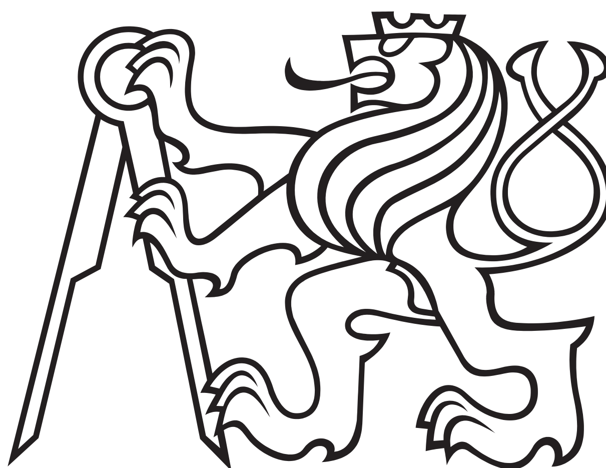
Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Microelectronics


Electronics and Communication

Specialization: Electronics



# MASTER'S THESIS

# Automatic 3D Model Creation of Critical Parts of Integrated Circuits


Author: Bc. Martin Šťastný

Supervisor: doc. Ing. Jiří Jakovenko, Ph.D.


Prague 2021

# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Šťastný** Jméno: **Martin** Osobní číslo: **437455**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Elektronika**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Návrh automatického generování 3D modelů kritických částí integrovaných obvodů**

Název diplomové práce anglicky:

**Automatic 3D Model Creation of Critical Parts of Integrated Circuits**

Pokyny pro vypracování:

1. Prostudujte polovodičové struktury používané v integrovaných obvodech (IC) a popište fyzikální jevy vyskytující se v těchto strukturách.
2. Definujte metodiku extrakce dat z 2D editoru topologie kritické části integrovaného obvodu.
3. Navrhněte následné zpracování získaných dat pro vytvoření přesné 3D vizualizace analyzovaných polovodičových struktur.
4. Importujte zavedený model do softwaru T-CAD a proveďte nezbytné simulace. Zhodnoťte dosažené výsledky

Seznam doporučené literatury:


1. BRAMBILLA, Angelo, Paolo MAFFEZZONI, Luca BORTESI and Loris VENDRAME. Measurements and Extractions of Parasitic Capacitances in ULSI Layouts. IEEE Transactions on Electron Devices. 2003, vol. 50, no. 11, p. 2236 – 2247.
2. FANG, Robert Ching-Yuh and John L. MOLL. Latchup Model for the Parasitic p-n-p-n Path in Bulk CMOS. IEEE Transactions on Electron Devices. 1984, vol. 31, no. 1, p. 113 – 120.
3. SCHRÖTER, Michael and David J. WALKEY. Physical Modeling of Lateral Scaling in Bipolar Transistors. IEEE Journal of Solid-State Circuits. 1996, vol. 31, no. 10, p. 1484 – 1492.
Silvaco TCAD Simulation Tool, DeckBuild Deck Editor Version 4.4.3.R, Synopsys Inc., 2018.


Jméno a pracoviště vedoucí(ho) diplomové práce:

**doc. Ing. Jiří Jakovenko, Ph.D., katedra mikroelektroniky FEL**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **11.02.2020** Termín odevzdání diplomové práce: ______________

Platnost zadání diplomové práce: **30.09.2021**

| doc. Ing. Jiří Jakovenko, Ph.D. | prof. Ing. Pavel Hazdra, CSc. | prof. Mgr. Petr Páta, Ph.D. |
|---|---|---|
| podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | podpis děkana(ky) |

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

| . | |
|---|---|
| Datum převzetí zadání | Podpis studenta |



© ČVUT v Praze, Design: ČVUT v Praze, VIC

**Declaration:**

I declare that this work is all my own work, and I have cited all sources I have used in the bibliography.

In Prague, date ............. ........................................

Author's signature

**Acknowledgments:**

I wish to thank my supervisor doc. Ing. Jiří Jakovenko, Ph.D. for the opportunity he gave me together with his time and experiences.

A special thanks to supervisors-specialists from the STMicroelectronics company, namely to Ing. Dalibor Barri, Ing. Vlastimil Kotě, Ph.D., Ing. Miloš Vacula, and Ing. Patrik Vacula, Ph.D. for countless hours, experiences, practical tips and patience throughout the process.

I would like to acknowledge and thank the Department of microelectronics for allowing me to conduct my research and provided any assistance requested.

