sides of the die, Apezion Wax W is applied as an etch resist. Application of this resist was done by dissolving the wax in a small amount of Toluene. This creates a paste that can be applied using a cotton swab. This way, all areas of the part of the PCB with the chip were covered, with the exception of the chip itself. This includes the edges. As the solvent evaporates from the wax, it leaves a thin layer of wax that is well attached.

Removal of the wax was later done by again dissolving it in some toluene, followed by a wash in acetone.

The chosen layout for the PCB can be seen in the schematic in Figure 7.5.

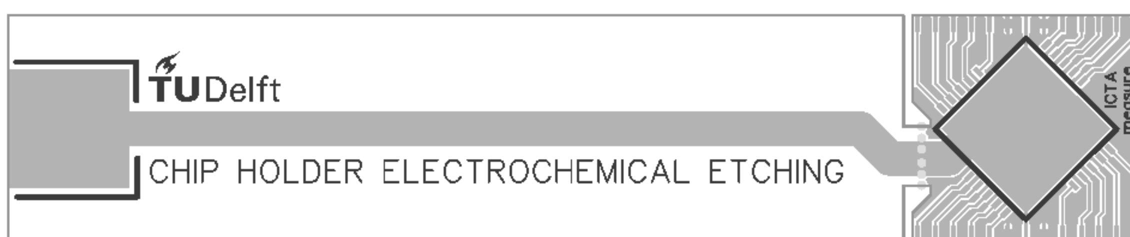


Figure 7.5: Schematic view of the PCB used for the electrochemical etching step

A limitation of the measurement setup was that the maximum PCB size that could be used was 2 by 2 centimeter. For this reason, the front part of the chip is connected using mouse bites. This allows it to be broken off to reduce the PCB to a smaller size to fit the measurement setup. The broken off part is shown in Figure 7.6.

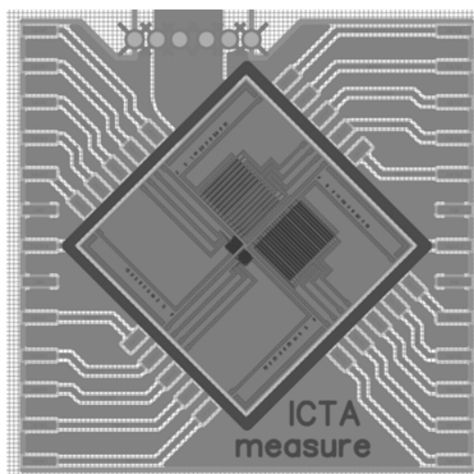


Figure 7.6: Schematic view of the part PCB with the chip attached for used for measurements.

In the measurement setup, needles make contact with the pads on the side of the PCB to make an electrical connection to the geometry on the chip.

7.2.3. Electrochemical etching setup and procedures

After a chip is glued to the chip holder PCB, and after wax is applied to parts of the PCB sensitive to the acid, the chip can be etched. An overview of the etching setup that was devised can be seen in Figure 7.7.

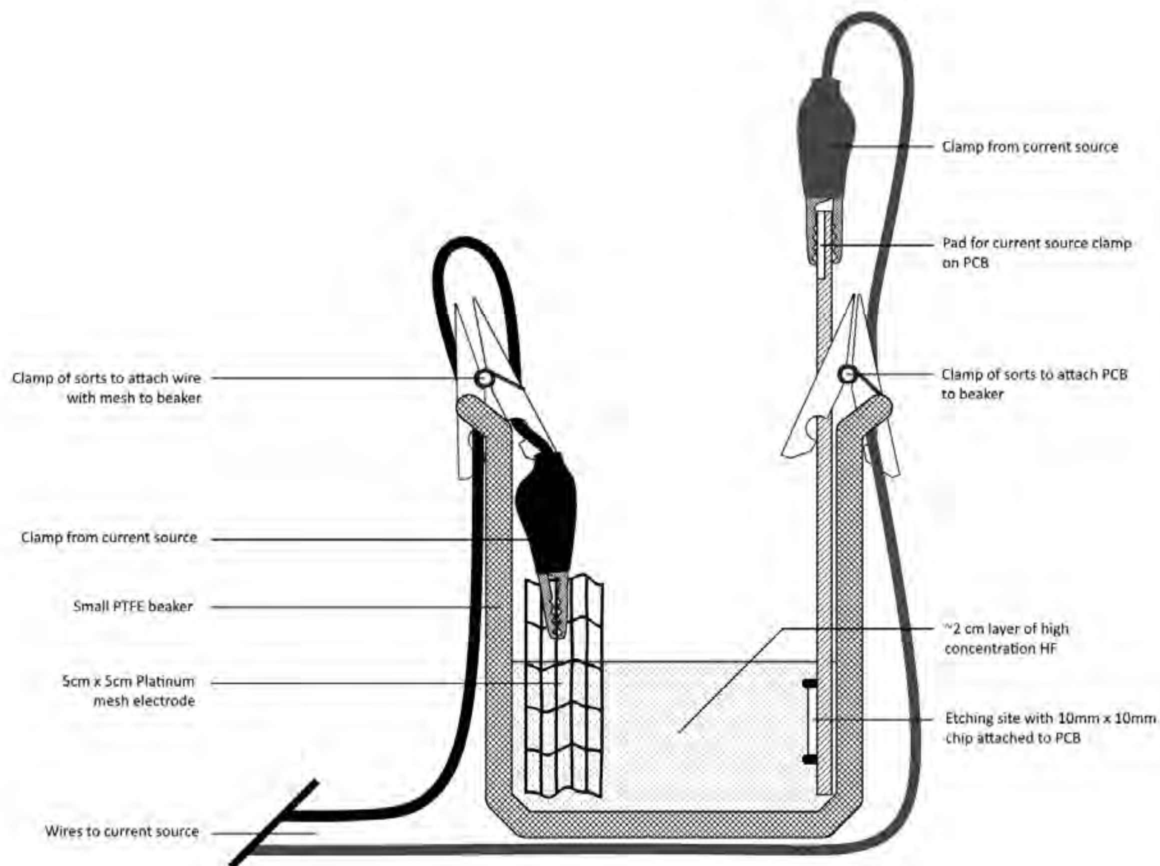


Figure 7.7: Schematic view of the electrochemical etching setup.

The clamps from the current source are attached to the Platinum mesh electrode and the pad on the PCB. Two plastic clothespins where then used to hold them in place within the beaker.

To this beaker, undiluted 40% HF was added, along with a drop of Triton X100 surfactant. the current source is then switched on for a set duration. After the etching time, the current source is switched off and disconnected. The chip is subsequently removed from the beaker and thoroughly rinsed with a number of water baths and left to dry on a paper towel. Another PCB with a chip attached can then be attached to the source and placed in the beaker for etching.

7.3. Conclusions

While the lithography process was successful, some issues with the starting material arose during the fabrication process.

When the wafer was inspected after deposition of the resist, two problems with the wafer became evident. Firstly, while the edges of the structures were well defined across the entire surface, any gaps below $1\,\mu m$ were not present. This was to be expected, as these lengths are very close to the minimum resolution of the used mask aligner. This is not a problem, as the capacitor array includes many more capacitors of larger sizes and the smaller designs were not critical for evaluation of the designs. Secondly and more importantly, cracks were visible in the films. This was known beforehand, as there was a note on the presence of cracks in the flowchart from which the wafers with the SiC originated. Upon inspection the other wafers from the previous process had the same cracks present.

However, the extent of the influence from the cracks on the lithography process was uncertain. The width and depth of the cracks, as well as what layers they were in were not known. It was decided to continue with the process regardless. Depending on the dimensions of the cracks, the deposited gold can bridge the gap without losing connectivity. Even with the cracks present, the chips resulting from the badge would be useful for experimenting with the etching setup.

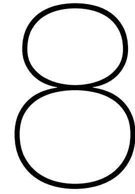
After the lift-off process, the wafers were inspected again. With the gold traces deposited, the cracks were still visible, including through the traces. Therefore, measurements were made to determine whether or not the gold traces were interrupted by the cracks using an LCR meter. Unfortunately, it turned out that this was indeed the case.

Two options to continue from this point were either attempting to use the chips with the cracks in them, or to redo the process from the start, including the deposition of the SiC and Al layers. The latter would involve getting the required training for these deposition steps, or finding someone with the required training who was available to carry out the steps. Due to time constraints, it was chosen to firstly continue with the chips for usage in experimenting with the etching setup.

It was also discussed to wirebond over the cracks within the wafer. It would have been necessary to make wirebonding connections from the pads on the chip to the pads on the PCB before the measurements, so making extra wirebonds here would only have been a minor inconvenience. Devices where the cracks did not run through the sensing area could then still be used for some measurements, provided that the cracks would not interfere with the etching process. The capacitors on the PCB were inspected to make a selection of devices which could be saved this way, and some wirebonding tests were done to verify this possibility.

Unfortunately, these options would ultimately be unusable, as following the etching experiments it became evident that the cracks would influence the etching process considerably, resulting in unusable devices.

Because a lot of time passed between the lithography process (mainly due to problems with the availability of the chemicals and regulations regarding their usage), there was little time to redo the process at this point. Therefore, the following Chapter on the measurement results will focus only on measurements of the unetched devices with a comparison to the simulation. The chapter will also include documentation on the inspection steps of the devices before and after etching.



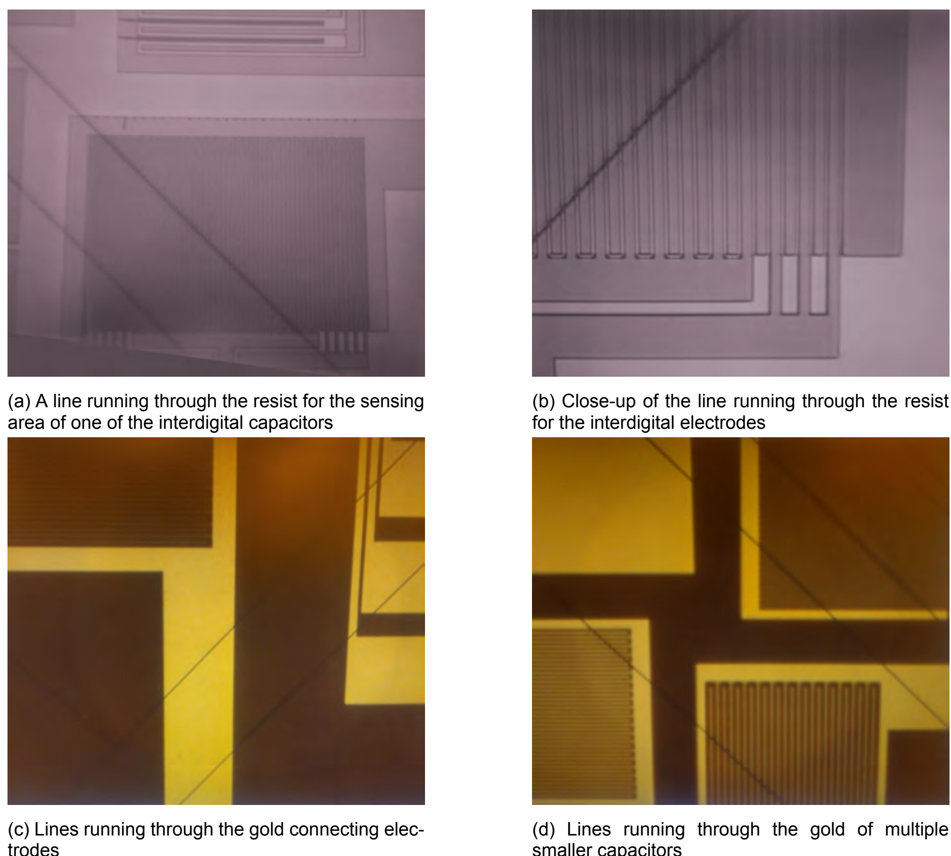
Device Inspection and Measurement Results

Because of the issues that arose during the fabrication process, results primarily involves the inspection of the devices to determine the cause of the defects and possible ways to mitigate the problems. Some measurements have been carried out still on the unetched devices. This section documents the results of these inspections and measurements.

8.1. Device inspections

8.1.1. inspections during and following the lift-off

Within the lift-off process, the wafer was inspected multiple times using an optical microscope, primarily to validate the quality of the resist structure and gold traces. During these inspections, lines were observed running through the devices, as is shown in Figure 8.1.



(a) A line running through the resist for the sensing area of one of the interdigital capacitors

(b) Close-up of the line running through the resist for the interdigital electrodes

(c) Lines running through the gold connecting electrodes

(d) Lines running through the gold of multiple smaller capacitors

Figure 8.1: Microscope images taken during Au line inspection after lift-off

As is visible in the images, the lines run through the structures mostly in diagonal directions. The lines were visible both during the inspection of the resist, and remained visible both on the substrate and the gold substrate after the lift-off step. The effect of these lines was at the time mostly unknown. It was assumed that they were a result of some height difference within the wafer being present already before the lift-off process. A note was made in the flowchart of the process for the starting wafers with deposited SiC, which the that stated that cracks were observed. Visual visual inspections of the unused starting wafers confirmed the lines were present in all these wafers. The most likely defect then seemed to be cracks running through the surface, although other options such as a step in height were not excluded.

Following these observations, it was decided to measure the capacitance of some unetched devices on the wafer to determine whether the cracks would interrupt the gold traces. Using an LCR meter, the capacitance was first measured with the probes at the bases of two of the interdigital electrodes (Figure 8.2). Then, the same device was measured with one of the probes moved to a position which had a crack between the probe and the capacitor.

The measurement setup of the LCR meter compares the measured results to a model of a capacitor in series with a resistor to estimate the value of the capacitance and parasitic resistance. Measuring across the gap resulted in a modelled resistance several orders of magnitude higher. As such, it was concluded that the cracks indeed break the gold traces.

Therefore, to decide what should be done in the continuation of the process, the entire wafer was inspected under an optical microscope to determine what capacitors would be usable. Figure 8.3 shows an overview of the wafer with the different defects.

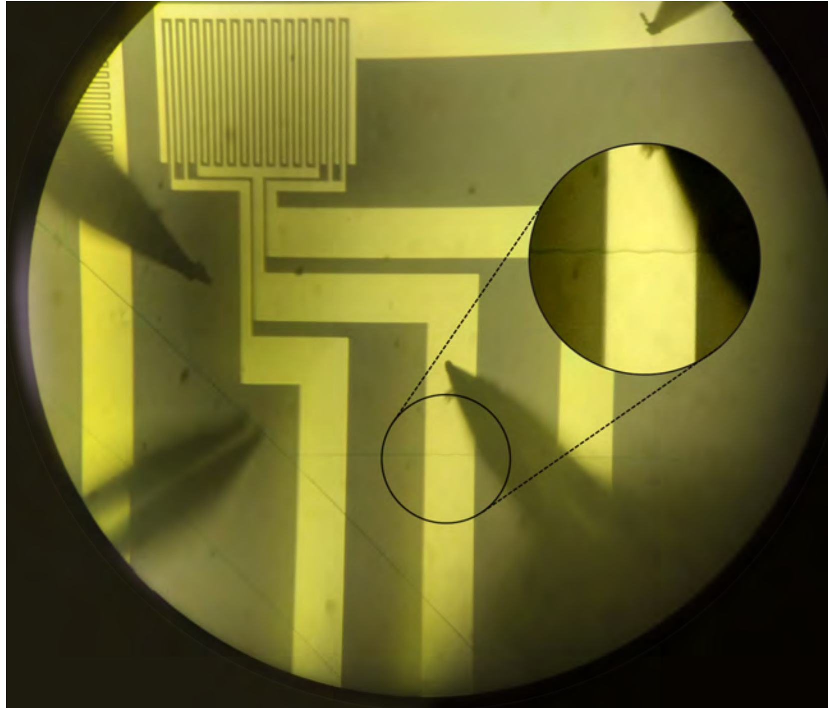


Figure 8.2: Microscope view of the probes lowered onto the base of the capacitor in order to test the influence of the lines. The faint, horizontal zigzagging crack below the probe across which was later measured is shown in an enlarged view.

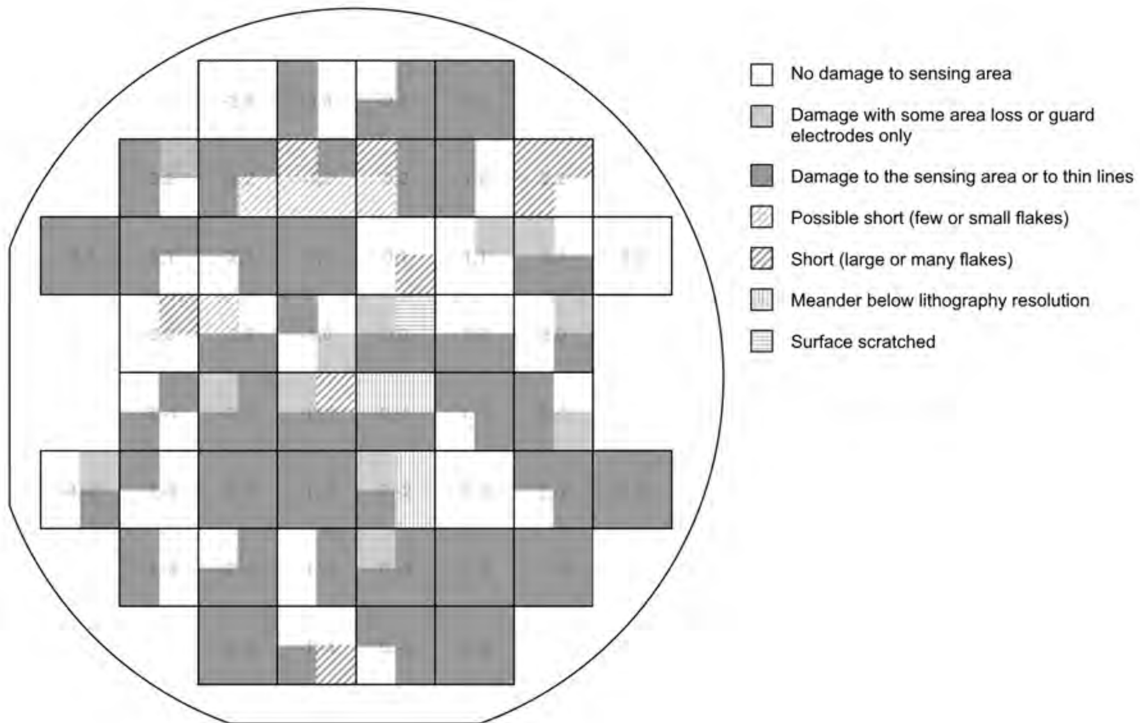


Figure 8.3: Schematic giving an overview of the state of the capacitors on the wafer

The colour of the device in this image represents the type of damage that the capacitor had. Different effects were documented, including a few scratches (likely due to a handling mistake at some point in the process), some shorts caused by residual flakes from the lift-off process, and the aforementioned cracks.

Though multiple defects were present, the cracks remained the most prominent form of damage. In fact, it turned out the cracks ran across the entire wafer, so that there were no devices that did not have cracks running through the electrodes at some position. Since the electrodes of the devices cover most of the die, the chance of a line running through the device becomes high. As a result, the yield is very low. However, although all devices had some damage to the connecting electrodes, there are still devices which do not have damage to the sensing area. Because the wafer had a large number of duplicate devices, these devices include a large enough variety of capacitor geometries to allow for some useful measurements.

It was therefore decided to at least try to conduct some measurements with the devices, and use them as a starting point in the etching process. By using the LCR probe station, it was possible to do measurements of devices that did not have damage to the sensing area by connecting the probes near the base of the capacitor. The results of these measurements will be discussed in Section 8.2.

8.1.2. Inspections after dicing

After these measurements, the wafer was diced into $1\,\mathrm{cm}$ by $1\,\mathrm{cm}$ dies, each containing four devices. These chips could be used to do inspections under a scanning electron microscope. SEM images were firstly taken of the image to determine the dimensions. Firstly, an image was made from above, as shown in Figure 8.4

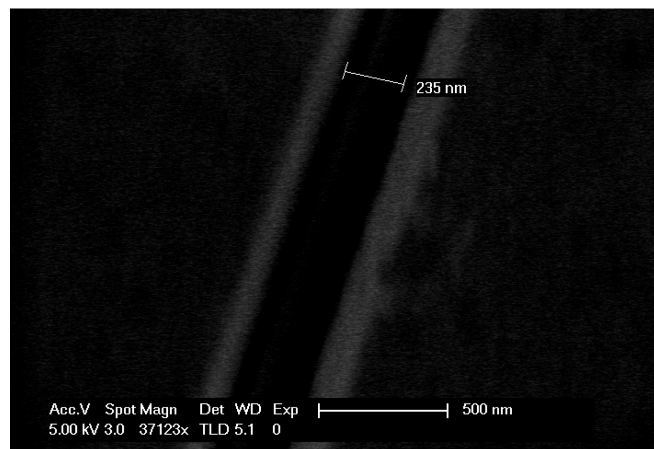


Figure 8.4: Schematic giving an overview of the state of the capacitors on the wafer

It can be seen that the crack causes separation in the electrodes. The separation was measured from the SEM image as about $240\,\mathrm{nm}$.

However, the depth can not be estimated. To this end, a chip was broken under pressure in an attempt to visualise a cross section of the crack. Because the chips were already diced, it was hard to break the die cleanly in the substrate direction, resulting in a rather rough break line. The broken chip was first observed under an optical microscope to determine a locations to be inspected. The chosen location is shown in Figure 8.5

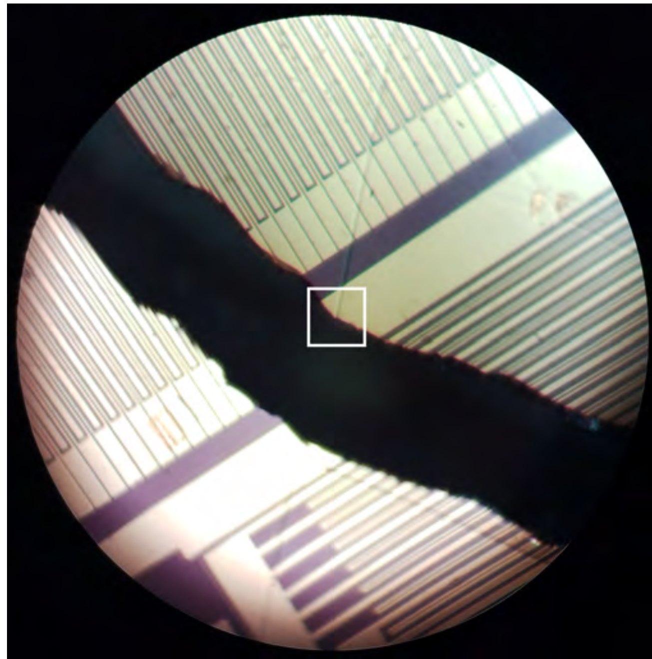
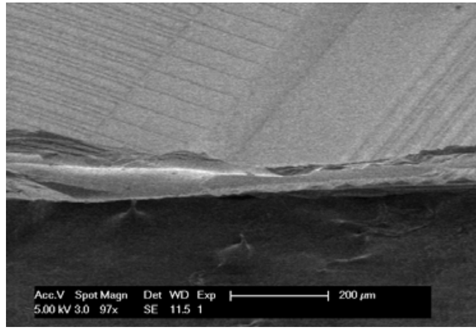
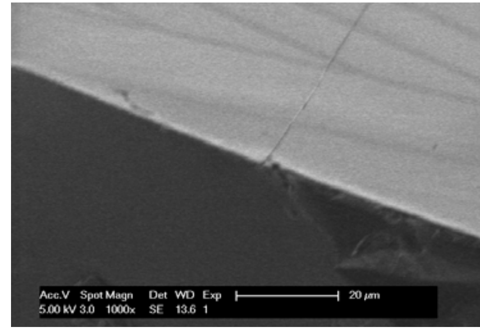


Figure 8.5: Schematic giving an overview of the state of the capacitors on the wafer. The square in the centre shows the gap distance to be about

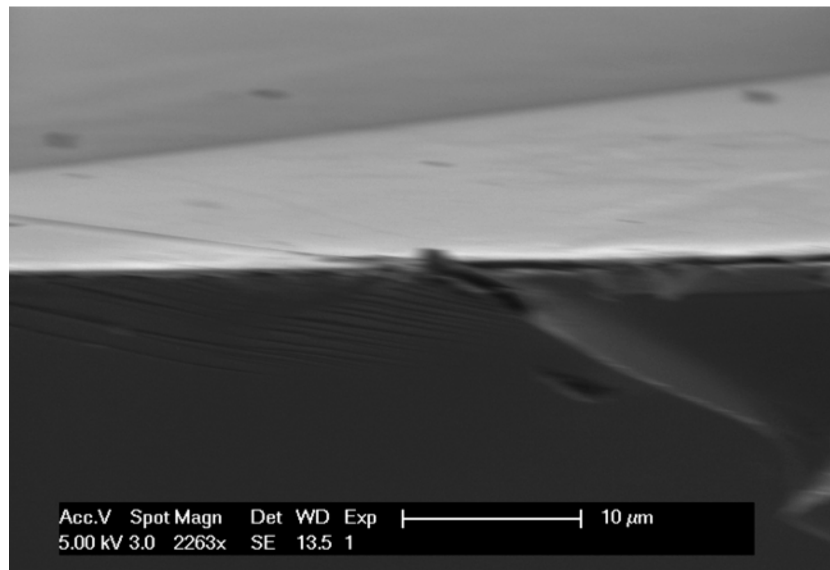
On the chosen location, a line runs through a gold line at the edge of the wafer. The fragment of the chip was observed under the SEM microscope from different angles. The resulting images are displayed in Figure 8.6.



(a) Crack location as viewed under the SEM



(b) Close up of the crack from the left side



(c) Close up of the crack from a side view

Figure 8.6: SEM images of a crack running through a gold trace on the edge of a broken die.

It is seen that the line is visible deeper in the substrate than the $500\,\mathrm{nm}$ SiC layer. From the close-up in Figure 8.6c, the crack seems to run at an angle to the surface, in alignment with the with the substrate crystal direction. It is possible that the act of breaking the chip may have caused the crack to propagate further into the substrate, but it remains notable that the cracks run both through the SiC layer into the Si substrate.