Bc. Martin Šťastný

# Abstract

This work is focused on the processing of 2D data from the standard format for the design of integrated circuits (GDSII) and its possible simulation in TCAD (technological computer-aided design) tool for the simulation of semiconductor structures. For a better understanding, this work also provides a 2D layout and a subsequent 3D model. Both, the layout and the model try to approach the real production processes of integrated circuits (IC). This work is the possibility of creating a 3D model and simulation of the circuits using TCAD from Silvaco, Inc. The implementation of this work is mainly done using free software, such as Blender, and Python programming language.

*Keywords:* 3D modeling, integrated circuits, BCD and CMOS processes, Silvaco TCAD simulations, Python programming

# Abstrakt

Tato práce je zaměřena na zpracování 2D dat ze standardního formátu pro návrh integrovaných obvodů (GDSII) a jejich možnou simulaci v TCAD (technological computer-aided design; překlad "techonolgický, počítačem podporovaný návrh") programu určenému pro simulaci polovodičových struktur. Pro lepší porozumění tato práce poskytne i 2D layout a následný 3D model. Layout i model se snaží přiblížit skutečným výrobním procesům integrovaných obvodů. Výsledkem této práce je možnost vytvoření 3D modelu a simulace daných obvodů za pomoci TCAD od společnosti Silvaco, Inc.. Pro realizaci této práce je používaný zejména svobodný software, např. Blender, a programovací jazyk Python.

*Klíčová slova:* 3D modelování, integrované obvody, BCD a CMOS procesy, Silvaco TCAD, Python

# Contents

**Abstract** v

**List of Abbreviations** 3

**Introduction** 5

**1 Literature Review** 7

1.1 State of the Art . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

1.1.1 Visualization Tools . . . . . . . . . . . . . . . . . . . . . 7

1.1.2 Simulation Tools . . . . . . . . . . . . . . . . . . . . . . 10

1.2 Technology of Integrated Circuits . . . . . . . . . . . . . . . . . . 13

1.2.1 Steps in Planar Process . . . . . . . . . . . . . . . . . . . 13

1.2.2 CMOS Process . . . . . . . . . . . . . . . . . . . . . . . 26

1.2.3 BCD Process . . . . . . . . . . . . . . . . . . . . . . . . 27

1.2.4 Parasitic Phenomena . . . . . . . . . . . . . . . . . . . . 28

1.3 File Formats . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30

1.3.1 Graphic Design System . . . . . . . . . . . . . . . . . . . 30

1.3.2 2D Formats . . . . . . . . . . . . . . . . . . . . . . . . . 31

1.3.3 3D Formats . . . . . . . . . . . . . . . . . . . . . . . . . 31

**2 Methods** 33

2.1 Definition File . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

2.2 2D Part . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

2.2.1 Definitions and GDSII Mapping . . . . . . . . . . . . . . . 37

2.2.2 2D Logic Operations . . . . . . . . . . . . . . . . . . . . 38

2.2.3 Generation of SVG . . . . . . . . . . . . . . . . . . . . . 39

2.3 3D Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

2.3.1 Blender Connector . . . . . . . . . . . . . . . . . . . . . 40

2.3.2 Generator of 3D View . . . . . . . . . . . . . . . . . . . . 40

2.4 TCAD Export . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

2.4.1 Generation of TCAD Input . . . . . . . . . . . . . . . . . 43

**3 Results** **45**
3.1 Visualization . . . . . . 46
3.1.1 Layout View . . . . . . 46
3.1.2 Generated 3D Structure . . . . . . 47
3.2 TCAD Data . . . . . . 49
3.3 Time Complexity . . . . . . 50
3.3.1 Comparison with Manual Work . . . . . . 50
3.3.2 Layout Size Influence . . . . . . 51

**4 Future Work** **53**
4.1 3D TCAD . . . . . . 53

**5 Discussion** **55**
5.1 Visualization . . . . . . 56
5.2 TCAD . . . . . . 57
5.3 Time Complexity . . . . . . 57

**Conclusion** **59**

**Bibliography** **61**

**List of Figures** **67**

# List of Abbreviations

| | |
|---|---|
| 2D | 2-dimensional |
| 3D | 3-dimensional |
| °C | Celsius degree |
| AI | Vector graphic format used by Adobe |
| ASCII | American standard code for information interchange |
| $\beta$ | Current factor of bipolar junction transistor |
| BCD | Bipolar-CMOS-DMOS (process) |
| BEOL | Back-end-of-line (part of manufacturing process) |
| CMOS | Complementary metal-oxide-semiconductor |
| CMP | Chemical-mechanical polishing |
| CAD | Computer-aided design (software) |
| CVD | Chemical vapor deposition |
| DMOS | Double-diffused metal-oxide-semiconductor |
| DTI | Deep trench isolation |
| DXF | Vector graphic format |
| EPS | Vector graphic format |
| FEOL | Front-end-of-line (part of manufacturing process) |
| FET | Field-effect transistor |
| FOX | Field oxide |
| GB | Gigabyte (unit of memory) |
| GDS or GDSII | Graphic design system format |
| GDML | Geometry description markup language format |
| GOX | Gate oxide |
| I-V | Current-Voltage (characteristic) |
| IC | Integrated circuit |
| LDD | Lightly doped drain |
| LOCOS | Local oxidation of silicon |
| MOS | Metal-oxide-semiconductor |
| $n^+$ | High doping N-type semiconductor |

| | |
|---|---|
| $n^-$ | Low doping N-type semiconductor |
| NAND | Logic operation |
| NMOS | N-type metal-oxide-semiconductor |
| OBJ | 3D graphic format |
| $p^+$ | High doping P-type semiconductor |
| $p^-$ | Low doping P-type semiconductor |
| PDF | Portable document format |
| PMOS | P-type metal-oxide-semiconductor |
| PN | Semiconductor p-n junction |
| POV | POV-Ray description file format |
| PVD | Physical vapor deposition |
| RAM | Random access memory |
| RGB | Color space format (red, green, blue) |
| RIE | Reactive ion etching |
| SOI | Silicon-on-insulator |
| STI | Shallow trench isolation |
| STL | Format for 3D printers |
| STR | Silvaco TCAD structure format |
| SVG | Scalable vector graphic format |
| TCAD | Technology computer-aided design |
| TDR | Synopsys structure format |
| TIF3D | Cogenda structure format |
| U3D | Universal 3D format |
| UV | Ultra-violet (light) |
| XML | Extensible markup language |

# Introduction

These days layout design reached the point, at which is needed to stack component structures on top of each other to save space on a chip. This development step has significant advantages and can help with higher integration of larger structures into smaller areas. It also may cause issues between components in separate layers, e.g., parasitic PN structures, interferences of electromagnetic fields and heat influences between components.

This thesis is focused on generating the 3D model considering technological and physical parameters. The output of this thesis is a 3D model generated from the GDSII file by Virtuoso (Cadence Virtuoso Layout Suite), and a possibility to import the model to TCAD (Technology Computer-Aided Design) using a Python program.

The output of this thesis is 3D visualization generated with a program written in Python. Parts of this thesis can be used as documentation for the program and libraries that have been created. Another output of this program is a file executable in TCAD software that allows setting and running various simulations.

Throughout the whole process, there are many issues to solve. Starting with translation from GDSII to a human-readable format and ending with the generation of TCAD structures. Everything in this thesis is based on 2D data from GDSII. This data must be processed and edited to serve as a template for easier and accurate 3D structure generation, the verge of the real process.

# 1. Literature Review

## 1.1 State of the Art

The development of ICs is with decreasing the size of transistors, and using the third dimension is getting delicate, and many errors can occur. Visualization of these circuits in 3D can reveal possible critical parts. Localization and solving potential issues, these parts may cause, can be simplified by simulations.

Most of the existing tools simply stretch the 2D layout and create a 3D model that serves only for visualization purposes. Only a few tools are capable of generating accurate 3D models and running simulations.

### 1.1.1 Visualization Tools

As mentioned, tools for 3D visualization can be useful for debugging layout issues. There is quite a lot of tools with this function. Tools that create 3D visualization from GDSII format are described in following sub-sections.

#### a) GDS2POV

The program GDS2POV, written by Roger Light, generates a POV-Ray scene description file from GDSII data [1]. POV-Ray is a free raytracer. It can be downloaded from [2]. GDS2POV is focused on the visual side more than the technical. The program is able to generate a POV file by simply extruding layout, POV-Ray afterward permits to set visual effects.

GDS2POV is one tool from the collection GDSTO3D written by Roger Light. The collection consists of GDS2POV, GDSOGLVIEWER, and GDS2SVG. GDSOGLVIEWER is 2D GDSII viewer based on OpenGL, which is provided with the developer's commentary "*It's not very good yet!*". GDS2SVG is still in the experimental version.

More information about this project are placed at [3]. Latest information on this page are from 2008.

#### b) GDS3D

The open-source software GDS3D is developed by Ph.D. students Jasper Velner and Michiel Soer at the University of Twente [4], [5] based on GDS2POV. The software can visualize complex topology up to 7 metal layers. Creators made a video to demonstrate the functionality, a link is available at [4].