Knowing that the cracks do interrupt the lines, but that there are devices on which the cracks do not run through the sensing surface, methods were investigated to wirebond across the cracks on the electrodes connecting the sensing area to the pads. This would bypass the open circuit, although it might have added some parasitics. A result of this experiment was seen in Figure 8.7

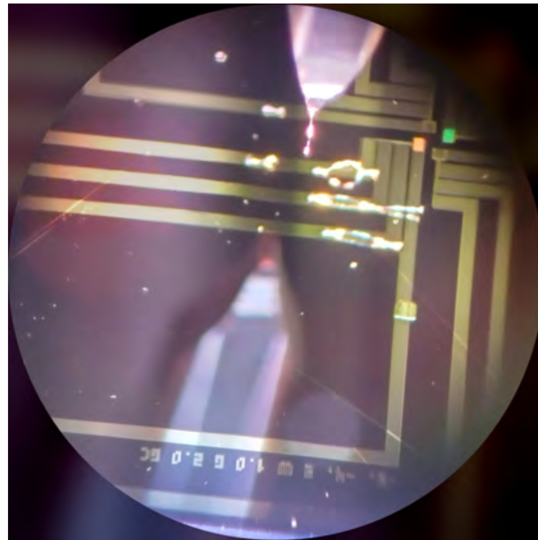
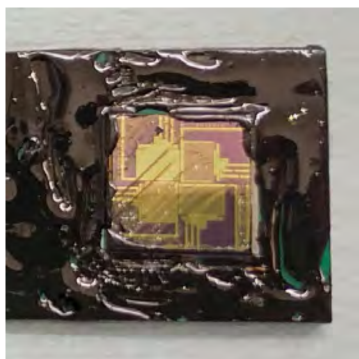


Figure 8.7: Image of a few test wirebonds on the connecting lines.

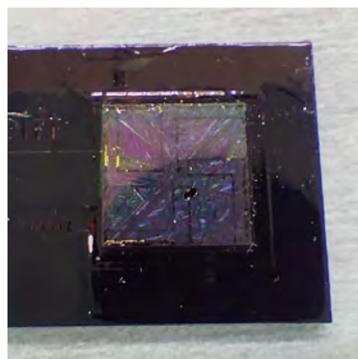
Using these wirebonds, it is possible to cross cracks on the connecting electrodes. Other parts of the electrodes may be too small to practically bond across. It should be noted that a second issue was found when experimenting with the wire bonding process. The pads on the CPB to which wirebonds from the chip should have led were covered by a thin layer of HASL. However, this material has a too low melting temperature, making it too soft to wirebond to under pressure. Therefore, if similar PCBs are made in the future, the design should be changed to leave the copper exposed, so that it is possible to bond to the pads directly. Ways to continue prototyping with the existing PCBs were explored to save on time. The first proposed solution was to mechanically remove insulation layer of the copper traces. It would then be possible to wirebond to the traces instead of to the pads. A second solution would be to attach small gold or other conductive shims to the HASL using a conductive glue, which would the serve as a more stable bonding surface.

8.1.3. Inspection of the devices after etching

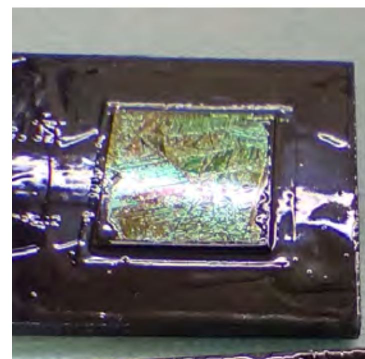
Following these tests, it was chosen to continue with etching of the chips to experiment the etching setup. Following the procedures described in Chapter 7, the chips were prepared and etched. After the etching process, it was immediately apparent that there was an issue with the resulting samples. Extensive damage to the electrodes was visible to the naked eye, as shown on the photographs in Figure 8.8.



(a) Patterned die, etched with $2\,\mathrm{mA}$ for 2 minutes. Large lines are visible through the electrodes.



(b) Patterned die, etched with $1\,\mathrm{mA}$ for 10 minutes. The electrodes have separated from the surface.



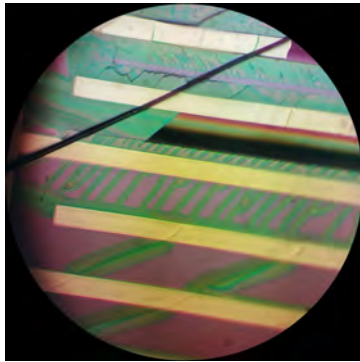
(c) Blank die, etched with $5\,\mathrm{mA}$ for 5 minutes. Surface has a shattered crystalline look.

Figure 8.8: Photographs of the chips attached to the PCBs after etching

Etching currents of $1\,\mathrm{mA}$, $2\,\mathrm{mA}$ and $5\,\mathrm{mA}$ were used with etching times between 1 and 10 minutes. Damage was present regardless of the etching times used, though as expected higher etching currents and longer etching times generally resulted in more damage.

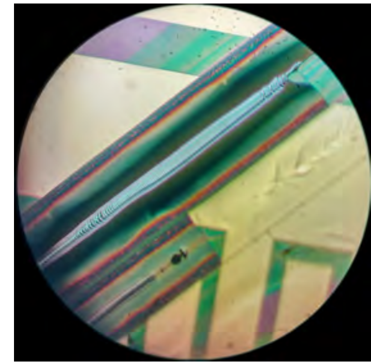
Even on devices which were etched for a short duration and with a low current, the cracks had visibly widened, interrupting the electrodes. On some of the devices with high etching times, the electrodes have been removed entirely. In general, rather than an more diffuse surface, which would be expected from PASiC, the surface took on a fractured, crystalline look.

Optical Microscope Inspections Further inspections were carried out to get insight in how the etching had changed the surface on a microscopic level, starting with inspection of all devices under an optical microscope. A series of photographs taken through the microscope that shows various forms of damage to the electrodes is displayed in Figure 8.9



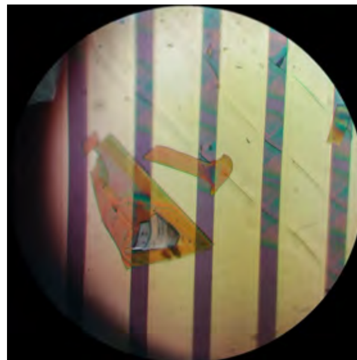
(a) Cracks through the wafer have become wider and more apparent. Deformations are seen in the surface.

(b) The cracks combined with damage to the surface cause the electrodes to lift off the chip

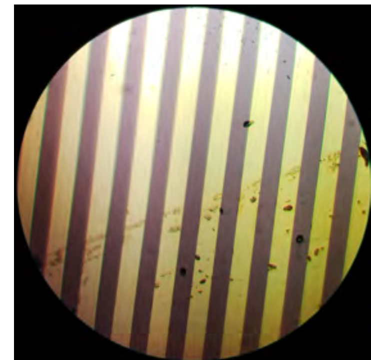


(c) With large etching currents and long times, wide trenches appear through the electrodes

(d) The top layer is fractured and separated from the surface, taking the gold traces with it



(e) Glassy flakes have separated from the surface and are strewn across the wafer, ending up on top of the electrodes



(f) Some areas without cracks have remained undamaged at lower etching currents and etching times

Figure 8.9: Microscope photographs of traces on various areas of the etched dies

From the microscope images the extent of the damage to the electrodes becomes more clear. The cracks have increased in size dramatically, resulting in large interruptions of the electrodes. At higher currents, the cracks. The number of cracks also seems to have increased, as areas on the chips that had been documented as not having cracks present before the etching show damage similar to the areas that were known to have cracks.

Furthermore, there are large areas where the surface separates from the surface of the die and flakes off. Gold traces that were on top of the surface in these areas is partially or entirely separated from the surface, causing further breaks in the electrodes. The flakes have been moved around the wafer, resulting in thin, transparent flakes across the wafer.

It is notable that not all areas were damaged. With combinations of low current and short etching times, there were areas without cracks that did not have damage to the electrodes. The devices were still not usable, however, as the electrodes were damaged in other areas close to where cracks were present.

A chip without electrodes was also inspected to verify the larger structure of the damage to the dies. Figure 8.10 shows a view of the large scale cracks running across the wafer.

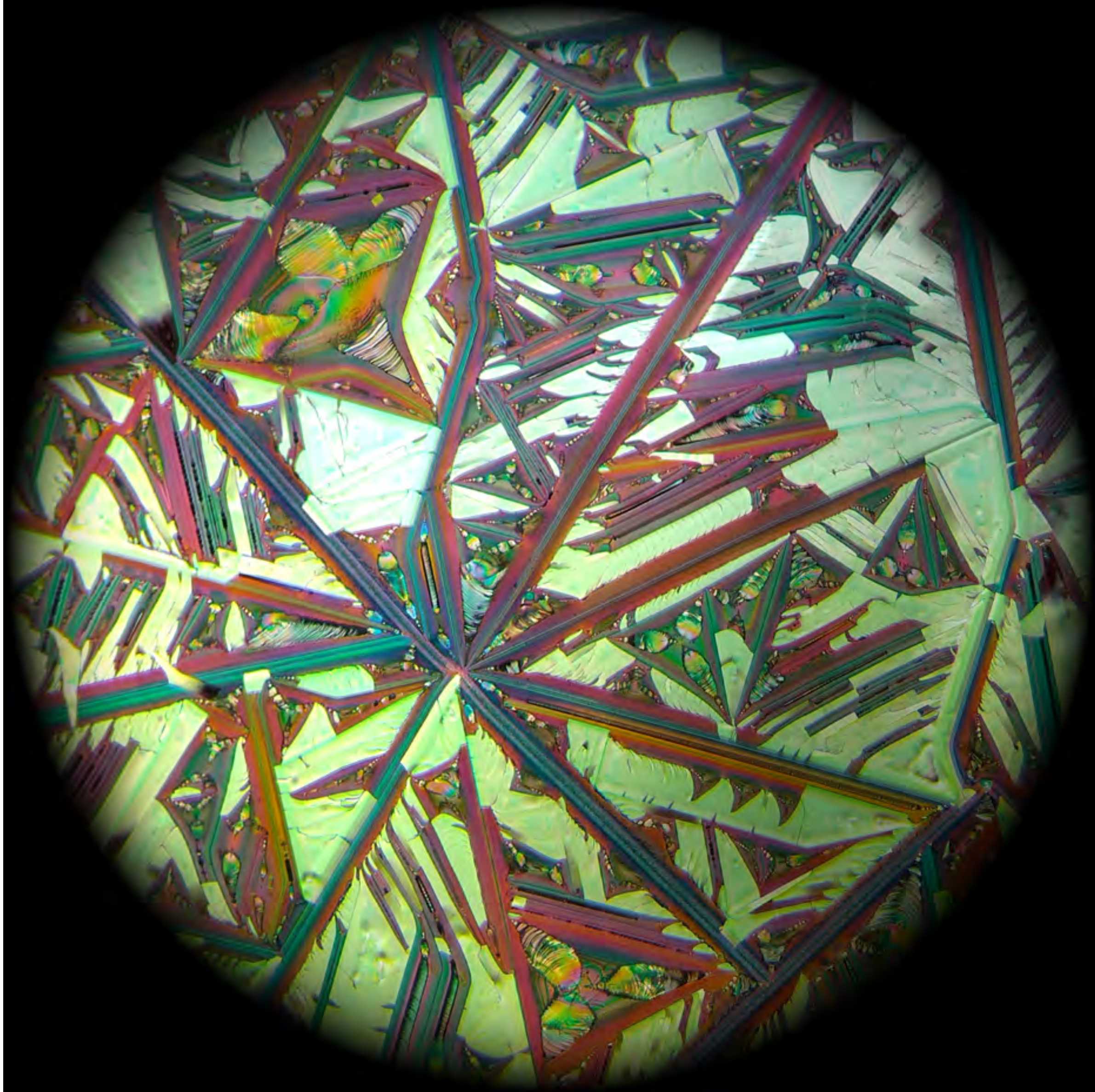


Figure 8.10: Image of a few test wirebonds on the connetive lines.

To create the image, three photos of overlapping areas were taken through a microscope and digitally stitched together using OpenCV. The script used for this can be found in Appendix **??**. The colour of the image has not altered otherwise.

While not particularly informative, it makes for a spectacular sight. Cracks run in various crystal directions, meeting at some. Especially Near the edges of the die, some areas have many lines running in parallel.

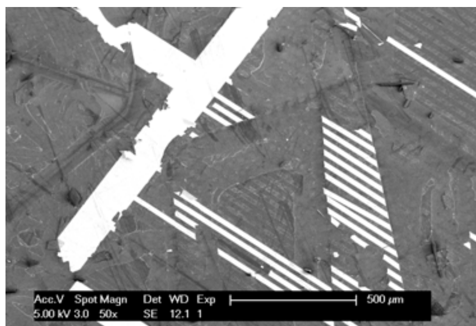
The colours are likely a result of optical effects following from the different slopes and textures of the surfaces.

At some places, yellowish flakes lay across the lines. The lighter colour indicates that the flakes are more reflective. It is reasonable to assume that the lighter areas in the image still have the top layer still attached to the surface. It can then be concluded that the flakes seem to separate mostly near the lines in the surface. Smaller fractures in the top layer are visible. The disorderly nature of these fractures compared to straight lines of the cracks makes it likely that these result from a different mechanism,

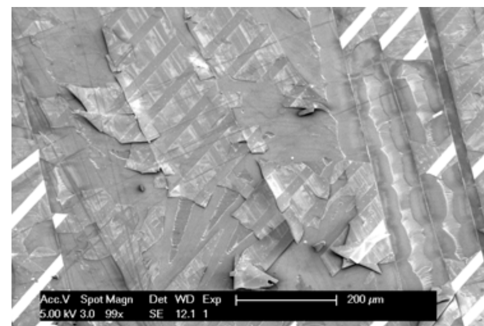
most probably stresses in the layer.

In areas darker areas where the surface seems to have separated, Numerous fractal-like patterns run from the corners of areas between the cracks, meeting at the centre. These are presumably formed when the etching process starts at a corner of the surface, and slowly exposes new crystal directions as the corner is etched further inwards. These areas also display holographic bands, which could be caused by effects of etching in these areas.

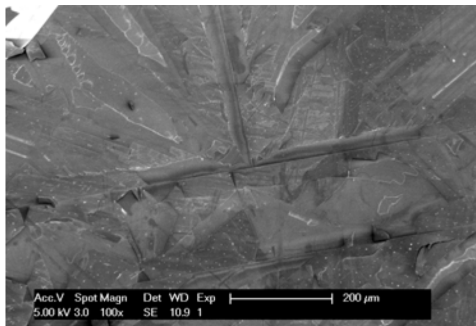
SEM Inspections Though the etched cracks appear quite wide and affect the gold traces within their apparent width, little can be said about the structure of the defects because of the various optical effects present. Therefore, SEM images were also taken to gauge the depth of the etched cracks. A selection of SEM pictures is shown in Figure 8.11



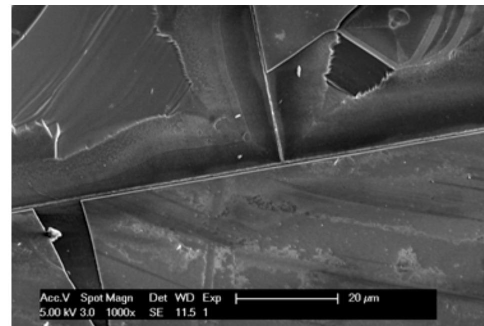
(a) SEM Image displaying the extensive damage to the electrodes caused by the etching process



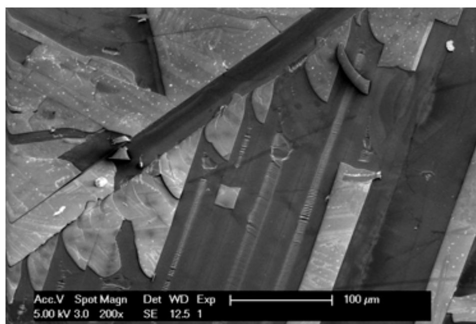
(b) A close-up of the damaged lines shows the surface flaking and separating from underneath the gold traces



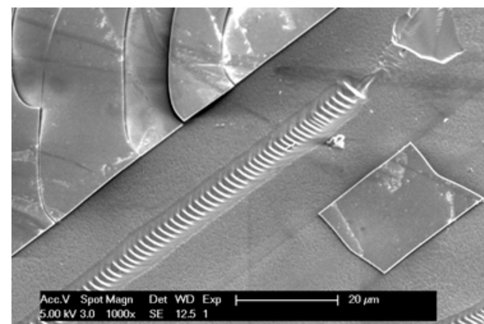
(c) A junction where two etched-out cracks meet. Cracks and flaking are visible on the surrounding surface



(d) An image with a higher magnification of the same junction. The top layer continues up the centre of the crack



(e) Image showing the parallel lines near the die edge at an angle.



(f) One of the parallel lines from close-by. The surrounding surface seems notably porous.

Figure 8.11: Microscope photographs of traces on various areas of the etched dies

Under the SEM, the surface defects become even more apparent, as seen in Figure 8.11a and 8.11b. The delamination typically ends at lines which are assumed to be the cracks. In some areas

where the gold is removed, the electrode patterns are still visible. From the close up, it becomes apparent that in these areas the top layer is still attached, while the electrodes have separated.

Figure 8.11c and Figure 8.11d show the same location where two cracks meet. On this particular junction, the cracks themselves seem to be fairly narrow but well defined. The top layer in the area surrounding the junctions appears heavily damaged. However, the surface appears to run up to near the crack on one side. Areas where the top layer is removed show patterns indicating that the area has been affected by the etching process.

In Figure 8.11e and 8.11f, some of the parallel cracks near the edge of the die are shown. In this area, two types of lines are visible. Firstly, there are wider, heavily ridged bands, which appear to be the most affected by the etching. Between these wider bands lie shallow, faint lines. An interesting question arises when considering how these lines are formed.

The largest 'trenches' where the top layer has separated, such as the ones predominantly visible in Figure 8.10, typically have the ridged band at the centre. One might therefore assume that this is the centre of where cracks were present before the etching.

However, in all these cases a faint line is present on both sides of the trench. Flakes and the edge of the top layer commonly end in a straight edge near the faint lines, and only rarely run up to the centre of the ridged lines. There are examples on the dies where the edge of the trenches continue beyond the trench, such as seen previously in Figure 8.9c. Furthermore, there are trenches without a ridged band, and ridged bands typically end before the lines.

This suggests that not at the ridged lines, but at the fainter lines is where the cracks were originally present in the top layer. The hypothesis is then that the ridged bands form between two cracks as the top layer between these lines delaminates.

What's further noticeable is that, as seen in Figure 8.11f, the area surrounding this type of cracks appears porous. Considering the thickness of the removed area, the surface here consists of mostly silicon. Silicon is known to be etched more strongly than SiC, and result in similar pore morphologies under electrochemical etching with HF. It is therefore is not unexpected to have some form of porous silicon in these areas.

8.2. Measurements

Because of the issues in the fabrication of the prototype devices, measurements on the response to ammonia have not been possible. It has been possible, however, to carry out some measurements on the unetched devices using a probe station. This way, a comparison to the model can still be made, albeit not an ideal one. Because of the low yield, only a limited amount of capacitor sizes could be measured, and the measurement results are of a lower quality.

Table 8.1 shows the measured capacitance values for a selection of capacitor sizes.

Table 8.1: Measured capacitance compared against simulations

G [µm]	W [µm]			
	2	5	10	20
2	1.8 pF 0.932 pF (1.94)	35 pF 0.915 pF (38.2)	33 pF 1.07 pF (30.7)	
5	24.4 pF 1.87 pF (13)	41.5 pF 1.55 pF (26.9)	29.7 pF 1.59 pF (18.6)	4.11 pF 0.959 pF (4.29)
10	41.2 pF 3.59 pF (11.5)	11.5 pF 2.69 pF (4.26)	25.5 pF 2.51 pF (10.2)	29.1 pF 1.35 pF (21.5)
20		42.4 pF 2.62 pF (16.2)	46 pF 2.25 pF (20.4)	32.7 pF 2.19 pF (15)

Table 8.2: Comparison of the measured capacitance to the simulated capacitance (in green) as well as the ratio between them (in purple)

This table was generated using the MATLAB code in Appendix A.9.1. It can be seen that the measured values are larger than the simulated values, but there is no clear pattern in the capacitance increase related to the device dimensions. That the capacitance is higher could have a number of

causes. One major contributor is that due to limits of the probe station meant that the measurements have been carried out on devices with guard electrodes, but without the guard electrodes properly grounded. This means that the measurements include fringe capacitances, and are influenced by charges on the other electrodes. Another explanation for a higher capacitance that's more related to the simulations is that the silicon carbide is unetched. This is expected to have less influence, however, as from the simulations one would expect this to affect the measurements on smaller devices more, whereas in the measurements such a bias is not clear. Defects in the wafer may also have contributed to variation in the measurement, since in some cases cracks were present near the capacitor that was measured.

8.3. Conclusions

It is evident that given the starting materials for the process the etching process resulted in unusable devices, even though etching parameters were used that have given good results in the literature.

Based on the damage surrounding the cracks and the extensive delamination, a logical conclusion is that tensile stresses in the deposited SiC surface cause the top layer of the wafer to fail during the etching process. There are two main hypothesis as to how the cracks affect the etching process.

Firstly, it is expected that the electrochemical etching process weakens the SiC, causing cracks to further propagate. Tensile stresses in the SiC layer could cause the material to curve, resulting in vertical forces between the layers which would result in the SiC layer delaminate.

The second hypothesis is that during the etching process, the cracks can allow the etchant to reach to the silicon underneath the silicon. This silicon has a higher etch rate. The SiC will be etched less as the current density at the silicon etching front will be higher. This way, the Silicon Carbide layer can be under etched this way, causing it to separate from the Silicon beneath.

Because the dies made with the starting materials all contained cracks, devices could not be finalised for gas measurements. However, it should be noted that since areas without cracks were not affected at low current densities for short durations, and since certain areas did appear porous as a result of the etching, the etching setup is promising for continued usage with better starting materials.

9 Conclusions

9.1. Conclusions

Though the device has not been developed entirely, the project has laid more groundwork for future research towards the development of these sensor types.

The layered simulation has allowed insight into how the electrode dimensions influence the sensing capabilities of the device for different material changes that occur under testing conditions. The simulations can be adjusted based on measurements to allow for designing devices with improved sensitivity.

The designed mask has been used to create a test bench with a range of capacitors to be used to compare the sensitivity of devices with different electrode geometries or etching parameters. Furthermore, with the Mask generation software, new test benches can be practically designed with other electrode geometries. With some adjustments, the same design models would also be suitable for other applications where a range of devices with variations is desired for testing.

Although the electrochemical etching parameters still have to be optimised with working wafers, the etching setup devised in this work has shown to be a practical way of carrying out these tests. It allows safe batch processing of chips using small amounts of HF, while allowing measurements to be done afterwards. The designed chip holder PCBs and the current source may need to be adjusted to suit other measurement setups and current ranges or etching loads, but the principle of using the PCB for etching as well as measurements was convenient as it removes the need to separate the device from the PCB.

9.2. Recommendations for future work

Future research towards these sensor types will show whether the it is a competitive option with respect to other ammonia sensors. There is still much research needed to fully characterise the working on capacitive sensors based on porous SiC. This section proposes a number of attributes that could be considered in this future work.

For the continuation of this project, the first objective would be to find out what caused the stresses in the SiC layer that resulted in the cracks. The settings used during the processing steps, especially those for the SiC deposition and the anneal, would have to be investigated. This to determine whether the problem may come from an error in the process parameters or a problem or change in the equipment. Once this has been sorted out and a stable SiC layer is formed on the wafer, the process of this fabrication process of this project can be repeated so that new etching tests can be done so that the sensitivity can be measured.

Further recommendations for future work involve possible changes to the model and device that may be made after this characterisation. These changes may improve the performance of some aspects of the sensor beyond what is possible with the design chosen here. The following list enumerates possible candidates for improvement:

Changes to the models On the test bench mask, devices with different electrode wavelengths have been included. This could be useful for improving the models, as the larger the wavelength of the

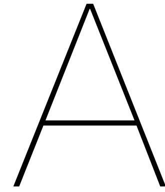
devices, the deeper their electric field penetrates the device and therefore the more sensitive the device is to deeper layers. This can be used to estimate the permittivity of the device with depth. This data could be fed back into the simulation to allow simulations that are closer to reality.

Investigation of the device durability One potentially attractive attribute of the PASiC sensor is it's durability when sensing corrosive gasses. Further work could be focused on investigating this durability in a comparison with for instance porous silicon or other sensor types. As mentioned earlier in this work, it is possible to place the electrodes underneath the carbide layer to protect them against the corrosive ammonia gas. The distance to the surface of the porous layer where the porosity is the highest would be increased, but since the carbide layer is very thin this is not expected to be a problem.

Optimising the device sensitivity To improve the sensitivity of the device, the device response for different etching parameters and size and electrode configurations would first have to be characterised. These parameters could then be optimised to get a starting benchmark for this specific sensor type. There is also room of improvement in the sensor material. Although in this work amorphous porous silicon carbide is used, ammonia sensors have used other types of porous SiC with good results. Investigation of combinations of these different types of Porous SiC with a capacitive sensor could reveal alternatives with more desirable sensing characteristics.

Investigating the device selectivity Another attribute of the sensor that has not been investigated is its cross sensitivity. From previous work, it is known that the sensitivity of the device to different gasses depends on the pore morphology and size. One suggested option is to include different porosities on the same device. If the sensitivity of devices with a certain pore size to certain gasses is characterised, different devices could be constructed on the same chip that are designed to be sensitive to the gasses to which the ammonia sensor is cross-sensitive. The difference in the response of capacitors on these areas of different porosity might then allow the effect of cross-sensitivity to be mathematically eliminated. Another option would be to borrow from existing technologies and use semi-permeable membranes that only allow the ammonia to pass through and not other gasses to which the device has cross-sensitivity.

Characterising the recovery time The interdigital capacitor test bench includes devices that use a resistive heater. While measurements with these devices have not been carried out, the hypothesis is that the resistive heater could be used to drive gas out of the pores periodically to improve the recovery time of the sensor. Configurations different than the one used in this work might also be considered. It would for instance be possible to have a separate heating electrode between the capacitor electrodes to limit resistive losses and parasitics in the sensing electrodes.



Appendices

A.1. MATLAB Simulation Code

This section lists the MATLAB code used to model and generate plots for the layered PASiC layer.