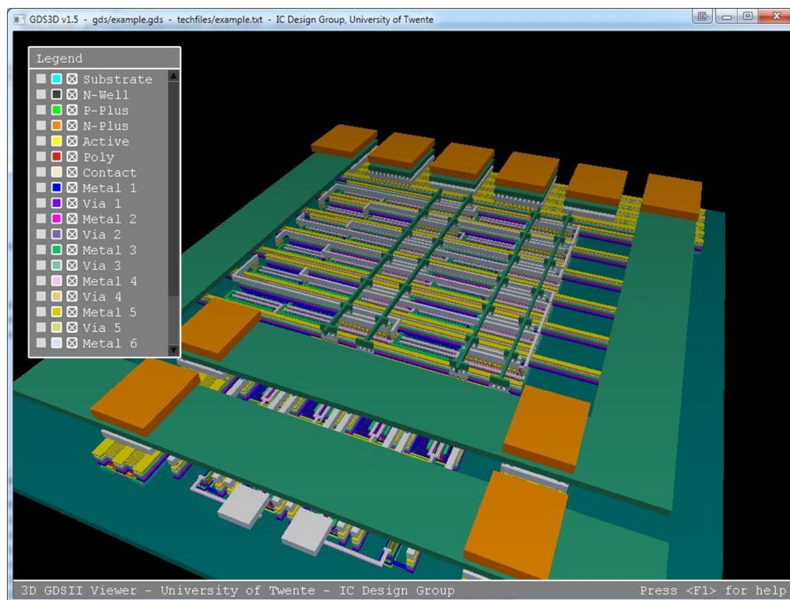



Figure 1.1: GDS3D window look [4].

In Fig. 1.1 is an example of software usage. From the figure can be seen that this software is only extruding a 2D layout and creating that way a 3D visualization. The software seems to be quick, even for larger layouts. Users can browse through the whole 3D visualization. It also has the capability of turning off and on single layers.

The software does not have the possibility to create a 3D output file, but it seems like a useful viewer that can be used as the Virtuoso plugin.

#### c) ShapeshifteR Koala

A program specialized in real-time, high-quality 3D visualization of GDSII [6]. The main advantage of this program is supposed to be speed, the possibility to quickly generate huge layouts with eye-catching visuals. Koala should be able to create layers from user-defined boolean operations and cutaways through its interactive 3D viewer.

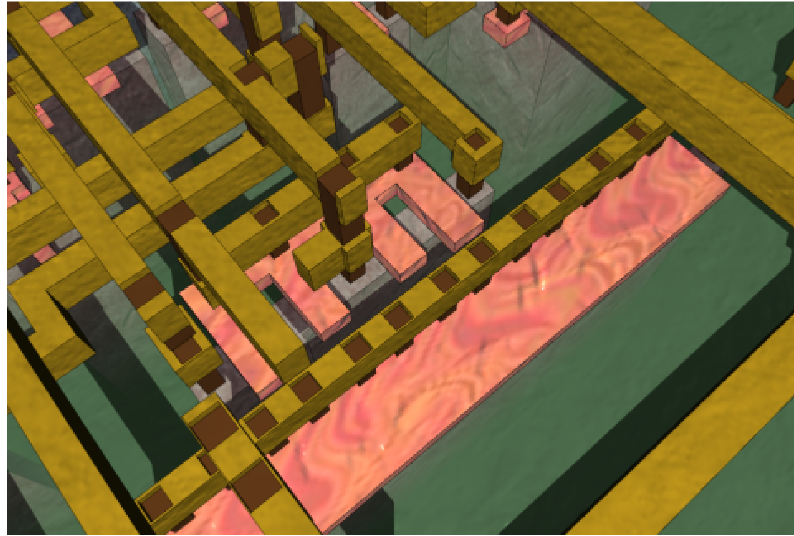


Figure 1.2: ShapeshifteR Koala visualization [6].

In Fig. 1.2 is an example of visualization in Koala with its texturing. It is also noticeable that 3D structures are created by simply extruding the original layout.

### d) Qckvu3 – Extract 3D

Qckvu is a GDSII viewer that can generate 3D output with its Extract 3D plugin [7], [8]. Qckvu has over 20 years of experience with layout viewing.

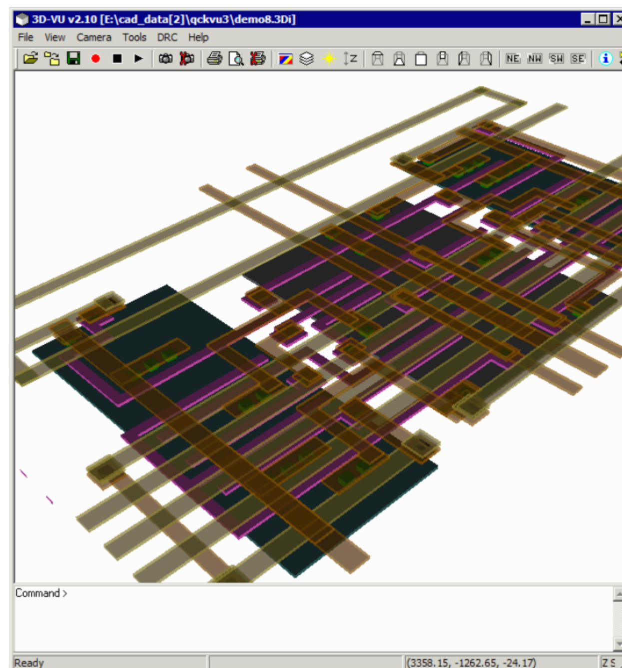



Figure 1.3: Qckvu3 – Extract 3D visualization window [8].