A.1.1. Plot Layers

```
function plotLayers(layerYs, epsilon, layerColours, ...
    layerColourIndices, plotBounds)
    %{
    FUNCTION NAME:
      plotLayers

    DESCRIPTION:
      Function to plot a multilayered structure with colours indicating
      their materials or their relative permittivity.

    INPUT:
      layerYs - A vector with the layer boundary heights in
      ascending order
      epsilon - (relative) permittivity of the layers around the
      specified layer_heights in ascending order. Length should be one
      greater than layerYs, as there are layers above and below
      the specified layer heights.
      layerColours - Colours for a few selected layers as a vector
      containing hexadecimal numbers (format #RRGGBB)
      layerColourIndices - index of the layers corresponding to the
      colours of layerColours. The colour of layers without specified
      colours for their indices will be given a weighted mix of the
      colours of the nearest indexed layers based on their permittivity.
      bounds - array with the boundary values to plot in the format
      [xMin, xMax; yMin, yMax];
    ASSUMPTIONS AND LIMITATIONS:
      layerHeights and epsilon should be in ascending order. Bounds must
      be outside layer heights

    AUTHOR:
      Jasper Rietveld
      Student number 4581881
    %}

    %% Input checks

    % Ensure that the layer number matches the number of entries in the
    % relative permittivity vectors
    layerCount = length(layerYs);
    if(length(epsilon) ~= layerCount+1)
        error("Number of entries in epsilon does not match " +...
            "layer_count + 1");
    end
```

```
% Ensure as much indices in layer_colour_indices as colours in
% layer_colours
if(length(layerColours) ~= ...
        length(layerColourIndices))
    error("Layer colour count does not match layer colour " + ...
        "index count")
end

% Ensure layer_colour_indices has unique indices
if(length(layerColourIndices) ~= ...
        length(unique(layerColourIndices)))
    error("Layer colour indices not unique")
end
% Ensure layer_colour_indices index count matches epsilon layer count
if(length(layerColourIndices)>length(epsilon))
    error("More indices in layer_colour_indices than layers in " + ...
        "epsilon")
end
% Ensure layer_colour_indices in range of epsilon layer count
if(~all(layerColourIndices > 0 & ...
        layerColourIndices <= length(epsilon)))
    error("Indices of layer_colour_indices out of range for " + ...
        "layers in epsilon")
end

% Ensure that bounds are of right format

% Ensure that bounds are in order

% Ensure that bounds surround layers

%% Input Processing

% Sort layer colours
[layerColourIndices, i_sorted] = sort(layerColourIndices);
layerColours = layerColours(i_sorted);

% Convert all colours to RGB [R, G, B] in the range 0 to 1 for
% interpolation
layerColoursRGB = zeros(length(layerColours), 4);
for i=1:length(layerColourIndices)
    if(length(layerColours(i)) <= 7)
       layerColours(i) = layerColours(i) + "FF";
    end
    layerColoursRGB(i,:) = ...
        sscanf(layerColours(i),'#%2x%2x%2x%2x',[1 4])/255;
    if(isempty(layerColoursRGB(i,:)))
        error("One of the colours specified is not a valid " + ...
            "hexadecimal format: #RRGGBB(AA)")
    end
end

% add bounds to layer_y
if(iscolumn(layerYs))
    layerYs = [plotBounds(2,1); layerYs; plotBounds(2,2)];
else
    layerYs = [plotBounds(2,1) layerYs plotBounds(2,2)];
end


%% Plot the different layers
% Create figure
hold on;

% Get epsilon of layers with indexed colour
colourEpsilon = epsilon(layerColourIndices);

% Set plot limits
xlim(plotBounds(1,:))
ylim(plotBounds(2,:));
```

```
    % Indices of surrounding layer colours
    interpIndices = [1 min(2, length(layerColourIndices))];
    for i=1:length(epsilon)

       % Get indices of next closest indexed colours
       if(i>=layerColourIndices(interpIndices(2)))
          interpIndices = min(interpIndices+1,...
              length(layerColourIndices));
          interpIndices(end)=min(interpIndices(end),...
              length(layerColourIndices));
       end
       % Interpolate colour between colours based on epsilon
       lerpfac = min(max(...
           diff([colourEpsilon(interpIndices(1)) epsilon(i)])/...
           diff(colourEpsilon(interpIndices)),0),1);
       layer_colour = [1-lerpfac lerpfac]*...
           layerColoursRGB(interpIndices,:);

       % Plot layer on graph
       plotLayer(plotBounds(1,1), layerYs(i), ...
           plotBounds(1,2)-plotBounds(1,1), ...
           layerYs(i+1)-layerYs(i), ...
           layer_colour);
    end


    % Helper function to plot layers
    function plotLayer(x, y, w, h, colour)
        xBox = [x, x, x+w, x+w, x];
        yBox = [y, y+h, y+h, y, y];
        p = patch(xBox, yBox, 'black', 'LineStyle', 'none', ...
            'FaceColor', colour(1:3));
        if(colour(4)<1)
            p.FaceVertexAlphaData = colour(4);
            p.FaceAlpha = 'flat';
            p.AlphaDataMapping = 'none';
        end
    end
end
```

A.1.2. Plot Layer Models

```
%{
SCRIPT NAME:
  plotLayerModels

DESCRIPTION:
  Function to draw the geometry for a certain layer configuration, along
  with thevalues of the porosity and the permittivity that follow from the
  layered model.

OUTPUT:
  figures are opened with the plotted data.

AUTHOR:
  Jasper Rietveld
  Student number 4581881
%}

close all;

%% Calculate permittivities for the the layered model

% Define model parameters

% depths are defined as distance from surface
dSiC = .5e-6;                             % Depth of entire SiC layer
dBuffer = .4e-6;                          % Depth at which porosity is zero
N = 15;                                   % Number of layers to use in model
layerDepths = linspace(0, dBuffer, N+1); % depth of SiC Layers for model
```

```
CNH3 = [0 100];                          % NH3 Concentrations to evaluate
NCNH3 = length(CNH3);                    % Number of concentrations

epsilonSub = 11.7;                       % Permittivity of the Si substrate
epsilonSiC = 9.7;                        % Permittivity of the SiC layer
epsilonAir = 1;                          % Permittivity of air

phiMax = 0.5;                            % Maximum porosity

phiSMmax = 0.5;                          % Porosity for maximum sensitivity
sMax = .2;                               % Sensitivity at this porosity

% Calculate model

[epsilon, porosity] = porosityModelLinear(layerDepths, CNH3, dSiC, ...
    dBuffer, epsilonSiC, epsilonAir, phiMax, sMax, phiSMmax);


%% plot the resulting distributions

% Add boundaries and permittivity for air and substrate layers
layerYs = [-dSiC linspace(-dBuffer, 0, N+1)];
epsilonLayers = [epsilonSub*ones(1,NCNH3);
                 flip(epsilon, 1);
                 ones(1,NCNH3)];

% add values at plot boundaries and duplicate values at layer boundaries
% to get straight lines at discontinuities in the plot
epsilonPlot = [repmat(epsilonSub,2,NCNH3);
               repmat(epsilonSiC,1,NCNH3);
               flip(epsilon, 1);
               ones(2,NCNH3)];
layerYsPlot = [-dSiC*2 -dSiC layerYs 0 dSiC];
porosityPlot = [0 0 0 flip(porosity) 0 0];

% Create a figure for the porosity plot
figure;
ax = axes;

% Define layer plotting inputs
layerColours = ["#BCBCBCCC", "#161626CC", "#C7E9FF"];
layerColourIndices = [1, 2, N+3];
plotBounds = [0, 1; -2*dSiC, dSiC];
% create figure for the permittivity plot
figure;
ax = axes;

% Define layer plotting inputs
layerColours = ["#BCBCBCCC", "#161626CC", "#C7E9FF"];
layerColourIndices = [1, 2, N+3];
plotBounds = [0, 1.1*max([(1+sMax)*epsilonSiC, epsilonSub, epsilonAir]);
              -2*dSiC, dSiC];

% plot the geometry of the model
plotLayers(layerYs, epsilonLayers(:,1), layerColours, ...
    layerColourIndices, plotBounds);

% plot the sensitivity as a line on the geometry
hold on;
h1 = plot(epsilonPlot(:,1), layerYsPlot,"black",'LineWidth', 2);
h2 = plot(epsilonPlot(:,2), layerYsPlot,"--black",'LineWidth', 2);
ylabel("height above surface [m]");
xlabel("relative permittivity");
title("Layer Geometry and Permittivity");
legend([h1, h2], {'Permittivity under air', 'Permittivity under NH3'});
ax.Layer = 'top';
% plot the geometry of the model
plotLayers(layerYs, epsilonLayers(:,1), layerColours, ...
    layerColourIndices, plotBounds);

% plot the porosity as a line on the geometry
```

```
hold on;
plot(porosityPlot, layerYsPlot,"black",'LineWidth',2);
ylabel("height above surface [m]");
xlabel("porosity");
title("Layer Geometry and Porosity Distribution");
ax.Layer = 'top';
```

A.1.3. Plot Configurations

```
%{
SCRIPT NAME:
  plotConfigurations

DESCRIPTION:
  Function to draw the geometry for a certain layer configuration. For two
  different sensitivities, the permittivity is plotted on these layers to
  allow for a comparison of the situations.

OUTPUT:
  figures are opened with the plotted data.

AUTHOR:
  Jasper Rietveld
  Student number 4581881
%}

close all;

%% Calculate permittivities for the the layered model

% Define model parameters

% depths are defined as distance from surface
dSiC = .5e-6;                           % Depth of entire SiC layer
dBuffer = .4e-6;                        % Depth at which porosity is zero
N = 15;                                 % Number of layers to use in model
layerDepths = linspace(0, dBuffer, N+1); % depth of SiC Layers for model

CNH3 = [0 100];                         % NH3 Concentrations to evaluate
NCNH3 = length(CNH3);                   % Number of concentrations

epsilonSub = 11.7;                      % Permittivity of the Si substrate
epsilonSiC = 9.7;                       % Permittivity of the SiC layer
epsilonAir = 1;                         % Permittivity of air

phiMax = 0.5;                           % Maximum porosity

phiSMmax = 0.5;                         % Porosity for maximum sensitivity
sMax = .2;                              % Sensitivity at this porosity

%% Calculate and plot permittivity model with small increase

[epsilon, porosity] = porosityModelLinear(layerDepths, CNH3, dSiC, ...
    dBuffer, epsilonSiC, epsilonAir, phiMax, sMax, phiSMmax);


% Add boundaries and permittivity for air and substrate layers
layerYs = [-dSiC linspace(-dBuffer, 0, N+1)];
epsilonLayers = [epsilonSub*ones(1,NCNH3);
                 flip(epsilon, 1);
                 ones(1,NCNH3)];

% add values at plot boundaries and duplicate values at layer boundaries
% to get straight lines at discontinuities in the plot
epsilonPlot = [repmat(epsilonSub,2,NCNH3);
               repmat(epsilonSiC,1,NCNH3);
               flip(epsilon, 1);
               ones(2,NCNH3)];
layerYsPlot = [-dSiC*2 -dSiC layerYs 0 dSiC];

% create figure for the permittivity plot
```

```
figure;
ax = axes;

% Define layer plotting inputs
layerColours = ["#BCBCBCCC", "#161626CC", "#C7E9FF"];
layerColourIndices = [1, 2, N+3];
plotBounds = [0, 1.1*max([(1+sMax)*epsilonSiC, epsilonSub, epsilonAir]);
              -2*dSiC, dSiC];

% plot the geometry of the model
plotLayers(layerYs, epsilonLayers(:,1), layerColours, ...
    layerColourIndices, plotBounds);

% plot the sensitivity as a line on the geometry
hold on;
h1 = plot(epsilonPlot(:,1), layerYsPlot,"black",'LineWidth', 2);
h2 = plot(epsilonPlot(:,2), layerYsPlot,"--black",'LineWidth', 2);
ylabel("height above surface [m]");
xlabel("relative permittivity");
title("Layer Geometry and Permittivity");
legend([h1, h2], {'Permittivity under air', 'Permittivity under NH3'});
ax.Layer = 'top';

%% Calculate and plot permittivity model with large increase

sMax = 0.8;

[epsilon, porosity] = porosityModelLinear(layerDepths, CNH3, dSiC, ...
    dBuffer, epsilonSiC, epsilonAir, phiMax, sMax, phiSMmax);


% Add boundaries and permittivity for air and substrate layers
layerYs = [-dSiC linspace(-dBuffer, 0, N+1)];
epsilonLayers = [epsilonSub*ones(1,NCNH3);
                 flip(epsilon, 1);
                 ones(1,NCNH3)];

% add values at plot boundaries and duplicate values at layer boundaries
% to get straight lines at discontinuities in the plot
epsilonPlot = [repmat(epsilonSub,2,NCNH3);
               repmat(epsilonSiC,1,NCNH3);
               flip(epsilon, 1);
               ones(2,NCNH3)];
layerYsPlot = [-dSiC*2 -dSiC layerYs 0 dSiC];

% create figure for the permittivity plot
figure;
ax = axes;

% Define layer plotting inputs
layerColours = ["#BCBCBCCC", "#161626CC", "#C7E9FF"];
layerColourIndices = [1, 2, N+3];
plotBounds = [0, 1.1*max([(1+sMax)*epsilonSiC, epsilonSub, epsilonAir]);
              -2*dSiC, dSiC];

% plot the geometry of the model
plotLayers(layerYs, epsilonLayers(:,1), layerColours, ...
    layerColourIndices, plotBounds);

% plot the sensitivity as a line on the geometry
hold on;
h1 = plot(epsilonPlot(:,1), layerYsPlot,"black",'LineWidth', 2);
h2 = plot(epsilonPlot(:,2), layerYsPlot,"--black",'LineWidth', 2);
ylabel("height above surface [m]");
xlabel("relative permittivity");
title("Layer Geometry and Permittivity");
legend([h1, h2], {'Permittivity under air', 'Permittivity under NH3'});
ax.Layer = 'top';
```

A.1.4. Plot Sensitivity

```
function plotSensitivity(simulationData)
    %{
    FUNCTION NAME:
      plotSensitivity

    DESCRIPTION:
      Processes a table with simulation data exported from COMSOL
      containing the simulated capacitance for different electrode
      dimensions. From the data, a plot is generated which calculates
      and displays the simulated sensitivity for the different
       dimensions.

    INPUT:
      The script requires the presence of an exported COMSOL table in the
      active workspace. This script is used as input to the program.

    OUTPUT:
      A figure is opened with the plotted data.

    AUTHOR:
      Jasper Rietveld
      Student number 4581881
    %}

    %% Pre-process input data
    simulationData = load(simulationData);

    % Add unique gap and width values to separate vectors
    gaps = unique(simulationData(1:2:end,1));
    widths = unique(simulationData(1:2:end,2));

    % Separate data of the capacitance per area for the different ammonia
    % concentration values
    capAir = simulationData(1:2:end,5);
    capNH3 = simulationData(2:2:end,5);

    % Calculate sensitivity as the percentage increase between the air and
    % ammonia capacitance values
    sensitivity = 100*capNH3./capAir - 100;

    % Restructure sensitivity to be a square matrix to allow it to be
    % plotted as a 2D image
    sensitivity = reshape(sensitivity, length(gaps), length(widths));

    %% Prepare plotting parameters

    % Define colours for the colour gradient
    colours = [255 255 255
        0 212 255
        0 52 135];

    % Interpolate the colours to form a gradient
    colour_points = 1:length(colours);

    steps = 100;
    gradientPoints = linspace(1,length(colours),steps);

    map = [interp1(colour_points, colours(:,1), gradientPoints)
        interp1(colour_points, colours(:,2), gradientPoints)
        interp1(colour_points, colours(:,3), gradientPoints)]';
    map = map./255;

    % Precalculate the min and max sensitivity values
    clims = [min(sensitivity, [], 'all') max(sensitivity, [], 'all')];

    %% Plot data
    figure;
    imagesc(sensitivity, clims);
    colormap(map)
    for i = 1:length(gaps)
        for j = 1:length(widths)
```

```
            % Format sensitivity as a percentage string with 3 digits
            sensitivityString = num2str(sensitivity(j,i),'%.3g');
            sensitivityString = strcat(sensitivityString,'%');

            % Calculate the value (lightness) of the gradient for this cell
            gradPos = 1+(length(colours)-1)*...
                (sensitivity(j,i)-clims(1))/diff(clims);
            gradColor = [interp1(colour_points, colours(:,1), gradPos)
                interp1(colour_points, colours(:,2), gradPos)
                interp1(colour_points, colours(:,3), gradPos)];
            gradColor = gradColor./255;

            % Get text colour that contrasts well with gradient
            textColor = getContrastColour(gradColor);

            % Place sensitivity on plot
            text(i, j, sensitivityString, 'Color', textColor, ...
                'FontSize', 12, 'HorizontalAlignment', 'center');
        end
    end

    %% Add title, labels and format axis to figure
    axis square
    title('Sensitivity for Different Electrode Dimensions')
    xticks(1:length(widths));
    xlabel('Electrode Width');
    xticklabels(cellstr(num2str(widths)))
    yticks(1:length(gaps));
    ylabel('Electrode Gap');
    yticklabels(cellstr(num2str(gaps)))
end


%% Helper functions

function contrastColour = getContrastColour(colour)
    %{
    FUNCTION NAME:
      contrastColour

    DESCRIPTION:
      Helper function to determine if black or white has more visual
      contrast against a certain colour.

    INPUT:
      colour - background colour for the comparison

    OUTPUT:
      contrastColour - a 3 component RGB vector with values [0-1].
      Either black or white depending on the contrast

    ASSUMPTIONS AND LIMITATIONS:
      The partial capacitance tecnhique rewuires that the values of the
      relative permittivity are either monotonically increasing or
      monotonically decreasing in the directions away from electrodes

    REFERENCES:
      The math behind calculating the contrast ratio for determining
      what colour best contrasts the background colour comes from the
      Web Content Accessibility Guidelines (WCAG) 2.0 at
      https://www.w3.org/TR/2008/REC-WCAG20-20081211/#contrastRatio

    AUTHOR:
      Jasper Rietveld
      Student number 4581881
    %}

    linColour = (colour > 0.03928).*((colour+0.055)./1.055).^2.4;
    linColour = linColour + (colour < 0.03928).*colour./12.92;
    relativeLuminance = dot([0.2126 0.7152 0.0722], linColour);
```

```
    contrastRatioWhite = 1.05 / (relativeLuminance + 0.05);
    contrastRatioBlack = (relativeLuminance + 0.05) / 0.05;

    if(contrastRatioWhite > contrastRatioBlack)
       contrastColour = [1 1 1];
    else
       contrastColour = [0 0 0];
    end
end
```

A.1.5. Model Sensitivity

```
%{
SCRIPT NAME:
  modelSensitivity

DESCRIPTION:
  Creates tables for the comparison of the analytical model for
  interdigital electrodes within a multilayered structure with COMSOL
  simulations of the same configuration. A comparison is made between the
  resulting capacitance values and the sensitivity calculated from them,
  and the size of precision errors for the analytical model is compared to
  the total capacitance value.

INPUT:
  The script requires the presence of an exported COMSOL table in the
  active workspace, as well as the presence of a valid LaTeX table
  template.

OUTPUT:
  The LaTeX for a series of tables with the comparison is printed to the
  command window for copying into the report

AUTHOR:
  Jasper Rietveld
  Student number 4581881
%}

%% Calculate permittivities for the the layered model

% Define model parameters

% depths are defined as distance from surface
dSiC = .5e-6;                               % Depth of entire SiC layer
dBuffer = .4e-6;                            % Depth at which porosity is zero
N = 10;                                     % Number of layers to use in model
layerDepths = linspace(0, dBuffer, N+1); % depth of SiC Layers for model

capNH3 = [0 100];                           % NH3 Concentrations to evaluate
NCNH3 = length(capNH3);                     % Number of concentrations

epsilonSub = 11.7;                          % Permittivity of the Si substrate
epsilonSiC = 9.7;                           % Permittivity of the SiC layer
epsilonAir = 1;                             % Permittivity of air

phiMax = 0.5;                               % Maximum porosity

phiSMmax = 0.5;                             % Porosity for maximum sensitivity
sMax = 10*epsilonSiC;                       % Sensitivity at this porosity

% Calculate model

[epsilon, porosity] = porosityModelLinear(layerDepths, capNH3, dSiC, ...
    dBuffer, epsilonSiC, epsilonAir, phiMax, sMax, phiSMmax);

%% Apply analytical capacitance estimation model for range of widths/gaps

% Epsilon is altered slightly to ensure a monotonically decreasing lower
% domain in the case of a high ammonia concentration.
epsilon(epsilon(:,2) < epsilonSub, 2) = epsilonSub;
```

```
% Add boundaries and permittivity for air and substrate layers
layerYs = [-dSiC linspace(-dBuffer, 0, N+1)]'; % y=0 at the SiC surface
epsilonLayers = [epsilonSub*ones(1,NCNH3);
                flip(epsilon, 1); ones(1,NCNH3)];
electrodeIndex = N+2; % Electrodes on top of SiC

% Create vectors with the widths and heights to be tested
widths = [1 2 5]*1e-6;
gaps = [1 2 5]*1e-6;

% Apply the analytical model for the cases with and without ammonia present
er = epsilonLayers(:,1);
[capAir, errAir] = estimateCapacitance(layerYs, electrodeIndex, ...
    er, gaps, widths);

er = epsilonLayers(:,2);
[capNH3, errNH3] = estimateCapacitance(layerYs, electrodeIndex, ...
    er, gaps, widths);

%% Load simulation data

simulationData = load('updated_paSiC_sweep_10x.txt');

% Add unique gap and width values to separate vectors
gaps = unique(simulationData(1:2:end,1));
widths = unique(simulationData(1:2:end,2));

% Separate data of the capacitance per area for the different ammonia
% concentration values
capAirSim = simulationData(1:2:end,5);
capNH3Sim = simulationData(2:2:end,5);

capAirSim = reshape(capAirSim, length(gaps), length(widths)); % per CM2
capNH3Sim = reshape(capNH3Sim, length(gaps), length(widths)); % per CM2

capAirSim = capAirSim(1:3,1:3);
capNH3Sim = capNH3Sim(1:3,1:3);

%% Output data formatted as LaTeX tabulars

% Value tables data
disp("capacitor size table air: ");
generateTableString(...
    widths, "%.2g", "[\si{\micro\meter}]",...
    gaps, "%.2g", "[\si{\micro\meter}]",...
    capAir./1e-9, "\SI{%.3g}{\nano\farad}",...
    capAirSim./1e-9, "{\color{tudelft-green}\SI{%.3g}{\nano\farad}}",...
    capAir./capAirSim, "{\color{tudelft-warm-purple}(%.3g)}");
disp("");

disp("capacitor size table NH3: ");
generateTableString(...
    widths, "%.2g", "[\si{\micro\meter}]",...
    gaps, "%.2g", "[\si{\micro\meter}]",...
    capNH3./1e-9, "\SI{%.3g}{\nano\farad}",...
    capNH3Sim./1e-9, "{\color{tudelft-green}\SI{%.3g}{\nano\farad}}",...
    capNH3./capNH3Sim, "{\color{tudelft-warm-purple}(%.3g)}");
disp("");

% Error tables data
disp("error table air: ");
generateTableString(...
    widths, "%.2g", "[\si{\micro\meter}]",...
    gaps, "%.2g", "[\si{\micro\meter}]",...
    errAir./1e-9, "\SI{%.3g}{\nano\farad}",...
    capAir./1e-9, "{\color{tudelft-green}\SI{%.3g}{\nano\farad}}",...
    errAir./capAir, "{\color{tudelft-warm-purple}(%.3g)}");
disp("");

disp("error table NH3: ");
generateTableString(...
```

```
    widths, "%.2g", "[\si{\micro\meter}]",...
    gaps, "%.2g", "[\si{\micro\meter}]",...
    errNH3./1e-9, "\SI{%.3g}{\nano\farad}",...
    capNH3./1e-9, "{\color{tudelft-green}\SI{%.3g}{\nano\farad}}",...
    errNH3./capNH3, "{\color{tudelft-warm-purple}(%.3g)}");
disp("");


% Sensitivity tables data
disp("sensitivity table: ");
generateTableString(...
    widths, "%.2g", "[\si{\micro\meter}]",...
    gaps, "%.2g", "[\si{\micro\meter}]",...
    round(capNH3./capAir*100,-1), "%.0f\%",...
    round(capNH3Sim./capAirSim*100,-1), "{\color{tudelft-green}%.3g\%}",...
    (capNH3./capAir*100)./(capNH3Sim./capAirSim*100),...
        "{\color{tudelft-warm-purple}(%.3g)}");
disp("");


function generateTableString(widths, widthformat, widthunitformat, ...
    gaps, gapformat, gapunitformat, mainval, mainvalformat,...
    subval1, subval1format, subval2, subval2format)
    %{
    FUNCTION NAME:
      generateTableString

    DESCRIPTION:
      Fills a template table with values and formats for these values
      passed as function parameters.

    INPUT:
      widths - A vector with the widths.
      widthformat - LaTeX string containing a formatSpec
      widthunitformat - LaTeX string to put after the column heading
      gaps - A vector with the gaps
      gapformat - LaTeX string containing a formatSpec
      gapunitformat - LaTeX string to put after the row heading
      mainval - matrix for the main value to insert for each W and G
      mainvalformat - LaTeX string with a formatSpec for the main value
      subval1 - matrix for a second value to insert for each W and G
      subval1format - LaTeX string with a formatSpec for the second value
      subval2 - matrix for a third value to insert for each W and G
      subval2format - LaTeX string with a formatSpec for the third value

    OUTPUT:
      The LaTeX for the table with the values inserted is printed to
      the command window for copying into the report

    ASSUMPTIONS AND LIMITATIONS:
      The script requires the presence of an exported COMSOL table in the
      active workspace, as well as the presence of a valid LaTeX table
      template. Format strings should contain one formatSpec. LaTeX can be
      used in the format strings.

    AUTHOR:
      Jasper Rietveld
      Student number 4581881
    %}

    % Read file with the LaTeX format to insert the values into
    tableFormat = readlines("tableFormat.txt");

    % Insert unit strings
    tableFormat = strrep(tableFormat, "widthformat", widthformat);
    tableFormat = strrep(tableFormat, "widthunitformat", widthunitformat);
    tableFormat = strrep(tableFormat, "gapformat", gapformat);
    tableFormat = strrep(tableFormat, "gapunitformat", gapunitformat);
    tableFormat = strrep(tableFormat, "mainvalformat", mainvalformat);
    tableFormat = strrep(tableFormat, "subval1format", subval1format);
    tableFormat = strrep(tableFormat, "subval2format", subval2format);
```

```
    % Replace % with %% to allow printing with fprintf
    tableFormat = strrep(tableFormat, "\%", "\%%");

    % Replace \ with \\ to allow printing with fprintf
    tableFormat = strrep(tableFormat, "\", "\\");


    % Add newlines to allow printing with fprintf
    tableFormat = strjoin(tableFormat, "\n");
    tableFormat = strcat(tableFormat, "\n");

    % Insert values
    fprintf(tableFormat, ...
        widths(1), widths(2), widths(3),...
        gaps(1), mainval(1,1), mainval(1,2), mainval(1,3),...
        subval1(1,1), subval2(1,1), subval1(1,2),...
            subval2(1,2), subval1(1,3), subval2(1,3),...
        gaps(2), mainval(2,1), mainval(2,2), mainval(2,3),...
        subval1(2,1), subval2(2,1), subval1(2,2),...
            subval2(2,2), subval1(2,3), subval2(2,3),...
        gaps(3), mainval(3,1), mainval(3,2), mainval(3,3),...
        subval1(3,1), subval2(3,1), subval1(3,2),...
            subval2(3,2), subval1(3,3), subval2(3,3)...
        );
end
```

A.1.6. Evaluate Capacitance

```
function C = evaluateCapacitance(layerYs, electrodeIndex, ...
    er, lambdaElectrodes, etaElectrodes)
    %{
    FUNCTION NAME:
      evaluate_capacitance

    DESCRIPTION:
      Function to estimate the sensitivity of the capacitance of an
      interdigital electrode array within a multilayered strucure of
      materials with varying relative permittivity

    INPUT:
      layerY - A vector with the layer boundary heights from bottom to
      top
      electrodeIndex - index of the layer boundary at which the
      electrodes are located.
      er - A vector that describes the relative permittivity in each
      of the layers, from bottom to top.
      lambdaElectrodes - A vector with different values of the spatial
      wavelength 2(W+G) of the electrodes to be evaluated
      etaElectrodes -  A vector with different values of the ratio
      W/(W+G) of the electrodes to be evaluated

    OUTPUT:
      C - Matrix with the resulting capacitances using the initial
      values. Eta changes in the direction of dimension 1, lambda changes
      in the direction of dimension 2

    ASSUMPTIONS AND LIMITATIONS:
      The partial capacitance tecnhique rewuires that the values of the
      relative permittivity are either monotonically increasing or
      monotonically decreasing in the directions away from electrodes

    REFERENCES:
      TODO

    AUTHOR:
      Jasper Rietveld
      Student number 4581881
    %}

    %% Input checks
```

```
% Ensure electrode_index is within bounds
if(electrodeIndex < 1 || electrodeIndex > length(layerYs))
    error("electrodeHeight is out of layerHeight's bounds");
end

% Ensure that layer_y is in order
layerYs = sort(layerYs);

% Ensure that the layer number matches the number of entries in the
% relative permittivity vectors
layerCount = length(layerYs);
if(length(er) ~= layerCount+1)
    error("Number of entries in er does not match " +...
        "layerCount + 1");
end

% Ensure epsilon either monotonically increasing or monotonically
% decreasing away from electrode
if(~issorted(er(1:electrodeIndex),'ascend') && ...
    ~issorted(er(1:electrodeIndex),'descend'))
    error("Relative permittivity in lower halfplane not " + ...
    "monotonically increasing or decreasing in er");
elseif(~issorted(er(electrodeIndex+1:end),'ascend') && ...
    ~issorted(er(electrodeIndex+1:end),'descend'))
    error("Relative permittivity in upper halfplane not " + ...
    "monotonically increasing or decreasing in er");
end

%% Main function body

e0 = 8.854187817e-12;

layerYsUpper = layerYs(electrodeIndex+1:end) - ...
    layerYs(electrodeIndex);
erUpper = er(electrodeIndex+1:end);
ChUpper = evaluateHalfplane(layerYsUpper, e0*erUpper, ...
        lambdaElectrodes, etaElectrodes);
layerYsLower = flip(layerYs(electrodeIndex) - ...
    layerYs(1:electrodeIndex));
erLower = flip(er(1:electrodeIndex));
ChLower = evaluateHalfplane(layerYsLower, e0*erLower, ...
        lambdaElectrodes, etaElectrodes);


C = ChUpper + ChLower;

%% Helper Functions

function Ch = evaluateHalfplane(layerYs, epsilon, ...
        lambdaElectrodes, etaElectrodes)
    % Calculate the capacitances in one of the halfplanes

    % Calculate k and ellip(k) for all lambda values
    k = zeros(length(lambdaElectrodes), length(layerYs));
    K = zeros(size(k));
    for i=1:length(lambdaElectrodes)
        for j = 1:length(layerYs)
            h = layerYs(j);
            % Calculate layer height
            r=h/lambdaElectrodes(i);

            q = exp(-4*pi*r);

            v2 = JacobiTheta2(0, q);
            v3 = JacobiTheta3(0, q);
            k(i) = (v2/v3)^2; % v2 v3 Jacobi theta functions
            K(i) = ellipk(k(i));
        end
    end
```

```
% Calculate contribution capacitance of infinite domain for all
% values of eta
Cinf = zeros(length(etaElectrodes),1);
for i=1:length(etaElectrodes)
    kIinf = sin((pi/2)*etaElectrodes(i));

    kIinfprime = sqrt(1-kIinf^2);

    % ellipk jacobi elliptic function
    Cinf(i) = epsilon(end)*ellipk(kIinf)/ellipk(kIinfprime);
end

% Calculate the total capacitance for all combinations of eta and
% lambda


if(length(epsilon)==1)
    Ch = Cinf;
    return
end
% Determine if epsilon is monotonically increasing or decreasing
decreasing = epsilon(1) > epsilon(end);

if(decreasing)
    Ch = Cinf * ones(length(etaElectrodes),length(lambdaElectrodes));
    for i=1:length(etaElectrodes)
        for j=1:length(lambdaElectrodes)
            for h = 1:length(epsilon)-1
                % Calculate layer partial capacitance contribution
                t4 = 1/k(j);

                % sn Jacobi elliptic function
                [t2, ~, ~] = ellipj(K(j)*etaElectrodes(i), k(j));

                % calculate kI based on epsilon increment or decrement
                kI = t2*sqrt((t4^2-1)/(t4^2-t2^2));

                kIprime = sqrt(1 - kI^2);

                Cl = (epsilon(h) - epsilon(h+1));
                Cl = Cl*ellipk(kI)/ellipk(kIprime);

                Ch(i,j) = Ch(i,j) + Cl;
            end
        end
    end
else
    Ch = 1/Cinf * ones(length(etaElectrodes),length(lambdaElectrodes));
    for i=1:length(etaElectrodes)
        for j=1:length(lambdaElectrodes)
            for h = 1:length(epsilon)-1
                % Calculate layer partial capacitance contribution
                t4 = 1/k(j);

                % sn Jacobi elliptic function
                [t2, ~, ~] = ellipj(K(j)*etaElectrodes(i), k(j));

                % calculate kI based on epsilon increment or decrement
                kI = t2;

                kIprime = sqrt(1 - kI^2);

                Cl = (1/epsilon(h) - 1/epsilon(h+1));
                Cl = Cl*1/(ellipk(kI)/ellipk(kIprime));
                Cl = 1/Cl;

                Ch(i,j) = Ch(i,j) + Cl;
            end
        end
    end
    Ch = 1/Ch;
```

```
        end
    end
end
```

A.1.7. Plot Precision Errors

```
%{
SCRIPT NAME:
  plotPrecisionErrors

DESCRIPTION:
  Function to generate a plot displaying issues concerning precision errors
  when applying the analytical layered capacitance model.

OUTPUT:
  A figure is opened with the plotted data.

REFERENCES:
  Based on model from by Rui Igreja, C.J. Dias, accessed at
  https://doi.org/10.1016/j.sna.2011.09.033

  Uses JacobiTheta2 and JacobiTheta3 from the elfun18 MATLAB add-on by
  milan batista (2022).
  https://www.mathworks.com/matlabcentral/fileexchange/65915-elfun18

AUTHOR:
  Jasper Rietveld
  Student number 4581881
%}

% Create a range of values for the layer adimensional thickness r
r = logspace(-4, 2, 100);

% Calcualte the resulting factor used in the calculation of the
% corresponding layer capacitance for different values of eta

eta = [0.2 0.4 0.6 0.8 0.9];

q = exp(-4*pi*r);

k = (JacobiTheta2(0, q)./JacobiTheta3(0, q)).^2;

K = ellipke(k);

[t2, ~, ~] = ellipj(K'*eta, k'*ones(size(eta)));

t4 = 1./(k'*ones(size(eta)));

kI = sqrt((t4.^2-1)./(t4.^2-t2.^2));

kIPrime = sqrt(1-kI.^2);

fac = ellipke(kI)./ellipke(kIPrime);

%% Plot the results

% Create logarithmic plot with the value fac for values of r and eta
plot(r, fac);
grid on;
handle = gca;
set(handle,'xscale','log')

% Add title and labels
title("K(k_{IP})/K(k'_{IP}) vs r for different values of \eta");
xlabel('adimensional layer thickness r = h/\lambda');
ylabel("K(k_{IP})/K(k'_{IP})");

% Replace colour gradient for the plot lines

% Define colours for the colour gradient
colours = [255 255 255
```

```
    0 144 255
    0 52 135];

% Interpolate the colours to form a gradient
colour_points = 1:length(colours);

steps = length(eta)+2; % Use extra values as the gradient includes white
gradientPoints = linspace(1,length(colours),steps);

map = [interp1(colour_points, colours(:,1), gradientPoints)
    interp1(colour_points, colours(:,2), gradientPoints)
    interp1(colour_points, colours(:,3), gradientPoints)]';
map = map./255;

colororder(map(3:end,:)); % Do not use the lighter values

% Add legend with values of eta
legendString = repmat("\eta = ", 1, 5);
legendString = strcat(legendString, compose("%.2f",eta));
legend(legendString, 'location', 'northwest');
```

A.2. COMSOL Simulation Code

In this section, the code used to carry out the COMSOL simulations is documented. The COMSOL API is based on the Java programming language.

A.2.1. Generate Pores

```
/**
 * Generate Pores
 *
 * Method to add pore geometry to a shape based on a distribution
 *
 * @param inputGeom     geometry bounding the area where pores should be placed
 * @param targetGeom    geometry to subtract the pores from
 * @param gasGeom       geometry of the gas to join the pores with
 *
 * @param poreNum       number of added pores
 * @param poreRMin      minimal pore radius
 * @param poreRMax      maximal pore radius
 *
 * @param distMaxDepth  maximum depth in the pore placement distribution
 * @param distExpConst  exponential falloff constant for the distribution
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */


/* variable definitions */
int ind = 0;
double gx, gy, gw, gh; // Input geometry dimensions
double px, py, pr = 0.0; // Pore dimensions

/* Get input geometry bounding box dimensions (assumed rectangular) */
gx = model.component("comp1").geom("geom1").feature(inputGeom).getDoubleArray("pos")[0];
gy = model.component("comp1").geom("geom1").feature(inputGeom).getDoubleArray("pos")[1];
gw = model.component("comp1").geom("geom1").feature(inputGeom).getDoubleArray("size")[0];
gh = model.component("comp1").geom("geom1").feature(inputGeom).getDoubleArray("size")[1];

/* Create selection to hold all pores */
model.component("comp1").geom("geom1").selection().create("csel2", "CumulativeSelection");

/* Create pores */
while (ind < poreNum) {
  /* Choose random pore position within input geometry bounding box */
  px = gx+Math.random()*gw;

  /* choose random pore size */
  pr = Math.random()*(poreRMax-poreRMin)+poreRMin;
```

```
  /* Generate uniformly random y pos above electrodes */
  py = gy+gh-Math.random()*distMaxDepth;

  /* Generate y coordinate according to quasi-negative exponential distribution */
  double exp = 1-Math.random()*(1-Math.exp(-distExpConst*distMaxDepth));
  exp = -1/(distExpConst)*Math.log(exp);
  py = gy+gh-exp;

  /* Create geometry for the pore and add it to the selection */
  model.component("comp1").geom("geom1").create("c"+ind, "Circle");
  model.component("comp1").geom("geom1").feature("c"+ind).set("r", pr);
  model.component("comp1").geom("geom1").feature("c"+ind).set("pos", new double[]{px, py});
  model.component("comp1").geom("geom1").feature("c"+ind).set("contributeto", "csel2");
  ind++;
}

/* Create union of pores */
model.component("comp1").geom("geom1").create("uni1", "Union");
model.component("comp1").geom("geom1").feature("uni1").selection("input").named("csel2");
model.component("comp1").geom("geom1").feature("uni1").set("intbnd", "off");

/* Remove portion of pores outside the target */
model.component("comp1").geom("geom1").create("int1", "Intersection");
model.component("comp1").geom("geom1").feature("int1").selection("input").set(targetGeom);
model.component("comp1").geom("geom1").feature("int1").selection("input").add("uni1");
model.component("comp1").geom("geom1").feature("int1").set("keep", "on");

/* Subtract pores from input geometry */
model.component("comp1").geom("geom1").create("dif2", "Difference");
model.component("comp1").geom("geom1").feature("dif2").selection("input").set(targetGeom);
model.component("comp1").geom("geom1").feature("dif2").selection("input2").set("uni1");
model.component("comp1").geom("geom1").feature("dif2").set("keep", "off");

/* Add pores to gas layer */
model.component("comp1").geom("geom1").create("uni2", "Union");
model.component("comp1").geom("geom1").feature("uni2").selection("input").set(gasGeom);
model.component("comp1").geom("geom1").feature("uni2").selection("input").add("int1");
model.component("comp1").geom("geom1").feature("uni2").set("contributeto", "csel1");
model.component("comp1").geom("geom1").feature("uni2").set("intbnd", "off");

/* Build the geometry */
model.component("comp1").geom("geom1").run();
```

A.2.2. Porosity Model

```
/**
 * Porosity Model
 *
 * Method to evaluate the porosity model at a certain depth.
 *
 * @param d                 depth at which to evaluate the model
 * @param concNH3           NH3 concentration at which to evaluate the model
 *
 * @param sicHeight         height of the SiC layer
 * @param bufferHeight      height of the SiC buffer where there is no porosity
 *
 * @param sicEpsilon        (relative) permittivity of the unetched SiC
 * @param airEpsilon        (relative) permittivity of air
 *
 * @param phiMax            maximal porosity of the paSiC
 * @param sensMax           maximal sensitivity of the paSiC, described as the change
 *                          in epsilon at the maximal NH3 concentration
 * @param phiSensMax        porosity at which the sensitivity of the paSiC is maximal
 *
 * @param exponentialModel  whether the exponential model should be used
 * @param cFalloff          falloff constant used if the exponential model is used
 *
 * @return epsilon          value of epsilon based on the NH3 concentration
 *
 *
```

```
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Determine the porosity based on the depth */
double phi = 0;
double bufferDepth = sicHeight-bufferHeight;
if (d >= 0 && d <= bufferDepth) {
  if (!exponentialModel) {
    phi = phiMax*(1-d/bufferDepth); // linear falloff with depth
  } else {
    phi = phiMax*(Math.exp(-(d/bufferDepth)*cFalloff)); // exponential falloff with depth
  }
}
message("phi: "+phi);

/* Determine the sensitivity based on the porosity */
double sens = 0;
if (0 <= phi && phi < phiSensMax) {
  sens = phi/phiSensMax*sensMax;
}
else {
  sens = (1-phi)/(1-phiSensMax)*sensMax;
}

/* determine the increase of the permittivity based on the NH3 concentration */
double nh3Epsilon = concNH3*sens/100;

/* interpolate between the permittivity of SiC and air based on the porosity */
epsilon = (1-phi)*sicEpsilon+phi*airEpsilon+nh3Epsilon;
```

A.2.3. Porosity Model Expression

```
/**
 * Porosity Model Expression
 *
 * Method to evaluate the porosity model at a certain depth as a string expression.
 *
 * @param d                    depth at which to evaluate the model
 * @param concNH3              NH3 concentration at which to evaluate the model
 *
 * @param sicHeight            height of the SiC layer
 * @param bufferHeight         height of the SiC buffer where there is no porosity
 *
 * @param sicEpsilon           (relative) permittivity of the unetched SiC
 * @param airEpsilon           (relative) permittivity of air
 *
 * @param phiMax               maximal porosity of the paSiC
 * @param sensMax              maximal sensitivity of the paSiC, described as the change
 *                             in epsilon at the maximal NH3 concentration
 * @param phiSensMax           porosity at which the sensitivity of the paSiC is maximal
 *
 * @param exponentialModel     whether the exponential model should be used
 * @param cFalloff             falloff constant used if the exponential model is used
 *
 * @return epsilon             Expression in string form for epsilon based on the NH3
 *                             concentration
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Determine the porosity based on the depth */
double phi = 0;
double bufferDepth = sicHeight-bufferHeight;
if (d >= 0 && d <= bufferDepth) {
  if (!exponentialModel) {
    phi = phiMax*(1-d/bufferDepth); // linear falloff with depth
  } else {
    phi = phiMax*(Math.exp(-(d/bufferDepth)*cFalloff)); // exponential falloff with depth
  }
```

```
}
message("phi: "+phi);

/* Determine the sensitivity based on the porosity */
double sens = 0;
if (0 <= phi && phi < phiSensMax) {
  sens = phi/phiSensMax*sensMax;
}
else {
  sens = (1-phi)/(1-phiSensMax)*sensMax;
}

/* determine the increase of the permittivity based on the NH3 concentration */
String nh3Epsilon = concParam+"*"+sens+"/100";

/* interpolate between the permittivity of SiC and air based on the porosity */
epsilon = ((1-phi)*sicEpsilon+phi*airEpsilon)+"+"+nh3Epsilon;
```

A.2.4. Apply Porosity Model

```
/**
 * Apply Porosity Model
 *
 * Method to set the material properties of the layers in the geometry according to the
 * porosity model.
 *
 * @param concParam         name of parameter in the model representing the NH3 concentration
 * @param ySurface          y position at which the surface of the SiC starts
 * @param xSurface          x position at the centre of the surface of the SiC
 *
 * @param exponentialModel  Use the exponential model instead of the linear model
 *
 * @param baseMaterial      name of the material in the model the layers should be based on
 *
 * @param colourSiC         RGB colour values of the SiC material
 * @param colourAir         RGB colour values of the air material
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Geometry parameters */
double bufferHeight = model.param().evaluate("D_buff");
double sicWidth = model.param().evaluate("W_sim");
double sicHeight = model.param().evaluate("D_sic");

/* Layer geometry parameters */
int layerCount = (int) Math.round(model.param().evaluate("Count"));
double layerHeight = (sicHeight-bufferHeight)/layerCount;

/* Get porosity parameters from the COMSOL model */
double sicEpsilon = model.param().evaluate("sicEpsilon");
double airEpsilon = model.param().evaluate("airEpsilon");
double phiMax = model.param().evaluate("phiMax");
double sensMax = model.param().evaluate("sensMax");
double phiSensMax = model.param().evaluate("phiSensMax");
double cFalloff = model.param().evaluate("cFalloff");

/* Loop to assign material to layer domains */
for (int i = 0; i < layerCount; i++) {
  /* determine depth for this layer */
  double d = layerHeight*i;

  /*
   * Get an expression for the epsilon of the material according to the porosity model
   * (The top of layer is sampled)
   */
  String layerEpsilonExpression = porosityModelExpression(d, concParam, sicHeight,
                                                         bufferHeight, sicEpsilon,
                                                         airEpsilon, phiMax, sensMax,
                                                         phiSensMax, exponentialModel,
```

```
                                                          cFalloff);

  /* Get the base epsilon value to lerp between the material colours */
  double airLayerEpsilon = porosityModel(d, 0, sicHeight, bufferHeight, sicEpsilon,
                                         airEpsilon, phiMax, sensMax, phiSensMax,
                                         exponentialModel, cFalloff);
  double lerpfac = Math.min(Math.max((airLayerEpsilon-sicEpsilon)/(airEpsilon-sicEpsilon),
                                     0), 1);
  double[] layerColour = new double[]{0, 0, 0};
  for (int j = 0; j < 3; j++) {
    layerColour[j] = colourSiC[j]*(1.0d-lerpfac)+colourAir[j]*lerpfac;
  }
  message(layerColour[0]+", "+layerColour[1]+", "+layerColour[2]+" ~ "+lerpfac);

  /* Create material for layer, based on SiC model used */
  model.component("comp1").material().duplicate("mat_layer_"+i, baseMaterial);
  model.component("comp1").material("mat_layer_"+i).label("Porous Silicon Carbide "+i);

  /* Set the permittivity to the expression resulting from the model */
  with(model.component("comp1").material("mat_layer_"+i).propertyGroup("def"));
    set("relpermittivity", new String[]{layerEpsilonExpression});
  endwith();

  /* Set the material colour to the colour resulting from the model */
  with(model.component("comp1").material("mat_layer_"+i));
    set("family", "custom");
    set("customdiffuse", layerColour);
  endwith();

  /* Select domain of the layer using a disk selection */
  model.component("comp1").geom("geom1").run("fin");
  model.component("comp1").geom("geom1").create("disksel"+i, "DiskSelection");
  with(model.component("comp1").geom("geom1").feature("disksel"+i));
    set("r", Math.min(layerHeight, bufferHeight)*.25d);
    set("posx", xSurface+sicWidth*.5d);
    set("posy", ySurface-layerHeight*i-layerHeight*.5d);
  endwith();

  /* Apply the created material to the layer */
  model.component("comp1").material("mat_layer_"+i).selection().named("geom1_disksel"+i);
}

/* Rebuild Geometry */
model.component("comp1").geom("geom1").run();
```

A.2.5. Remove Layer Data

```
/**
 * Remove Layer Data
 *
 * Method to undo the material assignments from the porosity model to allow replacing the
 * materials.
 *
 * @param layerCount  number of layers in the model to clear
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* loop over the layers */
for (int i = 0; i < layerCount; i++) {
  /* try removing the material of the layer */
  try {
    model.component("comp1").material().remove("mat_layer_"+i);
  }
  catch (Exception e) {
    message("Error when deleting mat_layer_"+i+": "+e.toString());
  }

  /* try removing the selection that was used to assign the material to the layer */
  try {
```

```
    model.component("comp1").geom("geom1").feature().remove("disksel"+i);
  }
  catch (Exception e) {
    message("Error when deleting disksel"+i+": "+e.toString());
  }
}
```

A.3. Mask Capacitor Dimensions

A.4. Capacitor Reference Tables

This appendix contains the tables documenting the dimensions and locations on the wafer of the different capacitors included on the mask. Figure A.1 can be used as a reference for the coordinates of the capacitors

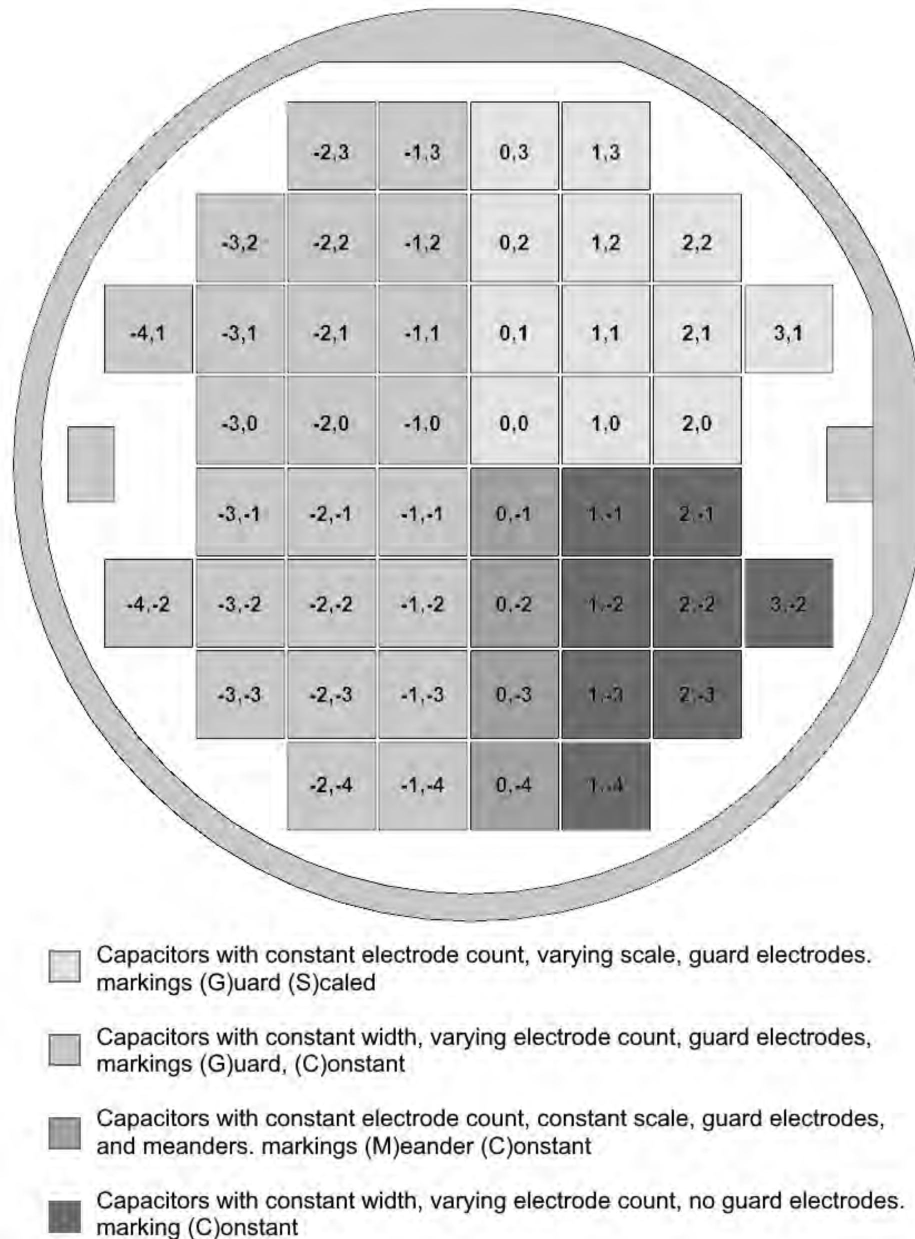


Figure A.1: Overview of a complete mask for the wafer.