### 1.1.2 Simulation Tools

The visualization may be useful while debugging; even more information can be obtained from simulations. In the following sub-section, there are a few tools that can run 3D simulations from GDSII data.

#### a) Victory Process

Victory process from Silvaco, Inc. is a tool capable of 3D visualization and more importantly it allows to run simulations [9], [10]. For creation of a 3D object is used among others the Monte Carlo method to achieve most accurate visualization and physical parameters. Algorithms and equations that ensure physical accuracy may be challenging for computing power. It can lead to larger computing times or inability to process bigger GDSII layouts. Nevertheless single components of layout can be precisely converted to 3D with the possibility of precise simulations.

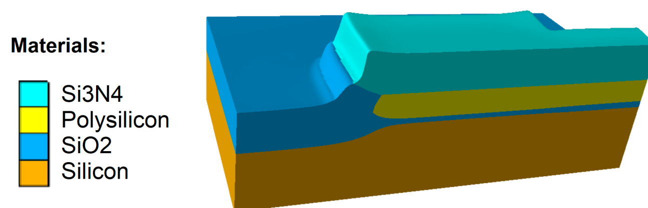



Figure 1.4: Victory process buffered oxidation [10].

The example of sophisticated modeling can be seen in Fig. 1.4. This is one of many features accompanying the process of 3D modeling. Every single one of these and all together lead to accurate model as can be seen in Fig. 1.5.

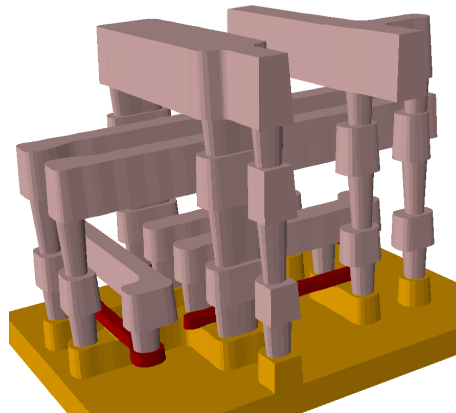


Figure 1.5: Structure from Victory process [11].

Output 3D format of Victory process is STR (Silvaco structure file format), this file format is a main format for Silvaco, Inc. 3D visualization and simulation tools.

#### b) Gds2Mesh

Cogenda is another company specialized in TCAD software development, one of their products enables 3D simulation from GDSII [12]–[14]. This tool is supposed to be similar with Victory process, main difference is in the solution. Gds2Mesh seems to have a simpler attitude using only extrusion in combination with distribution functions and a mask distortion. This rather straightforward attitude can enable generating of larger layouts and its simulations.

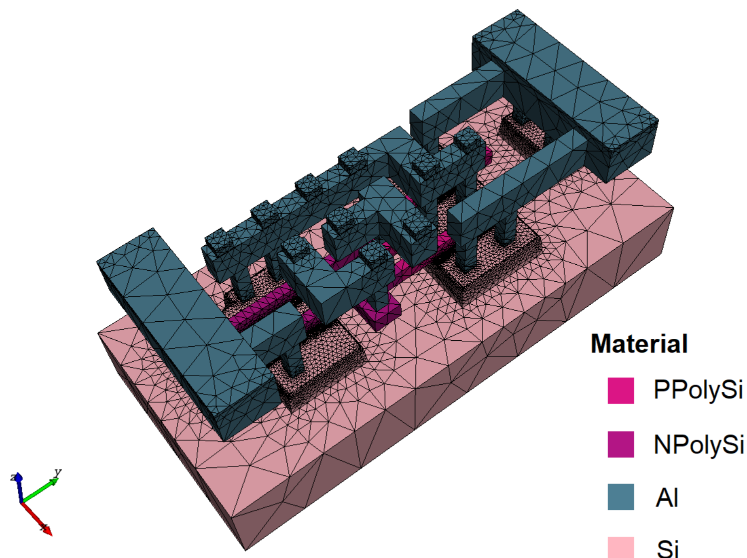



Figure 1.6: Structure from Gds2Mesh [12].

In Fig. 1.6, there is an example of a structure created with Gds2Mesh. The output file is in TIF3D format but it can also be exported to GDML format.

#### c) Sentaurus TCAD

A tool from Synopsys Inc., that does not allow to generate 3D form GDSII data, is able to visualize and run simulations [15], [16]. It is a renowned company with wide field of interests, among them the IC development.

Sentaurus Device is a useful tool for development IC, including nanoscale technologies. A NAND array structure with its mesh is in Fig. 1.7. This tool is using its own structure format TDR.

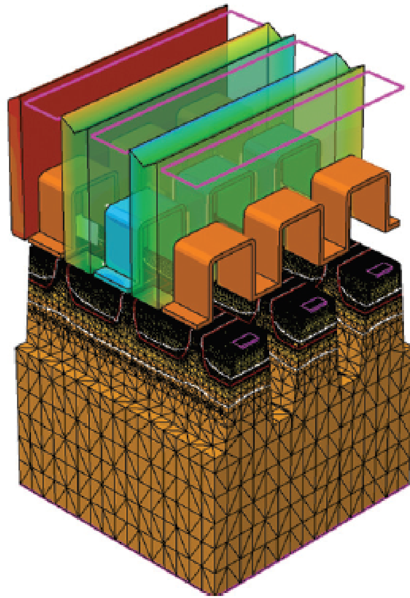


Figure 1.7: Simulated mesh and structure in Sentaurus Device [15].

#### d) Layout-Based TCAD Device Model Generation

A research aiming on visualization and simulation is described in [17]. This paper is focused on creation a 3D model of a single component and running mixed-mode simulations using tools from Global TCAD Solutions GmbH.

## 1.2 Technology of Integrated Circuits

Understanding the technology of IC is the key knowledge for the successful generation of their 3D structures. It is a dynamic and broad science discipline. Nowadays, the main engine of development is a race between companies to hold on Moore's Law [18] and gain a technological lead [19].

Moore's law is getting quite a lot of attention since there is a big question if the law will continue or if this prediction fails in the future (e.g., [20]–[23]). A graphical represented Moore's Law (Fig. 1.8) shows the actual transistor count per square millimeter by a year, from 1971 to 2020.

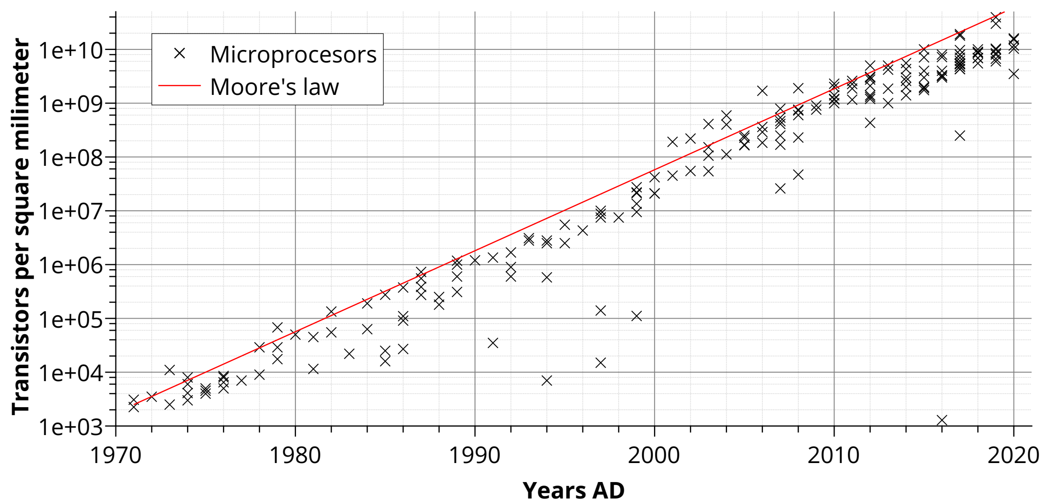



Figure 1.8: Graphical representation of Moore's Law (data from [24]).

### 1.2.1 Steps in Planar Process

The planar process of IC is a process used in semiconductor manufacturing. There are various processes, and each of them has a unique set of steps. The following section describes the key process, using information from [25] and [26].

#### a) Wafer Manufacture

The IC technology can be based on silicon, figuratively, and literally, too. The essential component of the silicon-based planar process is a silicon wafer. The silicon wafer is a thin plate of a pure monocrystalline silicon ingot. There

Table 1.1: Dependency of wafer thickness on diameter [27].

| Diameter (mm) | Thickness (µm) |
|---|---|
| 25 | —— |
| 51 | 275 |
| 76 | 375 |
| 100 | 525 |
| 125 | 625 |
| 150 | 675 |
| 200 | 725 |
| 300 | 775 |
| 450 | 925 |

are various methods for the ingot growth, resulting in various diameters of wafers. The wafer thickness changes with its diameter (Tab. 1.1).