Table A.1: Capacitors with constant width, varying electrode count, guard electrodes, markings (G)uard, (C)onstant

Width (μm)	Gap (μm)	Sensing area[1] (mm^2)	Electrode counts [2]	Peripheral scale [3]	Expected capacitance [4]	copies	Placement on wafer [5]
1	1	0.01	25/12	0.05	2.21E-13	6	-1,0,0; -2,1,0; -3,1,0; -3,-1,3; -2,-3,3; -2,-4,3
1	2	0.0096	16/8	0.05	1.01E-13	6	-1,0,1; -2,1,1; -3,1,1; -3,-1,2; -2,-3,2; -2,-4,2
2	1	0.0096	16/8	0.05	1.66E-13	6	-1,0,2; -2,1,2; -3,1,2; -3,-1,1; -2,-3,1; -2,-4,1
2	2	0.0096	12/6	0.05	8.93E-14	6	-1,0,3; -2,1,3; -3,1,3; -3,-1,0; -2,-3,0; -2,-4,0
2	5	0.1568	28/14	0.2	5.41E-13	6	-1,1,0; -3,0,0; -2,3,0; -1,-2,3; -3,-2,3; -3,-3,3
5	2	0.1568	28/14	0.2	1.05E-12	6	-1,1,1; -3,0,1; -2,3,1; -1,-2,2; -3,-2,2; -3,-3,2
5	5	0.16	20/10	0.2	4.93E-13	6	-1,1,2; -3,0,2; -2,3,2; -1,-2,1; -3,-2,1; -3,-3,1
5	10	0.156	13/6	0.2	2.26E-13	6	-1,1,3; -3,0,3; -2,3,3; -1,-2,0; -3,-2,0; -3,-3,0
10	5	0.156	13/6	0.2	3.71E-13	6	-2,0,0; -1,3,0; -3,2,0; -2,-1,3; -1,-3,3; -4,-2,3
10	10	0.16	10/5	0.2	2.16E-13	6	-2,0,1; -1,3,1; -3,2,1; -2,-1,2; -1,-3,2; -4,-2,2
10	20	3.96	33/16	1	2.73E-12	6	-2,0,2; -1,3,2; -3,2,2; -2,-1,1; -1,-3,1; -4,-2,1
20	10	3.96	33/16	1	4.47E-12	6	-2,0,3; -1,3,3; -3,2,3; -2,-1,0; -1,-3,0; -4,-2,0
20	20	4	25/12	1	2.60E-12	6	-1,2,0; -2,2,0; -4,1,0; -1,-1,3; -2,-2,3; -1,-4,3
20	50	3.92	14/7	1	1.04E-12	6	-1,2,1; -2,2,1; -4,1,1; -1,-1,2; -2,-2,2; -1,-4,2
50	20	3.92	14/7	1	1.99E-12	6	-1,2,2; -2,2,2; -4,1,2; -1,-1,1; -2,-2,1; -1,-4,1
50	50	4	10/5	1	1.01E-12	6	-1,2,3; -2,2,3; -4,1,3; -1,-1,0; -2,-2,0; -1,-4,0

[1] Calculated as $A = L \cdot 2 \cdot (W + G)$

[2] Formatted as "number of sensing interdigital electrodes/number of guard interdigital electrodes"

[3] Three different peripheral scales are used, to make sure the capacitances are in similar orders of magnitude. Ideally comparisons are made between capacitors with the same peripheral scales.

[4] Calculated as $C = A \cdot C_{\square}$, where $C_{\square}$ is the capacitance per area following from the COMSOL simulations

[5] Formatted as "X,Y,A", with the die X coordinate, the die Y coordinate, and the capacitor number on the die A. This table has been filled from the die creation output logs using a script. Please check the markings on the wafer to ensure the values are as expected.

Table A.2: Capacitors with constant width, varying electrode count, no guard electrodes. marking (C)onstant

Width (μm)	Gap (μm)	Sensing area[1] (mm^2)	Electrode counts [2]	Peripheral scale [3]	Expected capacitance [4]	copies	Placement on wafer [5]
1	1	0.01	25	0.05	2.21E-13	2	1,-1,0; 2,-2,0
1	2	0.0096	16	0.05	1.01E-13	2	1,-1,1; 2,-2,1
2	1	0.0096	16	0.05	1.66E-13	2	1,-1,2; 2,-2,2
2	2	0.0096	12	0.05	8.93E-14	2	1,-1,3; 2,-2,3
2	5	0.1568	28	0.2	5.41E-13	2	1,-2,0; 1,-4,0
5	2	0.1568	28	0.2	1.05E-12	2	1,-2,1; 1,-4,1
5	5	0.16	20	0.2	4.93E-13	2	1,-2,2; 1,-4,2
5	10	0.156	13	0.2	2.26E-13	2	1,-2,3; 1,-4,3
10	5	0.156	13	0.2	3.71E-13	2	2,-1,0; 2,-3,0
10	10	0.16	10	0.2	2.16E-13	2	2,-1,1; 2,-3,1
10	20	3.96	33	1	2.73E-12	2	2,-1,2; 2,-3,2
20	10	3.96	33	1	4.47E-12	2	2,-1,3; 2,-3,3
20	20	4	25	1	2.60E-12	2	1,-3,0; 3,-2,0
20	50	3.92	14	1	1.04E-12	2	1,-3,1; 3,-2,1
50	20	3.92	14	1	1.99E-12	2	1,-3,2; 3,-2,2
50	50	4	10	1	1.01E-12	2	1,-3,3; 3,-2,3

[1] Calculated as $A = L \cdot 2 \cdot (W + G)$
[2] The number of sensing interdigital electrodes.
[3] Three different peripheral scales are used, to make sure the capacitances are in similar orders of magnitude. Ideally comparisons are made between capacitors with the same peripheral scales.
[4] Calculated as $C = A \cdot C_{\square}$, where $C_{\square}$ is the capacitance per area following from the COMSOL simulations
[5] Formatted as ”X,Y,A”, with the die X coordinate, the die Y coordinate, and the capacitor number on the die A. This table has been filled from the die creation output logs using a script. Please check the markings on the wafer to ensure the values are as expected.

Table A.3: Capacitors with constant electrode count, varying scale, guard electrodes. markings (G)uard (S)caled

Width (μm)	Gap (μm)	Sensing area[1] (mm^2)	Electrode counts [2]	Peripheral scale [3]	Expected capacitance [4]	copies	Placement on wafer [5]
1	1	0.0032	20/6	0.02	7.07E-14	2	0,0,0; 2,1,0
1	2	0.0072	20/6	0.03	1.25E-13	2	0,0,1; 2,1,1
1	5	0.0288	20/6	0.06	3.04E-13	2	0,0,2; 2,1,2
2	1	0.0072	20/6	0.03	7.56E-14	2	0,0,3; 2,1,3
2	2	0.0128	20/6	0.04	1.19E-13	2	1,0,0; 1,2,0
2	5	0.0392	20/6	0.07	2.62E-13	2	1,0,1; 1,2,1
2	10	0.1152	20/6	0.12	5.31E-13	2	1,0,2; 1,2,2
5	1	0.0288	20/6	0.06	9.88E-14	2	1,0,3; 1,2,3
5	2	0.0392	20/6	0.07	1.35E-13	2	0,1,0; 0,3,0
5	5	0.08	20/6	0.1	2.46E-13	2	0,1,1; 0,3,1
5	10	0.18	20/6	0.15	4.28E-13	2	0,1,2; 0,3,2
10	2	0.25	20/6	0.25	4.39E-13	2	0,1,3; 0,3,3
10	5	0.1152	20/6	0.12	1.73E-13	2	2,0,0; 3,1,0
10	10	0.18	20/6	0.15	2.61E-13	2	2,0,1; 3,1,1
5	20	0.32	20/6	0.2	4.33E-13	2	2,0,2; 3,1,2
10	20	0.36	20/6	0.3	4.06E-13	2	2,0,3; 3,1,3
10	50	1.44	10/6	0.6	1.09E-12	2	1,1,0; 2,2,0
20	5	0.25	10/6	0.25	1.68E-13	2	1,1,1; 2,2,1
20	10	0.36	10/6	0.3	2.48E-13	2	1,1,2; 2,2,2
20	20	0.64	10/6	0.4	4.16E-13	2	1,1,3; 2,2,3
20	50	1.96	10/6	0.7	9.95E-13	2	0,2,0; 1,3,0
50	10	1.44	10/6	0.6	3.62E-13	2	0,2,1; 1,3,1
50	20	1.96	10/6	0.7	5.20E-13	2	0,2,2; 1,3,2
50	50	4	10/6	1	1.01E-12	2	0,2,3; 1,3,3

[1] Calculated as $A = L \cdot 2 \cdot (W + G)$

[2] The number of sensing interdigital electrodes. The larger capacitances use a separate electrode count, as the size of these devices is limited.

[3] The peripheral scale follows from the wavelength $2 * (W + G)$

[4] Calculated as $C = A \cdot C_{\square}$, where $C_{\square}$ is the capacitance per area following from the COMSOL simulations

[5] Formatted as ”X,Y,A”, with the die X coordinate, the die Y coordinate, and the capacitor number on the die A. This table has been filled from the die creation output logs using a script. Please check the markings on the wafer to ensure the values are as expected.

Table A.4: Capacitors with constant electrode count, constant scale, guard electrodes, and meanders. markings (M)eander (C)onstant

Width (μm)	Gap (μm)	Sensing area[1] (mm^2)	Electrode counts [2]	Peripheral scale [3]	Expected capacitance [4]	copies	Placement on wafer [5]
5	5	0.16	20/10	0.2	4.93E-13	2	0,-1,0; 0,-3,2
5	10	0.156	13/6	0.2	2.26E-13	2	0,-1,1
10	5	0.156	13/6	0.2	3.71E-13	2	0,-1,2
10	10	0.16	10/5	0.2	2.16E-13	2	0,-1,3; 0,-3,3
10	20	3.96	33/16	1	2.73E-12	1	0,-2,0; 0,-4,0
20	10	3.96	33/16	1	4.47E-12	1	0,-2,1; 0,-4,1
20	20	4	25/12	1	2.60E-12	2	0,-2,2; 0,-4,2
20	50	3.92	14/7	1	1.04E-12	1	0,-2,3
50	20	3.92	14/7	1	1.99E-12	1	0,-3,0
50	50	4	10/5	1	1.01E-12	2	0,-3,1; 0,-4,3

[1] Calculated as $A = L \cdot 2 \cdot (W + G)$

[2] Formatted as "number of sensing interdigital electrodes/number of guard interdigital electrodes"

[3] Two different peripheral scales are used, to make sure the capacitances are in similar orders of magnitude. Ideally comparisons are made between capacitors with the same peripheral scales.

[4] Calculated as $C = A \cdot C_{\square}$, where $C_{\square}$ is the capacitance per area following from the COMSOL simulations

[5] Formatted as "X,Y,A", with the die X coordinate, the die Y coordinate, and the capacitor number on the die A. This table has been filled from the die creation output logs using a script. Please check the markings on the wafer to ensure the values are as expected.

Figure A.2 displays the expected capacitance for the capacitors on the different positions on the wafer. This is the ideal value for the capacitance that follows from the capacitance per area found in the COMSOL simulations. Fringing effects, filtering effects relating to the electrode lengths, and effects of the connecting geometry are not included.

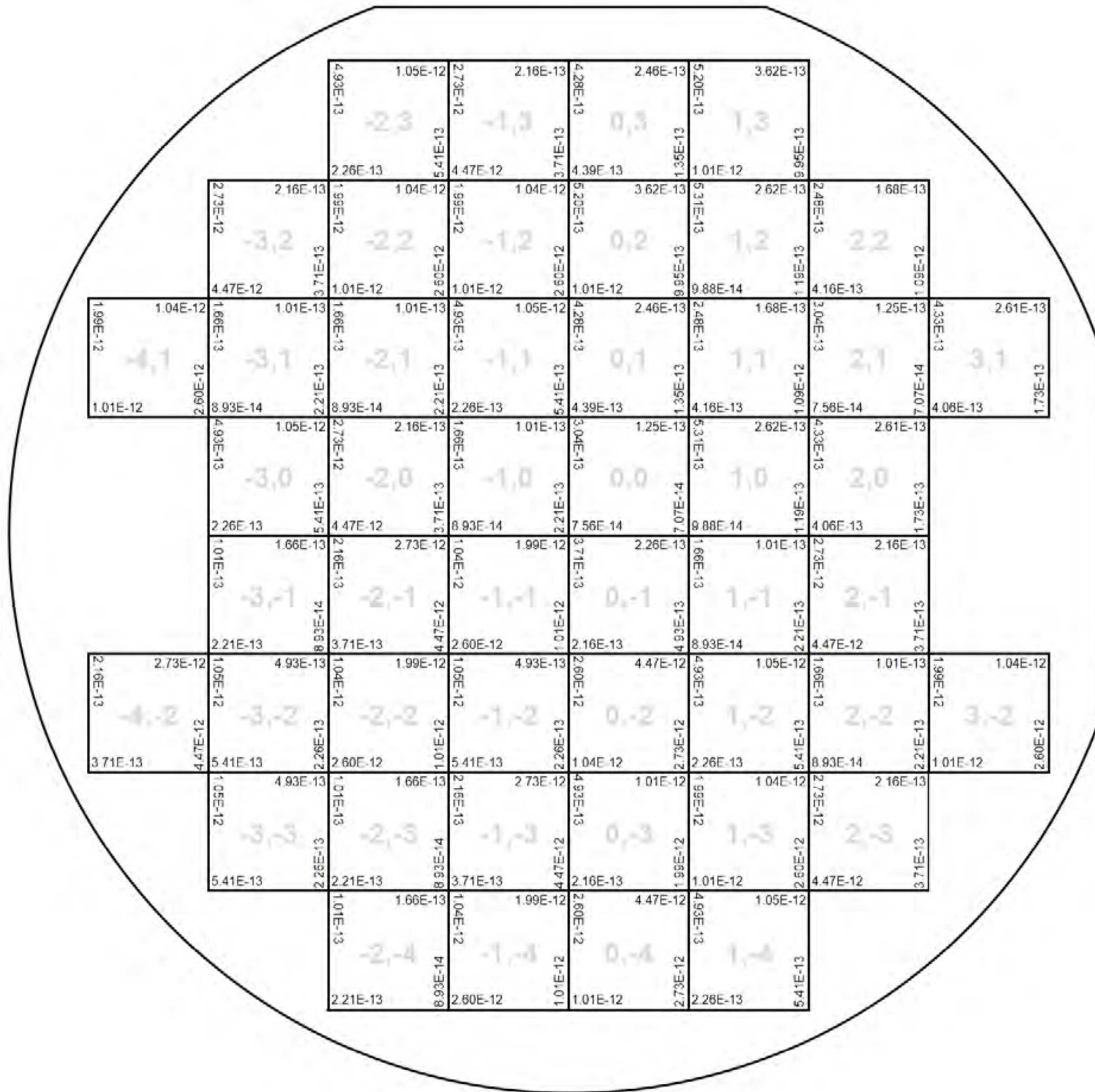


Figure A.2: Map of the wafer with the expected values of the capacitance for the capacitors on the dies.

A.5. Current Source

A.5.1. Current source measurements

This table reports the measured values for the output current of the fabricated current source for different resistor values. This value was measured directly in series with the load. A $9\,\mathrm{V}$ voltage source was used in both cases.

Table A.5: Current source measurements

R_{load} [Ω]	I_{load} [mA] throw 1	throw 2	throw 3
100	1	2	4.97
500	1	2	4.97
1000	1	2	4.97
1220	1	2	4.97
1330	1	2	4.97
1440	1	2	4.97[1]
1555	1	2	4.86[1]
2000	1	2	3.82[1]
3300	1	2	2.35[1]
4700	1	1.77[1]	1.63[1]
6800	1	1.21[1]	1.12[1]
7800	1	1.02[1]	0.97[1]
10000	0.86[1]	0.83[1]	0.76[1]

[1] Red low current warning LED on

A.5.2. Plot generation code

```
function plotCurrentSource(simulationDataPath, measurementDataPath)
    %{
    FUNCTION NAME:
      plotCurrentSource

    DESCRIPTION:
      Creates a plot showing the combined current source data and
      measurements for the different current settings.

    INPUT:
      simulationDataPath - path to the exported LTSpice simulation data.
      the table at the path should have columns r_load [Ohm], I_load [A],
      and I_LED [A}.
      measurementDataPath - path to a table with the measurement data. The
      table at the path should a column for r_load [Ohm] and three columns
      I_load [mA] for the different current settings.

    OUTPUT:
      A figure is opened with the plotted data.

    AUTHOR:
      Jasper Rietveld
      Student number 4581881
    %}

    % Import simualtion data
    simData = importdata(simulationDataPath);

    % Separate the imported data for the different parameters
    resistanceLoad = simData.data(1:201,1);
    currentLoad = reshape(simData.data(:,2), [201 3]);
    currentLED = reshape(simData.data(:,3), [201 3]);

    % Load table with measurement data
```

```
    measData = cell2mat(struct2cell(load(measurementDataPath)));

    % Threshold current through LED above which LED is considered on
    LEDThreshold = 10e-3;

    % Start figure
    figure;
    hold on;

    % Plot the current for each of the settings
    for i=1:3
        % Find load where LED current is first above threshold
        i_min = find( currentLED(:,i) > LEDThreshold, 1 );

        % Plot current up until this load with solid line
        plot(resistanceLoad(1:i_min), currentLoad(1:i_min,i), ...
            "k",'LineWidth',1.5);

        % Plot a vertical line at the load value
        xline(resistanceLoad(i_min),"--k",'LineWidth',1)

        % Plot current beyond the load value with dotted line
        plot(resistanceLoad(i_min:end,1), currentLoad(i_min:end,i), ...
            ":k",'LineWidth',1.5);
    end

    % Plot measurement data as X markers
    plot(measData(:,1), measData(:,2:end)*1e-3, ...
        "xk",'LineWidth',1,'MarkerSize',12);

    % Format plot
    plotBounds = [1, 10000; 0, 7e-3];
    xlim(plotBounds(1,:));
    ylim(plotBounds(2,:));
    grid minor;

    ylabel("Load current [A]");
    xlabel("Load resistance [\Omega]");
    title("Performance of the designed current source");
end
```

A.6. Mask Generation Application Overview

For the creation of the mask with a variety of capacitors, a COMSOL application was written. An overview of the application interface is shown in Figure A.3.

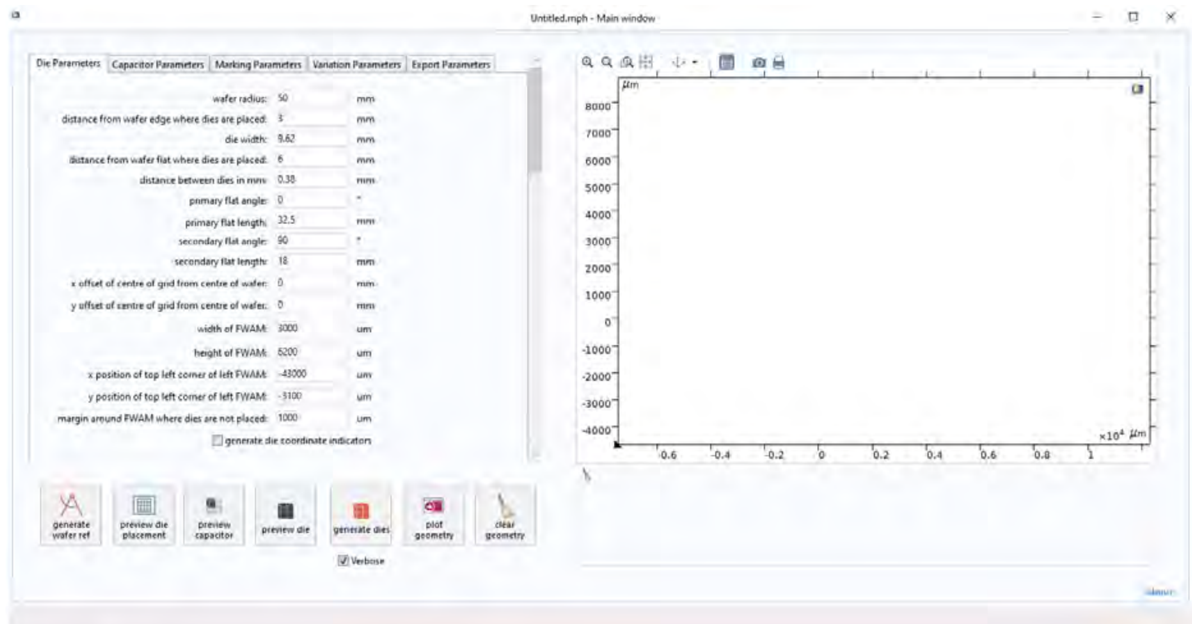


Figure A.3: overview of the layout of the mask generation application

The interface contains four areas with different functions. On the left side there is an area with several input fields, where the values for parameters of the die and mask generation can be entered. Below the input area is a series of buttons with which several methods can be executed to generate geometry or preview of geometry. The resulting geometry is then displayed in the geometry window on the right side. Lastly there is the output log below the geometry window that shows any errors or warnings that occur during the execution of the code.

A.6.1. Application Usage

The parameter input has five separate tabs with parameters for different parts of the mask generation. Values entered in the parameter fields will be used by the methods called using the buttons.

Die Parameters The first tab labelled ”Die Parameters” is used when setting up where on the wafer dies will be placed.

Firstly, wafer dimensions and flat locations can be specified. By pressing the ”generate wafer ref” button, one can then create and display the geometry of a preview of the wafer. The shape of the wafer with the flats removed will be drawn, an example of which is shown in Figure A.4. An area is removed from this shape based on specified margins that indicates where the dies will be placed on the wafer.

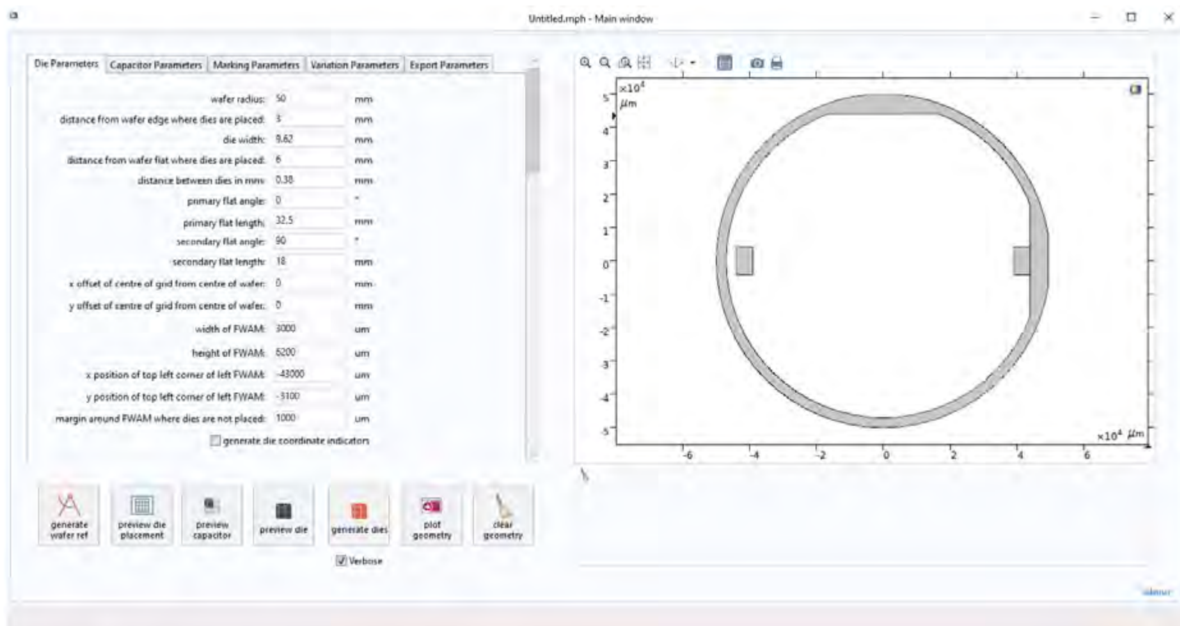


Figure A.4: an example of generated wafer reference geometry

Parameters are also included for the placement of FWAM markers. These markers can be used to align different layers during lithography steps in the device fabrication process. While the markers are not generated by the application, the die generation will take the margins for the markers into account and not place dies in these regions. Rectangular areas that indicate the FWAM marker positions are included on the wafer preview. The code for generating the wafer reference is found in Appendix A.7.1.

When the wafer shape is set up, the ”preview dies” button can be pressed to preview the layout of the dies on the wafer.

The die placement algorithm used is fairly simple. Coordinates for a grid centre, die dimensions, and a distance between the dies is specified. The grid centre is the place where two cut lines cross. Possible die placements in a grid around these coordinates are then checked to see if they do not overlap with the entered wafer margins.

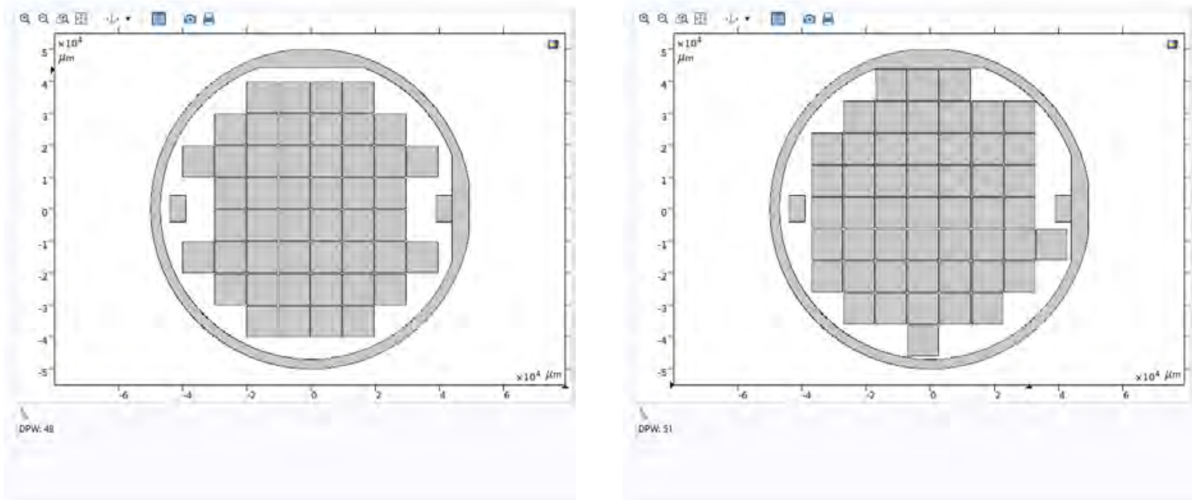
Any valid die positions that do not overlap are then drawn on the preview. The same placement algorithm will be used again in later die generation steps.

When all positions are considered, the number of valid positions will be printed to the output log. By playing around with the die spacing and grid centre, one can attempt to optimise the number of dies that can be placed on the wafer.

Lastly, if the tickbox ”generate die coordinate indicators” is ticked, geometry indicating the coordinates of each wafer will be placed on the die previews. This can be useful when specifying coordinate ranges in later steps.

One can find the code with the die placement algorithm in Appendix A.7.3.

Two examples of die placements preview on a wafer are shown in Figure A.5.



(a) die placement with the grid centred. 48 dies fit within the chosen margins

(b) die placement with an added offset, fitting an extra 3 dies in the chosen margins

Figure A.5: Result of using the die placement preview. The number of dies per wafer is showed in the message log

The ”plot geometry” and ”clear geometry” buttons can be used to manually update the shown geometry if the methods do not do this automatically when it is desired. The code for methods written to do this is found in Appendix A.7.8.

Capacitor Parameters In the ”capacitor parameters” tab one can specify the base capacitor dimensions and options. These are as specified in Appendix **??**. Because COMSOL parses the parameter fields as equations, it is possible to use parameter names in these fields as well as equations using these parameter fields.

The entered values will be used for all capacitors placed with the exception of parameters that are varied with the placement. What parameters are varied is specified in a different tab.

The capacitor geometry can previewed with the ”preview capacitor” button, which does the same as building the capacitor for simulations. An example of this is shown in Figure A.6.

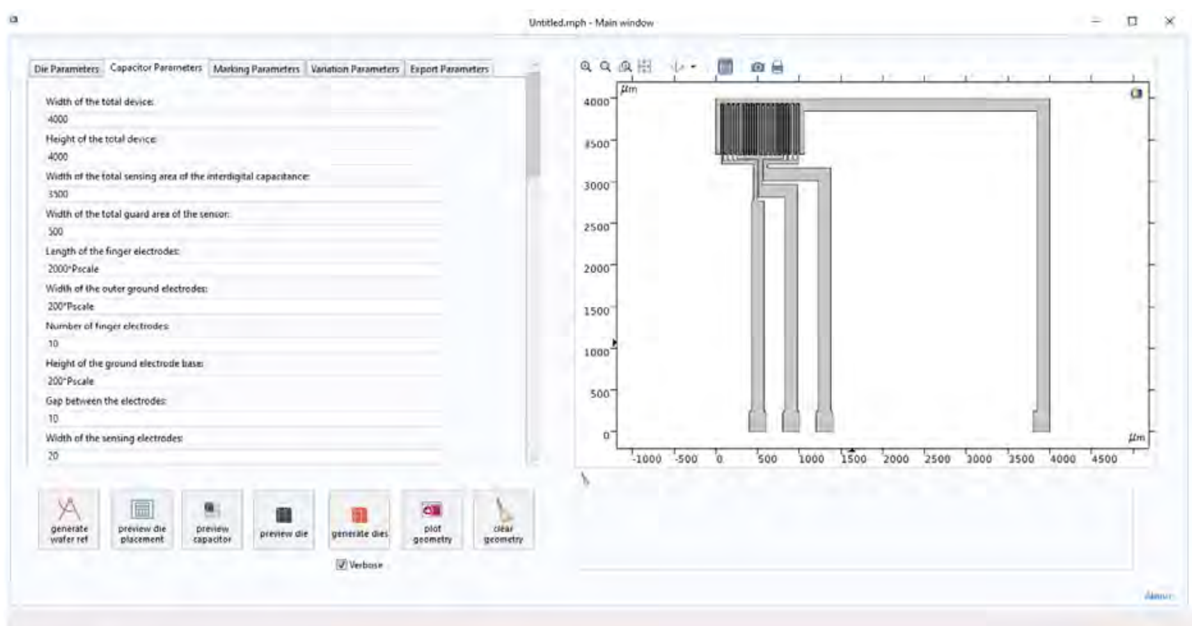


Figure A.6: an example of generated capacitor within the application

Marking Parameters The ”Marking Parameters” tab has input fields relating to the cut marks. These are the patterns that surround the capacitors to indicate the dimensions of the die. The width of these lines can be specified. Where the capacitors and their pads lie near the edge, the marking is not generated. Some margin between these markings and the capacitor can also be specified here.

Text markings for identifying the capacitors is also placed near these markings. Because COMSOL doesn’t implement text as geometry, methods were written to generate text by drawing a 5x3 bitmap using geometry squares. The size of these pixels can be specified in a text field. The code for the text generation can be found in Appendix A.7.2. The text given to each capacitors is chosen in the following tab.

The ”preview die” button can again be used to build the generated geometry to determine whether the markings fit and have appropriate sizes. This will run the code found in Appendix A.7.7. The die is then showed as seen in Figure A.7.

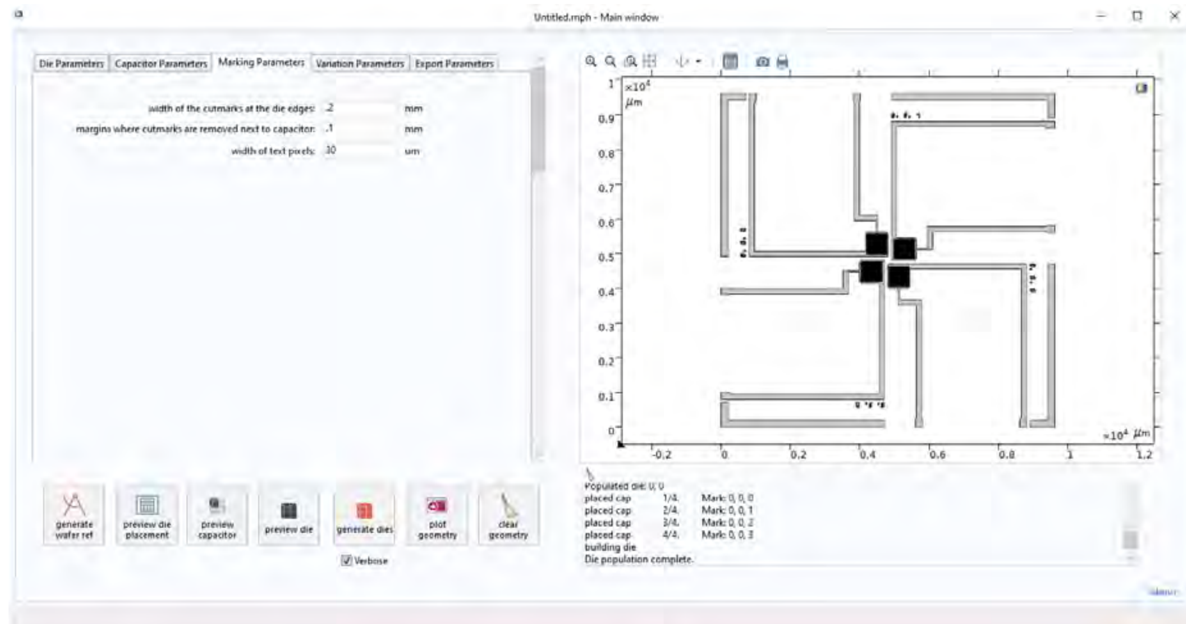


Figure A.7: A die preview displayed in the mask generation application

Variation Parameters The fourth tab labelled ”Variation Parameters” allows the user to specify a of parameters to be varied across a range of values. This has been used to, for instance, vary the interdigital electrode width and height, but also the total device size.

When the ”generate dies” button is clicked, the code displayed in Appendix A.7.6 is ran, generating the dies at the different positions with the parameter variation specified in this tab. A range of variation parameters can be controlled.

The user first specifies a range of coordinates to place the generated dies. While these coordinates do not affect the geometry, they are used in identifying text labels on the capacitors and the exported files. From a drop-down list, one can select whether to place the dies in a certain quadrant or to use custom placement.

If a quadrant is selected, the dies will be placed on the quadrant in a counter-clockwise, spiralling pattern from the centre outwards to make sure that as much of the variation range is placed near the centre. An example of this order for the first quadrant is shown in Figure A.8.

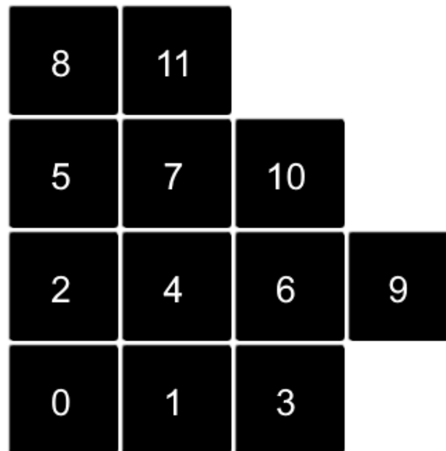


Figure A.8: a diagram showing the placement order of the dies in quadrant placement mode

With custom placement, the user specifies the coordinates manually. A string of the form “x0,y0;x1,y1;x2,y2;...” can be entered in the text field. This string is then parsed using RegEx to find the coordinates at which to place the dies. The order of the placement will be the order of the coordinates in the list.

The user can then specify parameters to be varied across the chosen placement range. There is no set number or limit for how many parameters can be varied, as long as the variables exist. Instead the user enters the amount of parameters to vary. The user can then select that number of parameter value ranges to edit from a drop-down list.

For each of the ranges one can then specify what parameter should be changed and with what values. For the parameter choice, the corresponding name is entered. Then for the variation, an option is selected from another drop-down list to enable input fields to enter a range of values. One can choose ”MinMax” to automatically generate a linear range of numbers between two values. Alternatively, ”MinMaxCenter” can be used to split the parameter range into two ranges around a centre. This can be useful when varying you want a specific median value in the range. Lastly, ”List” can be chosen to enable entering a comma separated list of values. The list, formatted as ”val1,val2,val3”, will again be parsed using RegEx to filter out the float values. Extra marking text can be specified here as well. if such a marking is added, the marking will be added to the coordinate marker along with the parameter value given for the specific capacitor.

For the variation of multiple parameters, one can also choose how to combine the ranges from a drop-down list. For each range, a list of values will be generated. These lists are combined into a list of parameter value combinations. Choosing ”Var1 under Var2”, the earlier specified parameters will change between each capacitor while the next parameter stays the same until the earlier parameters have varied through their entire range. ”Var2 under Var1” work similarly, but with later parameters in the list changing first. By choosing ”Var1 with Var2”, both values will change at the same time. This is useful for lists where you want to use explicit combinations.

Finally, it’s possible to add a marking that is appended to all the capacitors that will be placed. This way, one can add identifiers showing the type of capacitor, for instance whether they have guard electrodes or how parameters are scaled.

An example die generated using the ”preview die” button is shown in Figure A.9.

In this example, the width and gap vary, for which the markings ”W” and ”G” have been specified. Because the capacitors have guard electrodes and their connecting electrode size is constant, the global mark ”GC” is also added. This results in markings such as ”0, 0, 0 W 10.0 G 10.0 GC” for the first capacitor.

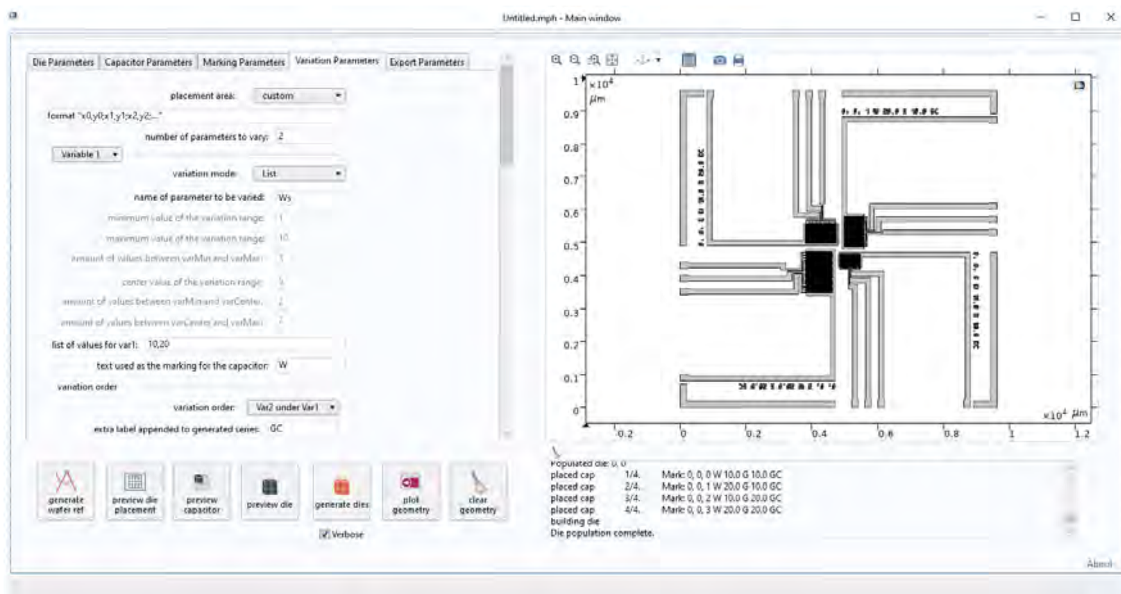


Figure A.9: example of a generated die with parameter variation. Both width and height vary. Note the labels given and how they appear in the message log.

Export Parameters The last tab labelled "Export Parameters" contains the "export" tickbox, as well as a file path to choose a location to export to, If the "export" tickbox is ticked, running "generate dies" will generate the geometry for the dies with parameter variation and export the results. The result is shown in Figure A.10.

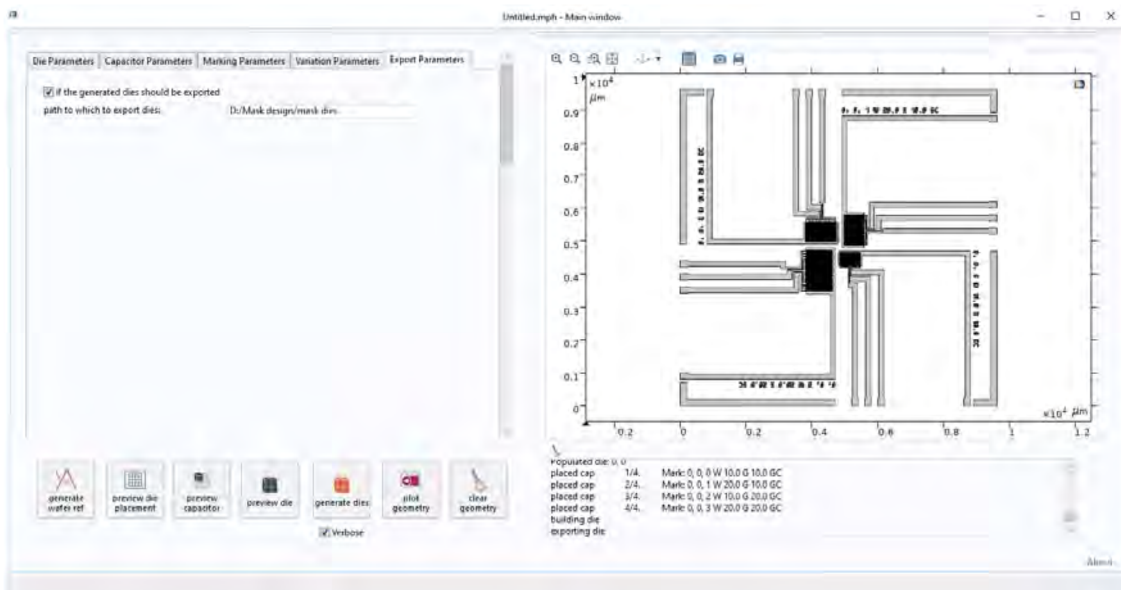


Figure A.10: Overview of the export tab in the application

Exporting the files is necessary in order for the geometry to be usable in a different program suitable for exporting to file formats used by mask manufacturers, which COMSOL can not do. The application exports the dies as DFX files. These can be imported into L-Edit, which has been used to create the Gerber files for the mask manufacturing.

COMSOL will build geometry very slowly if the geometry size increases. It is to be expected that building geometry becomes slower, as the evaluation of booleans, for instance, will become much more computationally expensive if the geometry increases. All included geometry has to be checked

against a large number of other shapes. However, geometry should not be rebuilt after each geometry addition, and it has also been noticed that insertions or removal from lists tends to become slow in these cases, suggesting that COMSOL also uses data-structures that are not optimised for large amounts of geometry.

For these reasons, the dies are built and exported one by one into separate files to limit the amount of geometry present at once. When imported into L-Edit, these can then be placed in the correct position. The file names contain the appropriate coordinates to make this process easier.

A.6.2. Further remarks

While the COMSOL software package is not intended as mask design software, nor is it particularly good for it, writing this application has allowed using the geometry already created for the simulations on the final mask.

It is possible to extend the application to usage with other geometries with relative ease. However, carrying out the process the other way round, first modelling the geometry in software for mask design and then importing that into COMSOL, is likely a better option. L-edit also offers methods to programmatically generate geometry. It is better suited for handling the large geometry that COMSOL struggles with. However, the COMSOL API has shown to be well polished and is well documented, which has made the process of implementing the application relatively easy.

Effort has been put into ensuring code quality across the methods of the application. There are checks in place to verify certain input combinations, such as ensuring the chosen parameters exist and that the chosen die locations are present on the wafer. Additionally, warnings will be displayed in the message log if the program uses the input in a way that the user may not intend (for instance if assumptions are made on the users intentions because of incorrect input). However, defensive programming has not been the focus of the development, so certain input may still result in errors.

A.7. Mask Generation Application COMSOL code

This section of the appendix lists the code for the COMSOL application used to generate the mask. The COMSOL API is based on Java. The code is separated into different methods which are called when the user interacts with the application. The function of each of the methods is listed in the header within the code.

A.7.1. Generate Wafer Reference

```
/**
 * Generate Wafer Reference
 *
 * Method to programatically add geometry to show the wafer size along with the
 * margins for die placement and clearance for FWAM markers.
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Input variables from the COMSOL model set via application */

double xFWAM = model.param().evaluate("xFWAM", "um");
double yFWAM = model.param().evaluate("yFWAM", "um");
double wFWAM = model.param().evaluate("wFWAM", "um");
double hFWAM = model.param().evaluate("hFWAM", "um");
double dFWAM = model.param().evaluate("dFWAM", "um");

/* Create geometry for the wafer. */
model.component("comp1").geom("geom1").create("wafer", "Circle");
with(model.component("comp1").geom("geom1").feature("wafer"));
  set("r", "rWafer");
endwith();

/* Flats are created by subtracting rectangles from the wafer. */

/* flat 1 */
model.component("comp1").geom("geom1").create("flat1", "Rectangle");
with(model.component("comp1").geom("geom1").feature("flat1"));
```

```
  set("base", "center");
  set("rot", "aFlat1");
  set("pos", new String[]{"cos(aFlat1)*rWafer", "sin(aFlat1)*rWafer"});
  set("size", new String[]{".5*(rWafer-sqrt(rWafer^2-(lFlat1)^2))", "lFlat1*2"});
endwith();

/* flat 2 */
model.component("comp1").geom("geom1").create("flat2", "Rectangle");
with(model.component("comp1").geom("geom1").feature("flat2"));
  set("base", "center");
  set("rot", "aFlat2");
  set("pos", new String[]{"cos(aFlat2)*rWafer", "sin(aFlat2)*rWafer"});
  set("size", new String[]{".5*(rWafer-sqrt(rWafer^2-(lFlat2)^2))", "lFlat2*2"});
endwith();

/* Subtract flats from wafer. */
model.component("comp1").geom("geom1").create("difWaferFlats", "Difference");
with(model.component("comp1").geom("geom1").feature("difWaferFlats").selection("input"));
  set("wafer");
endwith();
with(model.component("comp1").geom("geom1").feature("difWaferFlats").selection("input2"));
  set("flat1", "flat2");
endwith();

/* Create an inset circle to subtract from the wafer centre to indicate placement margins. */

model.component("comp1").geom("geom1").create("waferInset", "Circle");
with(model.component("comp1").geom("geom1").feature("waferInset"));
  set("r", "rWafer-wBorder");
endwith();

/* Add extra margin at the flats by subtracting an area form the inset circle. */

/* flat 1 */
model.component("comp1").geom("geom1").create("flat1Inset", "Rectangle");
with(model.component("comp1").geom("geom1").feature("flat1Inset"));
  set("base", "center");
  set("rot", "aFlat1");
  set("pos", new String[]{"cos(aFlat1)*rWafer", "sin(aFlat1)*rWafer"});
  set("size", new String[]{"2*wBorderFlat", "2*rWafer"});
endwith();

/* flat 2 */
model.component("comp1").geom("geom1").create("flat2Inset", "Rectangle");
with(model.component("comp1").geom("geom1").feature("flat2Inset"));
  set("base", "center");
  set("rot", "aFlat2");
  set("pos", new String[]{"cos(aFlat2)*rWafer", "sin(aFlat2)*rWafer"});
  set("size", new String[]{"2*wBorderFlat", "2*rWafer"});
endwith();

/* Subtract flats from inset wafer. */

model.component("comp1").geom("geom1").create("difWaferFlatsInset", "Difference");
with(model.component("comp1").geom("geom1").feature("difWaferFlatsInset")
     .selection("input"));
  set("waferInset");
endwith();
with(model.component("comp1").geom("geom1").feature("difWaferFlatsInset")
     .selection("input2"));
  set("flat1Inset", "flat2Inset");
endwith();

/* Subtract inset wafer from main wafer. */

model.component("comp1").geom("geom1").create("difWaferInset", "Difference");
with(model.component("comp1").geom("geom1").feature("difWaferInset").selection("input"));
  set("difWaferFlats");
endwith();
with(model.component("comp1").geom("geom1").feature("difWaferInset").selection("input2"));
  set("difWaferFlatsInset");
```

```
endwith();

/* Add the FWAM markers */

/* left FWAM marker */
model.component("comp1").geom("geom1").create("leftFWAM", "Rectangle");
with(model.component("comp1").geom("geom1").feature("leftFWAM"));
  set("base", "corner");
  set("pos", new double[]{xFWAM-dFWAM, yFWAM-dFWAM});
  set("size", new double[]{wFWAM+2*dFWAM, hFWAM+2*dFWAM});
endwith();

/* right FWAM marker placed on opposite side from the left FWAM marker */
model.component("comp1").geom("geom1").create("rightFWAM", "Rectangle");
with(model.component("comp1").geom("geom1").feature("rightFWAM"));
  set("base", "corner");
  set("pos", new double[]{-xFWAM-dFWAM-wFWAM, yFWAM-dFWAM});
  set("size", new double[]{wFWAM+2*dFWAM, hFWAM+2*dFWAM});
endwith();
```

A.7.2. Generate Text

```
/**
 * Generate Text
 *
 * Method to programatically add geometry to show the wafer size along with the
 * margins for die placement and clearance for FWAM markers.
 *
 * @param characters  characters to be placed. Only [0-9], [A-F] and [-,. ] are supported
 *                    at this time
 * @param positionX   x position of the first character
 * @param positionY   y position of the first character
 * @param pixWidth    size of the pixels making up the character. Characters are
 *                    5 pixels high and 3 pixels to 5 pixels wide.
 * @param stringID    id of the text geometry to be placed
 * @param drawFast    if true, the function doesn't join pixels with unions
 * @param rotation    rotation of the text
 * @param maxWidth    width of the string before it is cut off
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */


/* Create cumulative selection to contain all pixels */
if (!drawFast) {
  model.component("comp1").geom("geom1").selection()
    .create(stringID+"Charsel", "CumulativeSelection");
}

/* draw the characters of the text */
double xPlacement = 0; // Position at which the text placement starts
double pixRotation = Math.toRadians(rotation); // Rotation of the pixels in radians

for (int c = 0; c < characters.length(); c++)
{
  char character = Character.toUpperCase(characters.charAt(c));

  /* retrieve bitmap encoding the pixels to be placed for a specific characters */
  /* This could be defined in a dictionary for clarity */
  int charWidth = 3;
  String bitmap = "000000000000000";
  switch (character) {
    case '0':
      bitmap = "011101101101110";
      break;
    case '1':
      bitmap = "001011001001001";
      break;
    case '2':
      bitmap = "111001111100111";
```

```
    break;
  case '3':
    bitmap = "111001111001111";
    break;
  case '4':
    bitmap = "101101111001001";
    break;
  case '5':
    bitmap = "111100111001111";
    break;
  case '6':
    bitmap = "011100111101110";
    break;
  case '7':
    bitmap = "111001001001001";
    break;
  case '8':
    bitmap = "011101111101110";
    break;
  case '9':
    bitmap = "011101111001110";
    break;
  case 'A':
    bitmap = "010101111101101";
    break;
  case 'B':
    bitmap = "110101110101110";
    break;
  case 'C':
    bitmap = "011100100100011";
    break;
  case 'D':
    bitmap = "110101101101110";
    break;
  case 'E':
    bitmap = "111100111100111";
    break;
  case 'F':
    bitmap = "111100111100100";
    break;
  case 'G':
    bitmap = "111100101101111";
    break;
  case 'H':
    bitmap = "101101111101101";
    break;
  case 'I':
    bitmap = "010010010010010";
    break;
  case 'J':
    bitmap = "001001001101010";
    break;
  case 'K':
    bitmap = "101101110101101";
    break;
  case 'L':
    bitmap = "100100100100111";
    break;
  case 'M':
    charWidth = 5;
    bitmap = "1111010101101011010110101";
    break;
  case 'N':
    bitmap = "110101101101101";
    break;
  case 'O':
    charWidth = 4;
    bitmap = "01101001100110010110";
    break;
  case 'P':
    bitmap = "110101110100100";
```

```
    break;
  case 'Q':
    bitmap = "010101101010001";
    break;
  case 'R':
    bitmap = "110101110101101";
    break;
  case 'S':
    bitmap = "011100111001110";
    break;
  case 'T':
    bitmap = "111010010010010";
    break;
  case 'U':
    bitmap = "101101101101111";
    break;
  case 'V':
    bitmap = "101101101101010";
    break;
  case 'W':
    charWidth = 5;
    bitmap = "1010110101101011010101010";
    break;
  case 'X':
    bitmap = "101101010101101";
    break;
  case 'Y':
    bitmap = "101101010010010";
    break;
  case 'Z':
    bitmap = "111001010100111";
    break;
  case '-':
    bitmap = "000000111000000";
    break;
  case ',':
    bitmap = "000000000010010";
    break;
  case '.':
    bitmap = "000000000000010";
    break;
  case ':':
    bitmap = "010000000000010";
    break;
  case '#':
    charWidth = 4;
    bitmap = "11100011111111101010";
    break;
  case ' ':
  default:
    bitmap = "000000000000000";
    break;
}

/* Cut off string if it is too long */
if (xPlacement+charWidth*pixWidth > maxWidth) break;

/* Place the pixels for the character */
for (int j = 0; j < 5; j++)
{
  for (int i = 0; i < charWidth; i++)
  {
    if (bitmap.charAt(j*charWidth+i) == '1') {
      /* For each pixel with value 1, create a square geometry */
      String id = stringID+"_px_"+c+"_"+i+"_"+j;
      model.component("comp1").geom("geom1").create(id, "Square");

      /* Convert the position within the text to an absolute pixel position. */
      double[] pixPos = new double[]{xPlacement+i*pixWidth, (4-j)*pixWidth};
      double[] worldPos = new double[]{positionX+pixPos[0]*Math.cos(pixRotation)
                                      -pixPos[1]*Math.sin(pixRotation),
```

```
                                        positionY+pixPos[0]*Math.sin(pixRotation)
                                          +pixPos[1]*Math.cos(pixRotation)};

        with(model.component("comp1").geom("geom1").feature(id));
          set("pos", worldPos);
          set("size", pixWidth);
          set("rot", rotation);
        endwith();

        /* If drawFast is false, add the new geometry to the selection */
        if (!drawFast) {
          with(model.component("comp1").geom("geom1").feature(id));
            set("contributeto", stringID+"Charsel");
          endwith();
        }
      }
    }
  }

  /* Move the drawing position by the width of a character and a space */
  xPlacement += pixWidth*(charWidth+1);
}

/* If drawFast is false, join the pixels with a union using the selection */
if (!drawFast) {
  model.component("comp1").geom("geom1").create(stringID+"Charuni", "Union");
  model.component("comp1").geom("geom1").feature(stringID+"Charuni")
    .selection("input").named(stringID+"Charsel");
  with(model.component("comp1").geom("geom1").feature(stringID+"Charuni"));
    set("intbnd", false);
  endwith();
}
```