The wafer used as a substrate in IC manufacturing is from p-type silicon, which can be enhanced with an epitaxial layer to avoid latch-up [28]. For a maximal yield (and minimal cost) are used wafers with the larger diameters. The data in [29] are showing, that 300 µm diameters were the best selling in 2019. It is to be expected, that with new technologies and factories build for their manufacture, will be implemented technologies for the largest wafer diameters currently available, to maximize the efficiency.

### b) Photolithography

Once the wafer is prepared there are many processes for finishing IC. Application of photoresist is a first step of photolitography. There are two types of photoresist, a negative, and positive resist. Main difference between the negative, and positive resist is reaction under UV light. Negative resist polymerize, while the positive resist decompose.

The resist needs to be "shaped", for further use. This is done with a system of UV source, lens (mirrors), and the photomask. With decreasing size, this process is

more challenging, and it is needed to use mechanisms for minimization of distortions and even change wavelengths to reach smaller dimensions.

The polymerized resist stays on wafer after use of a developer, that washes down unpolymerized parts. Covered areas are not influenced with the other processes, they selectively affects uncovered areas. When there is no resist needed, it is washed down with more aggressive developer.

### c) Oxidation

Most important of silicon oxides, thanks to electrical properties, and easy generation, is silicon dioxide ($SiO_2$). Silicon dioxide is in IC manufacturing process used so massively, its called just oxide. Silicon itself does not have best electrical properties among semiconductors, but in combination with oxide it became dominant semiconductor.

#### Oxide Growth

It is a high temperature process, that creates oxide on top of silicon. Heat itself can be used for oxide growth, thanks to oxygen diffusion from atmosphere. Process can be fasten when are wafers placed in a furnace, with a precursor gases injection (oxygen, steam, and additional gasses) to achieve greater thickness in shorter time with possibility of adding substances to enhance material parameters (Fig. 1.9).

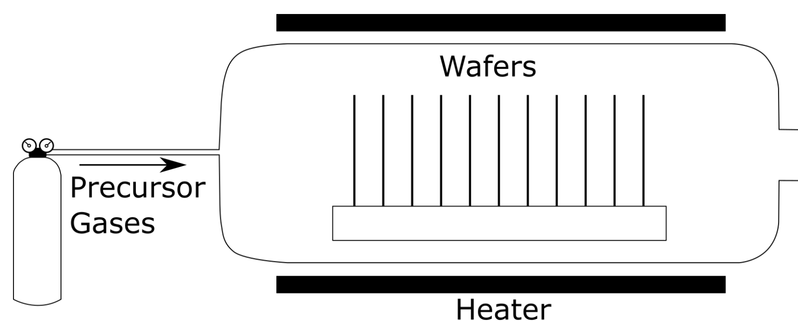



Figure 1.9: Diagram of oxidation furnace; [26] used as template.

The oxide can also be grown on top of the material. This method is often used for oxide between metalization layers. In this case, silicon and oxygen have to be added from an external source.

### Oxide Removal

This operation is called etching; it removes created oxide layer. To be able to create desired patterns, masking has to be applied. Masking is done by applying a photoresist. The unmasked area is ready to be etched.

There are two types of etching. The wet etching and the dry etching (Fig. 1.10). Both are used in planar processes.

The **wet etching** is no more used in modern processes for selective etching because of the "undercuts" it creates. These undercuts, which can be seen in Fig. 1.10, are causing inaccuracy in the planar process. Modern planar processes can not tolerate these inaccuracies, and wet etching is not used for selective etching but for dissolving whole layers of oxide.

A more precise type of etching is reactive ion etching (RIE), also called **dry etching**. It uses an ionized etching agent in the form of gas plasma. This technology is able of precise, selective etching. Unlike the wet etching, RIE can create trenches deeper than wider.

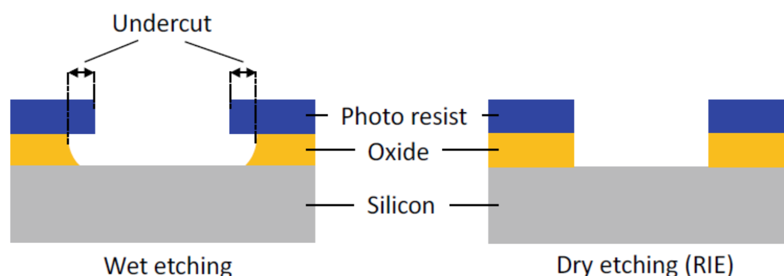



Figure 1.10: Comparison of wet (isotropic) and dry (anisotropic) etching [25].