A.7.3. Generate Die Previews

```
/**
 * Generate Die Previews
 *
 * Method to place die previews on the wafer reference to determine how many dies fit
 * given a specific layout.
 *
 * @param showCoords  Whether or not geometry showing the coordinates should be generated.
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Input variables from the COMSOL model set via application */

double rWafer = model.param().evaluate("rWafer", "um");
double aFlat1 = model.param().evaluate("aFlat1");
double aFlat2 = model.param().evaluate("aFlat2");

double wDie = model.param().evaluate("wDie", "um");
double dDie = model.param().evaluate("dDie", "um");

double xOffset = model.param().evaluate("xOffset", "um");
double yOffset = model.param().evaluate("yOffset", "um");

double wBorder = model.param().evaluate("wBorder", "um");
double wBorderFlat = model.param().evaluate("wBorderFlat", "um");

double xFWAM = model.param().evaluate("xFWAM", "um");
double yFWAM = model.param().evaluate("yFWAM", "um");
double wFWAM = model.param().evaluate("wFWAM", "um");
double hFWAM = model.param().evaluate("hFWAM", "um");
double dFWAM = model.param().evaluate("dFWAM", "um");

/* Determine maximum number of dies that fit the wafer size */
double xNum = 2*Math.ceil(rWafer/(wDie+dDie));
double yNum = 2*Math.ceil(rWafer/(wDie+dDie));
```

```
/* Calculate the coordinate of the bottom left die based on the input */
double xStart = xOffset-xNum*(wDie+dDie);
double yStart = yOffset-yNum*(wDie+dDie);

/* Die corners relative to the die position */
double[][] vertexOffsets = {{0, 0}, {wDie, 0}, {0, wDie}, {wDie, wDie}};

/* to allow a coordinate mark of the format "_-XXX,-XXX_",
 * 11 characters times 4 pixel width are needed. */
double markWidth = wDie/45;

int die_num = 0; // Number of dies succesfully placed

/* loop over all die positions on the grid */
for (double x = xStart; x < xNum*(wDie+dDie); x += (wDie+dDie)) {
  for (double y = yStart; y < yNum*(wDie+dDie); y += (wDie+dDie)) {

    /* Determine if all vertices of a die are within the margins */
    boolean inBounds = true;
    for (double[] offset : vertexOffsets)
    {
      double centreDist = Math.pow(x+offset[0], 2)+Math.pow(y+offset[1], 2);
      /*
       * To determine the distance from the wafer edge at the flats, the vertices are
       * essentially projected onto a vector rotated towards the flat.
       */
      double flatDist1 = (x+offset[0])*Math.cos(-aFlat1)-(y+offset[0])*Math.sin(-aFlat1);
      double flatDist2 = (x+offset[0])*Math.cos(-aFlat2)-(y+offset[0])*Math.sin(-aFlat2);

      if (centreDist > Math.pow(rWafer-wBorder, 2) ||
          flatDist1 > rWafer-wBorderFlat ||
          flatDist2 > rWafer-wBorderFlat
          ) {
        inBounds = false;
        break;
      }
    }
    /* Check for intersection with the left FWAM */
    if (x < xFWAM+wFWAM+dFWAM &&
        x+wDie >= xFWAM-dFWAM &&
        y+wDie >= yFWAM-dFWAM &&
        y < yFWAM+hFWAM+dFWAM) {
      inBounds = false;
    }
    /* Check for intersection with the right FWAM*/
    if (x < -xFWAM+dFWAM &&
        x+wDie >= -xFWAM-wFWAM-dFWAM &&
        y+wDie >= yFWAM-dFWAM &&
        y < yFWAM+hFWAM+dFWAM) {
      inBounds = false;
    }

    /* If a die is within the margins, create geometry for it */
    if (inBounds) {
      /* Increment the number of placed dies */
      die_num++;

      String id = "die_"+Math.round(x/(wDie+dDie))+"_"+Math.round(y/(wDie+dDie));

      if (showCoords) {
        /* Add geometry with text for the coordinates if showCoords is true */
        String mark = Math.round(x/(wDie+dDie))+","+Math.round(y/(wDie+dDie));
        generateText(mark, x+markWidth, y+markWidth, markWidth, id, fast, 0, 1e6);
      }

      model.component("comp1").geom("geom1").create(id, "Square");
      with(model.component("comp1").geom("geom1").feature(id));
        set("pos", new double[]{x, y});
        set("size", wDie);
      endwith();
```

```
    }
  }
}

/* Output the number of dies per wafer to the message log. */
message("DPW: "+die_num);
```

A.7.4. Generate Capacitor

```
/**
 * Generate Capacitor
 *
 * Method to place capacitor according to current capacitor parameters in model
 *
 * @param x          x position of the capacitor
 * @param y          y position of the capacitor
 * @param stringID   ID of the capacitor. prepended to identifiers for uniqueness
 * @param rotation   rotation of the capacitor
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Place part based on model parameters (which are set from the form) */
model.component("comp1").geom("geom1").create(StringID+"cap", "PartInstance");

with(model.component("comp1").geom("geom1").feature(StringID+"cap"));
  set("selkeepnoncontr", false);
  set("part", "part1");
  setEntry("inputexpr", "Wt", model.param().evaluate("Wt"));
endwith();

/* Set all parameters of the part according to model */
for (String paramName : model.param("par5").varnames()) {
  try {
    with(model.component("comp1").geom("geom1").feature(StringID+"cap"));
      setEntry("inputexpr", paramName, model.param().evaluate(paramName));
    endwith();
  }
  catch (Exception e) {
    message("Error: Capacitor has no definition for parameter named '"+paramName+"'");
  }
}

/* Position the part according to the method inputs */
with(model.component("comp1").geom("geom1").feature(StringID+"cap"));
  set("rot", rotation);
  set("displ", new String[]{Double.toString(x), Double.toString(y)});
endwith();
```

A.7.5. Generate Cutmarks

```
/**
 * Generate Cutmarks
 *
 * Method to generate cutmarks for a die
 *
 * @param stringID    ID of the die. prepended to identifiers for uniqueness
 * @param x           x position of the die
 * @param y           y position of the die
 *
 * @param width       width of a side of the die
 * @param markWidth   width of the cutmarks at the die edges
 * @param capWidth    width of clearance for capacitor
 * @param clearance   width of extra clearance margins around the capacitors
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Create main square with the die dimensions*/
```

```
model.component("comp1").geom("geom1").create(StringID, "Square");
with(model.component("comp1").geom("geom1").feature(StringID));
  set("pos", new double[]{x, y});
  set("size", width);
endwith();

model.component("comp1").geom("geom1").create(StringID+"CutCentre", "Square");
with(model.component("comp1").geom("geom1").feature(StringID+"CutCentre"));
  set("pos", new double[]{x+markWidth, y+markWidth});
  set("size", width-2*markWidth);
endwith();

/* Add clearance areas for capacitors */
double openingPos = Math.max(0.5*width-clearance, markWidth);
double openingWidth = Math.min(2*clearance+capWidth, width-markWidth-openingPos);

model.component("comp1").geom("geom1").create(StringID+"Cap0Cut", "Rectangle");
with(model.component("comp1").geom("geom1").feature(StringID+"Cap0Cut"));
  set("pos", new double[]{x+openingPos, y-markWidth});
  set("size", new double[]{openingWidth, 3*markWidth});
endwith();
model.component("comp1").geom("geom1").create(StringID+"Cap1Cut", "Rectangle");
with(model.component("comp1").geom("geom1").feature(StringID+"Cap1Cut"));
  set("pos", new double[]{x+width-2*markWidth, y+openingPos});
  set("size", new double[]{3*markWidth, openingWidth});
endwith();
model.component("comp1").geom("geom1").create(StringID+"Cap2Cut", "Rectangle");
with(model.component("comp1").geom("geom1").feature(StringID+"Cap2Cut"));
  set("pos", new double[]{x+width-openingPos-openingWidth, y+width-2*markWidth});
  set("size", new double[]{openingWidth, 3*markWidth});
endwith();
model.component("comp1").geom("geom1").create(StringID+"Cap3Cut", "Rectangle");
with(model.component("comp1").geom("geom1").feature(StringID+"Cap3Cut"));
  set("pos", new double[]{x-2*markWidth, y+width-openingPos-openingWidth});
  set("size", new double[]{3*markWidth, openingWidth});
endwith();

/* subtract clearance areas from main square */
model.component("comp1").geom("geom1").create(StringID+"DifDieCuts", "Difference");
with(model.component("comp1").geom("geom1").feature(StringID+"DifDieCuts").selection("input")
    );
  set(StringID);
endwith();
with(model.component("comp1").geom("geom1").feature(StringID+"DifDieCuts").selection("input2"
    ));
  set(StringID+"CutCentre", StringID+"Cap0Cut", StringID+"Cap1Cut", StringID+"Cap2Cut",
      StringID+"Cap3Cut");
endwith();
```

A.7.6. Generate Dies

```
/**
 * Generate Dies
 *
 * Method to place dies on the wafer and populate them with the capacitor and marking
 * geometry. The capacitor parameters and corresponding markings are varied according
 * to user input.
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Input variables from the COMSOL model set via application */

double rWafer = model.param().evaluate("rWafer", "um");
double aFlat1 = model.param().evaluate("aFlat1");
double aFlat2 = model.param().evaluate("aFlat2");

double wDie = model.param().evaluate("wDie", "um");
double dDie = model.param().evaluate("dDie", "um");
```

```
double xOffset = model.param().evaluate("xOffset", "um");
double yOffset = model.param().evaluate("yOffset", "um");

double wBorder = model.param().evaluate("wBorder", "um");
double wBorderFlat = model.param().evaluate("wBorderFlat", "um");

double xFWAM = model.param().evaluate("xFWAM", "um");
double yFWAM = model.param().evaluate("yFWAM", "um");
double wFWAM = model.param().evaluate("wFWAM", "um");
double hFWAM = model.param().evaluate("hFWAM", "um");
double dFWAM = model.param().evaluate("dFWAM", "um");

/* Create a list of positions at which to place the dies */

/*
 * an array for the positions is allocated with sizes equal to the maximal possible
 * die count.
 */
int xNum = 2*(int) Math.ceil(rWafer/(wDie+dDie));
int yNum = 2*(int) Math.ceil(rWafer/(wDie+dDie));
double[][] selectedDiePositions = new double[xNum*yNum][2];
int selectedDieCount = 0;

double[][] vertexOffsets = {{0, 0}, {wDie, 0}, {0, wDie}, {wDie, wDie}};

switch (placementMode) {
  case 1:
  case 2:
  case 3:
  case 4:
    /* quadrant mode placement */
    message("performing quadrant mode placement");

    /* The position of the first die and offsets to allow placement in a pattern
     * from the centre outwards are defined based on the chosen quadrant.
     * these offsets are the directions which result in counter-clockwise looping for the
     * quadrant.
     */
    double[] quadrant_start = {xOffset, yOffset};
    double[] quadrant_primaryOffset = {wDie+dDie, 0};
    double[] quadrant_secondaryOffset = {-(wDie+dDie), wDie+dDie};
    switch (placementMode) {
      case 2:
        quadrant_start = new double[]{xOffset-(wDie+dDie), yOffset};
        quadrant_primaryOffset = new double[]{0, wDie+dDie};
        quadrant_secondaryOffset = new double[]{-(wDie+dDie), -(wDie+dDie)};
        break;
      case 3:
        quadrant_start = new double[]{xOffset-(wDie+dDie), yOffset-(wDie+dDie)};
        quadrant_primaryOffset = new double[]{-(wDie+dDie), 0};
        quadrant_secondaryOffset = new double[]{wDie+dDie, -(wDie+dDie)};
        break;
      case 4:
        quadrant_start = new double[]{xOffset, yOffset-(wDie+dDie)};
        quadrant_primaryOffset = new double[]{0, -(wDie+dDie)};
        quadrant_secondaryOffset = new double[]{wDie+dDie, wDie+dDie};
        break;
    }

    /* Loop such that the die positions are added from center outwards. This is
     * essentially looping over a triangle.
     */
    double x = quadrant_start[0];
    double y = quadrant_start[1];
    for (int i = 0; i < xNum*2; i++) {
      for (int j = 0; j <= i; j++) {
        x = quadrant_start[0]+i*quadrant_primaryOffset[0]+j*quadrant_secondaryOffset[0];
        y = quadrant_start[1]+i*quadrant_primaryOffset[1]+j*quadrant_secondaryOffset[1];

        /* Determine if all vertices of a die are within the margins */
        boolean inBounds = true;
```

```
        for (double[] offset : vertexOffsets)
        {
          double centreDist = Math.pow(x+offset[0], 2)+Math.pow(y+offset[1], 2);
          /*
           * To determine the distance from the wafer edge at the flats, the vertices are
           * essentially projected onto a vector rotated towards the flat.
           */
          double flatDist1 = (x+offset[0])*Math.cos(-aFlat1)-(y+offset[0])*Math.sin(-aFlat1);
          double flatDist2 = (x+offset[0])*Math.cos(-aFlat2)-(y+offset[0])*Math.sin(-aFlat2);

          if (centreDist > Math.pow(rWafer-wBorder, 2) ||
              flatDist1 > rWafer-wBorderFlat ||
              flatDist2 > rWafer-wBorderFlat
              ) {
            inBounds = false;
            break;
          }
        }

        /* Check for intersection with the left FWAM */
        if (x < xFWAM+wFWAM+dFWAM &&
            x+wDie >= xFWAM-dFWAM &&
            y+wDie >= yFWAM-dFWAM &&
            y < yFWAM+hFWAM+dFWAM) {
          inBounds = false;
        }

        /* Check for intersection with the right FWAM*/
        if (x < -xFWAM+dFWAM &&
            x+wDie >= -xFWAM-wFWAM-dFWAM &&
            y+wDie >= yFWAM-dFWAM &&
            y < yFWAM+hFWAM+dFWAM) {
          inBounds = false;
        }

        /* Positions that satisfy the margins are added to the position array */
        if (inBounds) {
          if (verbose) message("queued die: "+Math.round(x/(wDie+dDie))+
                               ", "+Math.round(y/(wDie+dDie))+" for placement");
          selectedDiePositions[selectedDieCount++] = new double[]{x, y};
        }
      }
    }

    break;
  case 0:
    /* Custom placement based on a list of coordinates */
    message("performing custom placement");

    /*
     * Use RegEx to ensure that the input is a semicolon-separated list of comma-separated
     * coordinates
     */
    if (!positions.matches("^(-?\\d*,-?\\d*)(?:;-?\\d*,-?\\d*)*")) {
      message("Error! Incorrect position format. Format should be x0,y0;x1,y1;x2,y2;...");
      return;
    }

    String positionsConsumable = positions; // Input list from which read pairs are truncated

    /*
     * RegEx matching is used to continually match and then truncate the first coord pair
     * in the string
     */
    java.util.regex.Pattern p = java.util.regex.Pattern.
      compile("^(?<pair>(?<x>-?\\d*),(?<y>-?\\d*);?)");
    java.util.regex.Matcher m = p.matcher(positionsConsumable);
    while (m.find()) {
      x = xOffset+Integer.parseInt(m.group("x"))*(wDie+dDie);
      y = yOffset+Integer.parseInt(m.group("y"))*(wDie+dDie);
```

```
      /* Determine if all vertices of a die are within the margins */
      boolean inBounds = true;
      for (double[] offset : vertexOffsets)
      {
        double centreDist = Math.pow(x+offset[0], 2)+Math.pow(y+offset[1], 2);
        /*
         * To determine the distance from the wafer edge at the flats, the vertices are
         * essentially projected onto a vector rotated towards the flat.
         */
        double flatDist1 = (x+offset[0])*Math.cos(-aFlat1)-(y+offset[0])*Math.sin(-aFlat1);
        double flatDist2 = (x+offset[0])*Math.cos(-aFlat2)-(y+offset[0])*Math.sin(-aFlat2);

        if (centreDist > Math.pow(rWafer-wBorder, 2) ||
            flatDist1 > rWafer-wBorderFlat ||
            flatDist2 > rWafer-wBorderFlat
            ) {
          inBounds = false;
          break;
        }
      }

      /* Check for intersection with the left FWAM */
      if (x < xFWAM+wFWAM+dFWAM &&
          x+wDie >= xFWAM-dFWAM &&
          y+wDie >= yFWAM-dFWAM &&
          y < yFWAM+hFWAM+dFWAM) {
        inBounds = false;
      }

      /* Check for intersection with the right FWAM*/
      if (x < -xFWAM+dFWAM &&
          x+wDie >= -xFWAM-wFWAM-dFWAM &&
          y+wDie >= yFWAM-dFWAM &&
          y < yFWAM+hFWAM+dFWAM) {
        inBounds = false;
      }

      if (inBounds) {
        if (verbose) message("queued die: "+Math.round(x/(wDie+dDie))+", "
                             +Math.round(y/(wDie+dDie))+" for placement.");
        selectedDiePositions[selectedDieCount++] = new double[]{x, y};
      }
      else {
        /* Display a warning in the message log if the coordinate does not satisfy margins */
        message("Warning! die at "+Math.round(x/(wDie+dDie))+
                ", "+Math.round(y/(wDie+dDie))+": skipped, position is not within margins.");
      }

      /*
       * The considered coordinate pair is truncated from the string start so that
       * the next pair is now at the start.
       */
      positionsConsumable = positionsConsumable.substring(m.group("pair").length());
      m = p.matcher(positionsConsumable);
    }
    break;
  default:
    /* This is unreachable */
    message("Error! Placement mode not recognised!");
    return;
}

message("Die selection complete");

/* Create list of capacitor parameter values based on the specified variation ranges */

/* Ensure that a parameter with name specified for the range exists */
for (String name : varsName) {
  boolean varExists = false;
  for (String paramName : model.param().varnames()) {
    if (paramName.equals(name)) varExists = true;
```

```
  }
  if (!varExists) {
    message("Error: No parameter exists with name\""+name+"\"");
    return;
  }
}

message("Calculating parameter variation values");

/* Find maximum count of variation range */
int maxVarCount = 1;
for (int var = 0; var < variationCount; var++)
{
  switch (varsVariationMode[var]) {
    case 0: // None
    default:
      break;
    case 1: // MinMax
      maxVarCount = (int) Math.max(maxVarCount, 2+varsN[var]);
      break;
    case 2: // MinMaxCenter
      maxVarCount = (int) Math.max(maxVarCount, 3+varsNMin[var]+varsNMax[var]);
      break;
    case 3: // List
      /* count number of occurances of ',' in the list to estimate the value count */
      java.util.regex.Pattern pattern = java.util.regex.Pattern.compile("[^,]*,");
      java.util.regex.Matcher matcher = pattern.matcher(varsList[var]);
      int VarCount = 1; // 1 more value than the number of ',' occurances
      while (matcher.find())
        VarCount++;
      maxVarCount = (int) Math.max(maxVarCount, VarCount);
      break;
  }
}


/* Create an array to hold a variation range for each specified parameter */
double[][] varsValues = new double[maxVarCount][variationCount];
int[] varsCount = new int[variationCount];

/* Calculate the values in the variation range based on the variation mode */
for (int var = 0; var < variationCount; var++) {
  switch (varsVariationMode[var]) {
    case 0: // None
    default:
      /* Add default capacitor value as only value */
      varsValues[varsCount[var]++][var] = model.param().evaluate(varsName[var]);
      break;
    case 1: // MinMax
      /* Determine a range of values between a minimum and maximum */
      varsValues[varsCount[var]++][var] = varsMin[var];
      double increment = (varsMax[var]-varsMin[var])/(varsN[var]+1);
      for (int i = 0; i < varsN[var]; i++) {
        varsValues[varsCount[var]++][var] = varsMin[var]+(i+1)*increment;
      }
      varsValues[varsCount[var]++][var] = varsMax[var];
      break;
    case 2: // MinMaxCenter
      /* Determine a range of values from a minimum up to a centre */
      varsValues[varsCount[var]++][var] = varsMin[var];
      increment = (varsCenter[var]-varsMin[var])/(varsNMin[var]+1);
      for (int i = 1; i < varsNMin[var]; i++) {
        varsValues[varsCount[var]++][var] = varsMin[var]+(i+1)*increment;
      }
      /* Determine a range of values from a centre and a maximum */
      varsValues[varsCount[var]++][var] = varsCenter[var];
      increment = (varsNMax[var]-varsCenter[var])/(varsNMax[var]+1);
      for (int i = 1; i < varsNMax[var]; i++) {
        varsValues[varsCount[var]++][var] = varsCenter[var]+(i+1)*increment;
      }
      varsValues[varsCount[var]++][var] = varsMax[var];
```

```
      break;
    case 3: // List
      /* Parse a list of values */

      /* Use RegEx to ensure that the input is a list of comma-separated values */
      if (!varsList[var].matches("^(-?\\d+(\\.\\d*)?)(?:,-?\\d+(\\.\\d*)?)*")) {
        message("Error! Incorrect value list format. Format should be val1,val2,val3,...");
        return;
      }

      /*
       * RegEx matching is used to continually match and then truncate the first coord pair
       * in the string
       */
      String varListConsumable = varsList[var]; // list from which read vars are truncated

      java.util.regex.Pattern p = java.util.regex.Pattern.
        compile("^(?<valgroup>(?<val>-?\\d+(\\.\\d*)?),?)");
      java.util.regex.Matcher m = p.matcher(varListConsumable);
      while (m.find()) {
        /*
         * The considered value is parsed and then truncated from the string start, so that
         * the next pair is now at the start.
         */
        varsValues[varsCount[var]++][var] = Double.parseDouble(m.group("val"));
        varListConsumable = varListConsumable.substring(m.group("valgroup").length());
        m = p.matcher(varListConsumable);
      }
      break;
  }

}

/* Combine the parameter value lists into a single list based on the variation order */

message("Combining parameters");

/*
 * Calcualte the sub and product of the value count for each parameter. Used to
 * check the total combinations against the number of capacitors that will be placed
 */
int varCountProduct = 0;
for (int count : varsCount) varCountProduct *= count;
int varCountSum = 0;
for (int count : varsCount) varCountSum += count;

/* List to hold the combinations for the different dies */
double[][] variationValues = new double[selectedDieCount*4][variationCount];

switch (variationOrder) {
  default:
  case 0: // var2 under var1
    if (varCountProduct > selectedDieCount*4)
      message("Warning: Not enough dies selected for all parameter combinations. "+
              "Extra combinations are ignored.");

    /*
     * Find for each parameter to be varied after how many combinations it should change.
     * a second parameter should change after a first parameter has covered its entire
     * range. For instance, with three parameters with 2, 3 and 4 values respectively,
     * repCounts will become [1, 2, 6].
     */
    int[] repCounts = new int[variationCount];
    int repCount = 1;
    for (int i = 0; i < variationCount; i++) {
      repCounts[i] = repCount;
      if (varsCount[i] != 0)
        repCount *= varsCount[i];
    }

    /* Combine the ranges using the repetition counts. */
```

```
    for (int i = 0; i < selectedDieCount*4; i++) {
      for (int j = 0; j < variationCount; j++) {
        /* The integer division i/repCounts[j] will increase every repCounts[j] increments of
             i */
        variationValues[i][j] = varsValues[(i/repCounts[j])%varsCount[j]][j];
      }
    }
    break;
  case 1: // var1 under var2
    if (varCountProduct > selectedDieCount*4)
      message("Warning: Not enough dies selected for all parameter combinations."+
              "Extra combinations are ignored.");

    /*
     * Find for each parameter to be varied after how many combinations it should change.
     * a first parameter should change after a second parameter has covered its entire
     * range. For instance, with three parameters with 2, 3 and 4 values respectively,
     * repCounts will become [12, 4, 1].
     */
    repCounts = new int[variationCount];
    repCount = 1;
    for (int i = variationCount-1; i >= 0; i--) {
      repCounts[i] = repCount;
      if (varsCount[i] != 0)
        repCount *= varsCount[i];
    }

    /* Combine the ranges using the repetition counts. */
    for (int i = 0; i < selectedDieCount*4; i++) {
      for (int j = 0; j < variationCount; j++) {
        /* The integer division i/repCounts[j] will increase every repCounts[j] increments of
             i */
        variationValues[i][j] = varsValues[(i/repCounts[j])%varsCount[j]][j];
      }
    }

    break;
  case 2: // var1 with var2
    if (varCountSum > selectedDieCount*4)
      message("Warning: Not enough dies selected for all parameter combinations."+
              "Extra combinations are ignored.");

    /* check that all lists are of equal size and find minimum list length. */
    int minVarCount = varsCount[0];
    boolean warningGiven = false;
    for (int count : varsCount) {
      if (minVarCount != count && !warningGiven) {
        message("Warning: the amount of values in the lists for the different "+
                "parameters are not equal. Extra values in larger list are ignored...");
        warningGiven = true; // Only give the warning once
        break;
      }
      minVarCount = Math.min(count, minVarCount);
    }

    /* Combine lists using the minimum list length as a modulo. */
    for (int i = 0; i < selectedDieCount*4; i++) {
      for (int j = 0; j < variationCount; j++) {
        variationValues[i][j] = varsValues[i%minVarCount][j];
      }
    }
    break;
}

message("Parameter calculation complete");

/* List all parameter combinations if verbose is true. */
if (verbose) {
  message("parameter combinations: ");
  for (int i = 0; i < selectedDieCount*4; i++) {
    String col = "";
```

```
    for (int j = 0; j < variationCount; j++) {
      col += varsMark[j]+", "+variationValues[i][j]+" \t";
    }
    message("cap "+i+": \t"+col);
  }
}

/* Populate dies with capacitors. */

/* Input variables from the COMSOL model set via application. */

double textPixWidth = model.param().evaluate("textPixWidth", "um");

double markWidth = model.param().evaluate("markWidth", "um");
double clearance = model.param().evaluate("clearance", "um");

double capWidth = model.param().evaluate("Wt");

double centerClearance = model.param().evaluate("centerClearance", "um");

/* Calculate and set capacitor height to accomodate for center clearance on the die. */
double capHeight = .5*wDie-centerClearance;
with(model.param());
  set("Ht", capHeight);
endwith();

message("Populating dies");
for (int i = 0; i < selectedDieCount; i++) {

  /* Clear previous dies to prevent COMSOL from slowing down. */
  clearGeometry();

  /* Get die coordinates */
  double x = selectedDiePositions[i][0];
  double y = selectedDiePositions[i][1];

  /* Create string ID based on coordinates for logging and markings. */
  String id = "die_"+Math.round(x/(wDie+dDie))+"_"+Math.round(y/(wDie+dDie));

  if (verbose) message("Populating die: "+Math.round(x/(wDie+dDie))+
                     ", "+Math.round(y/(wDie+dDie)));

  /*
   * Constant angles and positions used to place the capacitors on the die in
   * with radial symmetry.
   */
  double[] angles = {0.0d, 90.0d, 180.0d, 270.0d};
  double[][] capPositions = {{.5*wDie, 0}, {wDie, .5*wDie}, {(1-.5)*wDie, wDie}, {0, (1-.5)*
      wDie}};

  double[] markAngles = {-90.0d, 0, 90.0d, 180.0d};
  double[][] markPositions = {{.5*wDie+capWidth+clearance, capHeight},
                             {wDie-capHeight, .5*wDie+capWidth+clearance},
                             {wDie-(.5*wDie+capWidth+clearance), wDie-capHeight},
                             {capHeight, wDie-(.5*wDie+capWidth+clearance)}};

  /* Generate the capacitors on the die */
  for (int j = 0; j < 4; j++) {
    String mark = Math.round(x/(wDie+dDie))+", "+Math.round(y/(wDie+dDie))+", "+j; ;

    /* Set the parameters for the capacitor according to the parameter variation */
    for (int k = 0; k < variationCount; k++) {
      if (variationMode != 0) {
        /* get the parameter value and set it in the model */
        double value = variationValues[(i*4+j)][k];
        with(model.param());
          set(varsName[k], value);
        endwith();

        /* Append the value to the marking if marking specified */
        if (!varsMark[k].isEmpty()) {
```

```
          mark = mark+" "+varsMark[k]+" ";
          /* format integer or float/double */
          if (variationValues[(i*4+j)][k] == (int) variationValues[(i*4+j)][k])
            mark += variationValues[(i*4+j)][k];
          else
            mark += String.format("%.2f", variationValues[(i*4+j)][k]);
        }
      };
    }

    /* Append the group marking */
    if (!groupMark.isEmpty()) {
      mark += " "+groupMark;
    }

    /* Generate marking text */
    generateText(mark, x+markPositions[j][0], y+markPositions[j][1], textPixWidth, id+"_"+j,
        false, markAngles[j], capHeight-markWidth*2);

    /* Generate the capacitor. */
    generateCapacitor(id+"_cap"+j, x+capPositions[j][0], y+capPositions[j][1], angles[j]);

    if (verbose) message("placed cap \t"+(i*4+j+1)+"/"+(selectedDieCount*4)+
                         ". \tMark: "+mark+" "+groupMark);
  }

  /* Generate the cutmarks showing the die outlines */
  generateCutmarks(id, x, y, wDie, markWidth, capWidth, clearance);

  /* Build die */
  if (verbose) message("building die");
  model.component("comp1").geom("geom1").run("fin");

  if (export)
  {
    /* Export die as .DXF*/
    if (verbose) message("exporting die");
    model.component("comp1").geom("geom1").exportFinal(exportPath+"/"+id+".dxf");
  }

}
message("Die population complete.");
```

A.7.7. Generate Die

```
/**
 * Generate Die
 *
 * Method to generate a single die as a preview for the die generation
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Store the current variation mode */
int storedPlacementMode = placementMode;
String storedPositions = positions;
boolean storedExport = export;

/* Temporarily set the variation settings to generate just one die */
placementMode = 0; // custom placement
positions = "0,0";
export = false;

/* Generate the die using the generateDies method */
generateDies();

/* Restore the variation settings */
placementMode = storedPlacementMode;
positions = storedPositions;
export = storedExport;
```

A.7.8. Clear Geometry

```
/**
 * Clear Geometry
 *
 * Method to quickly clear the selections and geometry
 *
 * @Author: Jasper Rietveld
 * @Student number: 4581881
 */

/* Delete selections */
model.component("comp1").geom("geom1").selection().clear();

/* Delete geometry */
model.geom("geom1").feature().clear();
```

A.8. Process flowcharts

This section contains the flowcharts used during the fabrication process. The flowcharts list the detailed processing steps and the equipment used.

A.8.1. Preparation of wafers with SiC layer

Annotated flowchart of the process supplying the wafers for this project.

FLOWCHART SiC-LS-B doped

Starting material

Wafer number	Wafer type
939	$1.5\,\Omega$.cm, P-type, $< 100 >$
947	$1.5\,\Omega$.cm, P-type, $< 100 >$
997	$1.5\,\Omega$.cm, P-type, $< 100 >$
996	$1.5\,\Omega$.cm, P-type, $< 100 >$

Novellus SiC, low stress, B-doped

Cr/Au lift-off process

1. DIBAR

200Å in A1

2. IMPLANTATION

B+, 15keV, 3E15, BACKSIDE

3. STRIP OXIDE

0,5%HF, HFOOB

4. MEASURE

Flexus: First meas – wafer 939, 947, 997, 996

5. DEPOSITION

Novellus: 500nm #SiC_4%B on wafer 939, 947 for 54 sec Novellus: 500nm #SiC_8%B on wafer 997, 996 for 50 sec

6. MEASURE THICKNESS

SP Leitz: Thickness meas

7. MEASURE STRESSES

Flexus: Single meas

8. MEASURE RSQ

Rsq meas

Wfr no.	SP Leitz (nm)	Flexus (MPa)	$R\square(\Omega/\square)$
939	521+/-2	-248 (Comp)	∞
947	521+/-3	-246 (Comp)	∞
997	518+/-4	-198 (Comp)	∞
996	519+/-5	-198 (Comp)	∞

9. Anneal

Anneal A1: recipe#tst-var, 1000C, 35min

10. MEASURE THICKNESS

SP Leitz: Thickness meas

11. MEASURE STRESSES

Flexus: Single meas

12. MEASURE RSQ

Rsq meas

Wfr no.	SP Leitz (nm)	Flexus (MPa)	$R\square(\Omega/\square)$
939	409+/-1	+2129 (Tens)	∞
947	408+/-1	+2116 (Tens)	∞
997	360+/-2	+2270 (Tens)	∞
996	360+/-2	+2193 (Tens)	∞

Cracks visible in all films

10. CLEANING

Cleaning: R+K HNO3

10. HF DIP

0,5% HF dip, 4 min

10. ALUMINIUM DEPOSITION

Sigma: 1,4mu Al/Si at 350C, BACKSIDE

A.8.2. Fabrication of Au/Cr planar capacitors

FLOWCHART
Fabrication of Au/Cr planar capacitors on porous Silicon Carbide

---

Starting material

Wafer number	Wafer type
939	$1.5\,\Omega$.cm, P-type, $< 100 >$
947	$1.5\,\Omega$.cm, P-type, $< 100 >$
997	$1.5\,\Omega$.cm, P-type, $< 100 >$
996	$1.5\,\Omega$.cm, P-type, $< 100 >$

The process will be carried out on wafers resulting from a previous project (Project: 20191108-0357, Run number SB2413) with wafer numbers 939, 947, 997, and 996. Carried out steps include forming a SiC layer on the top side of the wafer and depositing an Al layer on the backside.

Cr/Au lift-off process

1. RINSING AND DRYING

Cleaning — 10 minutes in 99% HNO_3 at room temperature at green metal cleaning line in CR100. Use the carrier with a red dot and short handle. (Do not forget to clean drops between baths with DI water)

QDR — Rinse in DI water until resistivity is 5 MΩ.

Drying — Use the Semitool ”rinser/dryer” with the standard program, and the white carrier with a red dot. Use the bottom SRD (black button)

2. SPIN COATING

Coat wafers with negative photoresist using the EVG120. Program CO –Nlof - 3.0µm.

3. EXPOSURE

Processing will be performed using the SUSS MicroTec MA/BA8 mask aligner. Use the mask with the label “ICTA”. Exposure time: 5,5 sec (based on 15 mW/cm2Hg-lamp in-tensity, recalculate for lamp used in SUSS accordingly).

4. BAKING

Bake resist using the EVG120. Program: Only – X-link bake

5. DEVELOPMENT

Development of the resist on the EVG120. Program: Dev – lift-off

6. INSPECTION

Using the microscope at the Lithography bay. Visual inspection of edges and open areas of resist structures. Inspect the devices with the smallest dimensions extra carefully.

7. O2 PLASMA FLASH

On the TEPLA Plasma 300. Program 2 (flash)

8. DEPOSITION

Deposition of 15 nm Cr/185 nm Au using the CHA. Use the dedicated Au half-dome, and make sure to inspect it for flakes. Wafers are contaminated after this step, use dedicated material.

9. LIFT-OFF PROCEDURE

The lift-off will be performed at the SAL lab in the fume hood dedicated for organics.

- Heat up beaker with DI-water to 80°C
- Transfer water to HBM Ultrasonic Bath
- Switch on Temperature to 70°C
- Heat up beaker with NMP to 70°C
- Transfer heated NMP to rectangle beaker that fits basket
- Add lift-off water to rectangle beaker
- Switch on the HBM Ultrasonic bath, leave for 15 min
- periodically remove lift-off residus with cotton bud and rotate wafer

10. RINSING AND DRYING

Rinsing	Rinse in DI-water for 5 minutes.
Drying	Using the spin dryer with the contaminated chuck.

11. INSPECTION

Use a blue napkin under the wafer against contamination.

ELECTROCHEMICAL ETCHING

12. DICING

The processed wafers must be diced to create 1cm by 1cm chips as the wafer can not be etched at once to get a consistent current density.

13. ATTACH CHIPS TO PCBS

A selection of the 1cm by 1cm chips is glued to custom PCB etch holders (stored in vacuum sealed bag). This is done using a conductive silver epoxy which dries based on acetone evaporation. The glue is applied using a small cotton swab and left to dry in the fume hood.

14. APPLY WAX AS ETCH RESIST

Areas on the PCB that should not be etched and have a low etch resistance (chip sides, pads) are to be covered in Black Wax (Apezion Wax W, sticks, stored in vacuum sealed bag). This is done in a fume hood in the MEMS lab.

Preparation
Fill a small container with a few centimetres of toluene.
Prepare a shallow tray.
Break or cut off small pieces of a stick of Black Wax.
Dissolve enough Black Wax in the toluene to form a runny paste.

Application
Use a cotton swab to apply the toluene paste to the areas of the PCB to be protected.
Place the PCB on the tray to dry .
Periodically rotate the PCB
When the solvent has evaporated from the PCB.

inspection
Using a multimeter, test the pads for proper insulation.

Note: glassware used for this step will be heavily stained. Thorough rinsing with more solvent is required for cleanup unless disposable hardware is used.

15. ELECTROCHEMICAL ETCHING

Carried out at the dedicated HF wet bench in the SAL lab. Take caution to properly label equipment. Take extra caution in handling chemicals.

Preparation
Prepare a container with DI water.
Prepare a second etch resistant container with about 2cm of the HF (enough to cover chip on lower part of PCB and to cover electrode).
Attach the negative clamp of the current source to the platinum electrode. Attach platinum electrode to the HF container so that it is submerged and positioned against the side wall.

Etching
Attach the other clamp of the current source to the large pad on the PCB.
Set the current source to the desired etching current.
Start a timer for the etching duration.
Attach the PCB to the HF container so that the chip is submerged and positioned against the side wall opposite from the electrode (keep the distance consistent between batches, too close together and etching will be uneven. HF solution should be conductive enough to allow distance without a large resistance increase.)
Check that the voltage is within compliance and current is not deviate while waiting. (battery voltage will reduce and. resistance may change). Once timer ends, remove pcb from HF container.
Thoroughly rinse PCB in DI water (can use a running tap into a large container).
disconnect current source clamp from PCB.
Leave PCB on cleanroom paper towel to dry.
Set current source to 0.0A.
repeat etching step for further chips/PCBs in batch.

A.9. MATLAB Simulation Code

In this section, the MATLAB code for processing the measurements is listed.

A.9.1. Plot Measurements

```
%{
SCRIPT NAME:
    plotMeasurements

DESCRIPTION:
    Generates the LaTex for a tabular comparing measurement results and
    the simulations

INPUT:
    measurementPaths specifies output files from the measurements along
    with the corresponding width and gap sizes. The files for these
    measurements should be in the active workspace.



OUTPUT:
    The LaTeX for a tabular is printed to the command window.

AUTHOR:
    Jasper Rietveld
    Student number 4581881
%}

%% Load the measurement data

% Define the paths to the measurement files and the corresponding width and
% gap values
measurementPaths = {2, 2, "measurements/1_2_0_1MHz_remeas.mdm";
                    2, 5, "measurements/1_2_1_1MHz_recal.mdm";
                    2, 10, "measurements/1_0_2_1MHz.mdm";
                    5, 2, "measurements/0_1_0_1MHz.mdm";
                    5, 5, "measurements/0_1_1_1MHz.mdm";
                    5, 10, "measurements/0_1_2_1MHz_recal.mdm";
                    5, 20, "measurements/3_1_2_1MHz_cal.mdm";
                    10, 2, "measurements/0_1_3_1MHz.mdm";
                    10, 5, "measurements/3_1_0_1MHz.mdm";
                    10, 10, "measurements/3_1_1_1MHz.mdm";
                    10, 20, "measurements/3_1_3_1MHz.mdm";
                    20, 5, "measurements/1_1_1_1MHz_recal.mdm";
                    20, 10, "measurements/1_1_2_1MHz.mdm";
                    20, 20, "measurements/1_1_3_1MHz.mdm"};

% Get the unique width and gap values
widths = unique(cell2mat(measurementPaths(:,1)));
gaps = unique(cell2mat(measurementPaths(:,2)));

% Create matrices for the measured capacitance and resistance values
capSizes = zeros(length(gaps), length(widths));
resSizes = zeros(length(gaps), length(widths));


for i = 1:size(measurementPaths,1)
    % Extract the data from the measurement files
    data = readtable(measurementPaths{i,3}, 'FileType', 'text');
    cap = data(data.x_V_bias == 0,2).(1)(1);
    res = data(data.x_V_bias == 0,3).(1)(1);

    % Set the corresponding entry in the capacitance table
    width = measurementPaths{i,1};
    gap = measurementPaths{i,2};
    capSizes(gaps==gap, widths==width) = cap;
    resSizes(gaps==gap, widths==width) = res;
end

% Calculate capacitor areas based on values put on wafer

%% Load and process the simulation data

% Specify the electrode counts for the measured devises
```

```
capElectrodeCounts = [20 20 20 0
                      20 20 20 10
                      20 20 20 10
                      0  10 10 10];

% Wavelengths in um
capWavelengths = 2*(widths'+gaps);

% cap area in cm^2 for the measured devices
capDimensions = (capElectrodeCounts.*capWavelengths*1e-4); % width in cm
capDimensions = capDimensions.*(2000*capWavelengths*1e-6); % length in cm

% Get capacitance per area from sims
simulationData = readtable('updated_paSiC_sweep_10x.txt');
% Var1: width, Var2: gap,
% Var3: ammonia concentration, Var5: capacitance
filteredData = simulationData(...
    arrayfun(@(x) any(x == widths), simulationData.Var1) &...
    arrayfun(@(x) any(x == gaps), simulationData.Var2) &...
    simulationData.Var3 == 100, [1 2 5]);

% Create a matrix to hold the simulated capacitance values
simCapSizes = zeros(length(gaps), length(widths));

% Fill the matrix in accordance to the matrix for measured data
for i = 1:size(filteredData,1)
    width = filteredData{i,1};
    gap = filteredData{i,2};
    cap = filteredData{i,3};
    simCapSizes(gaps==gap, widths==width) = cap;
end

% Use the
simCapSizes = simCapSizes.*capDimensions;

%% Output a table with the comparison

% Comparison table
disp("sensitivity table: ");
generateTableString(...
    widths, "%.2g", "[\si{\micro\meter}]",...
    gaps, "%.2g", "[\si{\micro\meter}]",...
    capSizes./1e-12, "\SI{%.3g}{\pico\farad}",...
    simCapSizes./1e-12, "{\color{tudelft-green}\SI{%.3g}{\pico\farad}}",...
    capSizes./simCapSizes, "{\color{tudelft-warm-purple}(%.3g)}");
disp("");


function generateTableString(widths, widthformat, widthunitformat, ...
    gaps, gapformat, gapunitformat, mainval, mainvalformat,...
    subval1, subval1format, subval2, subval2format)
    %{
    FUNCTION NAME:
      generateTableString

    DESCRIPTION:
      Fills a template table with values and formats for these values
      passed as function parameters.

    INPUT:
      widths - A vector with the widths.
      widthformat - LaTeX string containing a formatSpec
      widthunitformat - LaTeX string to put after the column heading
      gaps - A vector with the gaps
      gapformat - LaTeX string containing a formatSpec
      gapunitformat - LaTeX string to put after the row heading
      mainval - matrix for the main value to insert for each W and G
      mainvalformat - LaTeX string with a formatSpec for the main value
      subval1 - matrix for a second value to insert for each W and G
      subval1format - LaTeX string with a formatSpec for the second value
      subval2 - matrix for a third value to insert for each W and G
```

```
    subval2format - LaTeX string with a formatSpec for the third value

  OUTPUT:
    The LaTeX for the table with the values inserted is printed to
    the command window for copying into the report

  ASSUMPTIONS AND LIMITATIONS:
    The script requires the presence of an exported COMSOL table in the
    active workspace, as well as the presence of a valid LaTeX table
    template. Format strings should contain one formatSpec. LaTeX can be
    used in the format strings.

  AUTHOR:
    Jasper Rietveld
    Student number 4581881
  %}

  % Read file with the LaTeX format to insert the values into
  tableFormat = readlines("tableFormat4x4.txt");

  % Insert unit strings
  tableFormat = strrep(tableFormat, "widthformat", widthformat);
  tableFormat = strrep(tableFormat, "widthunitformat", widthunitformat);
  tableFormat = strrep(tableFormat, "gapformat", gapformat);
  tableFormat = strrep(tableFormat, "gapunitformat", gapunitformat);
  tableFormat = strrep(tableFormat, "mainvalformat", mainvalformat);
  tableFormat = strrep(tableFormat, "subval1format", subval1format);
  tableFormat = strrep(tableFormat, "subval2format", subval2format);

  % Replace % with %% to allow printing with fprintf
  tableFormat = strrep(tableFormat, "\%", "\%%");

  % Replace \ with \\ to allow printing with fprintf
  tableFormat = strrep(tableFormat, "\", "\\");


  % Add newlines to allow printing with fprintf
  tableFormat = strjoin(tableFormat, "\n");
  tableFormat = strcat(tableFormat, "\n");

  % Insert values
  fprintf(tableFormat, ...
      widths(1), widths(2), widths(3), widths(4),...
      gaps(1), mainval(1,1), mainval(1,2), mainval(1,3), mainval(1,4),...
      subval1(1,1), subval2(1,1), subval1(1,2), subval2(1,2),...
      subval1(1,3), subval2(1,3), subval1(1,4), subval2(1,4),...
      gaps(2), mainval(2,1), mainval(2,2), mainval(2,3), mainval(2,4),...
      subval1(2,1), subval2(2,1), subval1(2,2), subval2(2,2),...
      subval1(2,3), subval2(2,3), subval1(2,4), subval2(2,4),...
      gaps(3), mainval(3,1), mainval(3,2), mainval(3,3), mainval(3,4),...
      subval1(3,1), subval2(3,1), subval1(3,2), subval2(3,2),...
      subval1(3,3), subval2(3,3), subval1(3,4), subval2(3,4),...
      gaps(4), mainval(4,1), mainval(4,2), mainval(4,3), mainval(4,4),...
      subval1(4,1), subval2(4,1), subval1(4,2), subval2(4,2),...
      subval1(4,3), subval2(4,3), subval1(4,4), subval2(4,4)...
      );
end
```

Bibliography

[1] R. Ma, K. Li, Y. Guo, B. Zhang, X. Zhao, S. Linder, C. Guan, G. Chen, Y. Gan, and J. Meng, "Mitigation potential of global ammonia emissions and related health impacts in the trade network." *Nature communications*, vol. 12, p. 6308, 2021.

[2] B. Timmer, W. Olthuis, and A. van den Berg, "Ammonia sensors and their applications - a review," *Sensors and actuators. B: Chemical*, vol. 107, no. 2, pp. 666–677, Jun. 2005.

[3] N. R. C. U. C. on Acute Exposure Guideline Levels., *Acute Exposure Guideline Levels for Selected Airborne Chemicals: Volume 6.* Washington (DC): National Academies Press (US), 2008. [Online]. Available: https://www-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/books/NBK207883/

[4] E. Connolly, B. Timmer, H. Pham, J. Groeneweg, P. Sarro, W. Olthuis, and P. French, "A new ammonia sensor based on a porous sic membrane," vol. 2, pp. 1832–1835 Vol. 2, 2005.

[5] N. Docquier and S. Candel, "Combustion control and sensors: a review," *Progress in Energy and Combustion Science*, vol. 28, no. 2, pp. 107–150, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360128501000090

[6] C. G. Rodríguez, M. I. Lamas, J. d. D. Rodríguez, and A. Abbas, "Possibilities of ammonia as both fuel and nox reductant in marine engines: A numerical study," *Journal of Marine Science and Engineering*, vol. 10, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2077-1312/10/1/43

[7] P. Dimitriou and R. Javaid, "A review of ammonia as a compression ignition engine fuel," *International Journal of Hydrogen Energy*, vol. 45, no. 11, pp. 7098–7118, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360319920300124

[8] M. J. Lefferts and M. R. Castell, "Ammonia breath analysis," *Sens. Diagn.*, vol. 1, pp. 955–967, 2022. [Online]. Available: http://dx.doi.org/10.1039/D2SD00089J

[9] D. Kwak, Y. Lei, and R. Maric, "Ammonia gas sensors: A comprehensive review," *Talanta*, vol. 204, pp. 713–730, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0039914019306599

[10] G. Korotcenkov and B. K. Cho, "Porous semiconductors: Advanced material for gas sensor applications," *Critical Reviews in Solid State and Materials Sciences*, vol. 35, no. 1, pp. 1–37, 2010. [Online]. Available: https://doi.org/10.1080/10408430903245369

[11] E. Connolly, P. French, H. Pham, and P. Sarro, "Relative humidity sensors based on porous polysilicon and porous silicon carbide," vol. 1, pp. 499–502 vol.1, 2002.

[12] L. Seals, J. L. Gole, L. A. Tse, and P. J. Hesketh, "Rapid, reversible, sensitive porous silicon gas sensor," *Journal of Applied Physics*, vol. 91, no. 4, pp. 2519–2523, 2002. [Online]. Available: https://doi.org/10.1063/1.1436556

[13] A. Chaillou, J. Charrier, N. Lorrain, M. Sarret, and L. Haji, "Porous silicon ammonia gas sensor." *SPIE Photonics Europe 2006*, 2006. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00153811

[14] W. Daves, A. Krauss, N. Behnel, V. Häublein, A. Bauer, and L. Frey, "Amorphous silicon carbide thin films (a-sic:h) deposited by plasma-enhanced chemical vapor deposition as protective coatings for harsh environment applications," *Thin Solid Films*, vol. 519, no. 18, pp. 5892–5898, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040609011006353

[15] Y. Xu, L. Cheng, and L. Zhang, “Composition, microstructure, and thermal stability of silicon carbide chemical vapor deposited at low temperatures,” *Journal of Materials Processing Technology*, vol. 101, no. 1, pp. 47–51, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924013600004283

[16] E. Connolly, B. Timmer, H. Pham, J. Groeneweg, P. Sarro, W. Olthuis, and P. French, “A porous sic ammonia sensor,” *Sensors and Actuators B: Chemical*, vol. 109, no. 1, pp. 44–46, 2005, e-MRS Fall Meeting. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925400505002649

[17] G. Gautier, T. Defforge, G. Gommé, D. Valente, and D. Alquier, “Electrochemical formation of porous silicon carbide for micro-device applications,” *Materials Science Forum*, vol. 924, pp. 943–946, 06 2018.

[18] D. T. Cao, C. T. Anh, N. T. T. Ha, H. T. Ha, B. Huy, P. T. M. Hoa, P. H. Duong, N. T. T. Ngan, and N. X. Dai, “Effect of electrochemical etching solution composition on properties of porous SiC film,” *Journal of Physics: Conference Series*, vol. 187, p. 012023, sep 2009. [Online]. Available: https://doi.org/10.1088/1742-6596/187/1/012023

[19] M. Leitgeb, C. Zellner, M. Schneider, S. Schwab, H. Hutter, and U. Schmid, “Metal assisted photochemical etching of 4h silicon carbide,” *Journal of Physics D: Applied Physics*, vol. 50, 08 2017.

[20] A. Boukezzata, H. Menari, and S. Kaci, “Al-assisted photochemical etching of a-sic thin films for nh 3 sensor,” *Acta Physica Polonica A*, vol. 137, pp. 454–457, 04 2020.

[21] J. Senthilnathan, A. Selvaraj, J. Lee, K.-H. Kim, and M. Yoshimura, “Formation of highly porous electrochemically etched silicon carbide: A novel reusable adsorbent for air purification technology,” *Journal of Cleaner Production*, vol. 218, pp. 521–528, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652619304123

[22] S. Chen, C. L. Chen1, Z. L. Huang, and J. S. Huang, “Porosity control of porous silicon carbide ceramics,” *Journal of the European Ceramic Society*, vol. 29, pp. 2867–2872, 2009.

[23] Z. Li, Z. Zhang, W. Zhao, X. Li, G. Han, and J. Zhang, “A simple method to control the pore structure and shape of freeze-cast porous sic ceramics,” *Ceramics International*, vol. 46, no. 16, Part A, pp. 26 078–26 084, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0272884220321015

[24] A. Rivadeneyra, J. Fernández-Salmerón, J. Banqueri, J. A. López-Villanueva, L. F. Capitan-Vallvey, and A. J. Palma, “A novel electrode structure compared with interdigitated electrodes as capacitive sensor,” *Sensors and Actuators B: Chemical*, vol. 204, pp. 552–560, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925400514009782

[25] T. Chen and N. Bowler, “Design of interdigital spiral and concentric capacitive sensors for materials evaluation,” vol. 1511, pp. 1593–1600, 01 2013.

[26] P. Sarafis and A. Nassiopoulou, “Dielectric properties of porous silicon for use as a substrate for the on-chip integration of millimeter-wave devices in the frequency range 140 to 210 ghz,” *Nanoscale Research Letters*, vol. 4022, 08 2014.

[27] R. Igreja and C. Dias, “Extension to the analytical model of the interdigital electrodes capacitance for a multi-layered structure,” *Sensors and Actuators A: Physical*, vol. 172, pp. 392–399, 2011.

[28] E. G. Hill and A. P. Sirkar, “The electric conductivity and density of solutions of hydrogen fluoride.” *Proc. R. Soc. Lond. A*, vol. 83, p. 130–148, 1909.

[29] J. Zhu, G. Riskowski, and R. Mackie, “A laboratory study on metal corrosion by ammonia gas,” *Transactions - American Society of Agricultural Engineers: General Edition*, vol. 42, no. 3, pp. 783–787, 1999.