### Local Oxidation

This type of oxidation is also known as LOCOS (local oxidation of silicon), and it is used to produce a field oxide. Silicon nitride ($Si_3N_4$) is used as a masking layer. Silicon nitride is the most commonly used nitride in the planar process, so it is called just "nitride". The pad oxide is a thin layer that enables to bond of nitride to the silicon. An unwanted structure called "bird's beak" can appear when are no precautions taken, such as etching of silicon, as shown in Fig. 1.11.

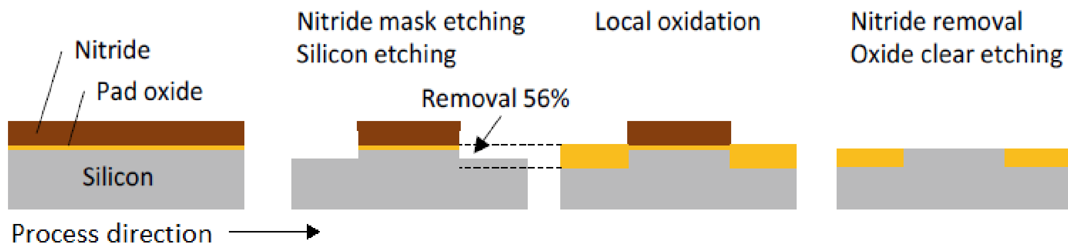



Figure 1.11: Schematic representation of the idealized LOCOS process [25].

### Oxidation Aftereffects

Oxidation is a delicate process that can create unwanted uneven surfaces in the planar process. There are quite a few inaccuracies that can be created, such as steps on the silicon surface and the "bird's beak" (Fig. 1.12). Both mentioned are caused by the isotropic characteristics of the oxidation process.

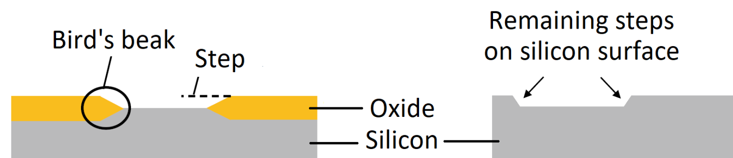



Figure 1.12: Uneven surfaces caused by oxide process [25].

## d) Doping

The make IC work, there are needed two types of silicon, they are called p-type and n-type. Their name comes from the main type of conduction.

The conduction type that uses electrons is called n-type. It is created by doping a 5-valent foreign substance such as phosphor, arsenic, and antimony. This causes the silicon to have more free electrons thanks to the doping of the foreign substance, thanks to the donation of its electron is the substance being called a donor.

P-type is uses holes (missing electrons) for the conduction. Doping a 3-valent foreign substance creates a material with more holes. Most common material for doping is boron. Created holes are used to accept electrons, the foreign substance of this type is also being called acceptor.

### Diffusion

Diffusion is a high temperature (800 °C – 1250 °C) process when substances spontaneously penetrate the substrate. When the substrate cools down, dopants are locked in the crystal lattice. Depth and profile are dependent on temperature, time, and dopant concentration; the graph is shown in Fig. 1.13. The dopant can be delivered as a layer or in the form of a gas.

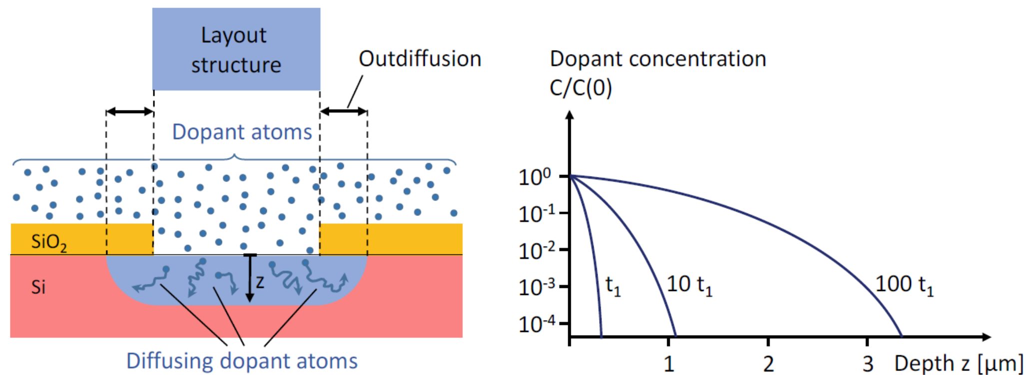



Figure 1.13: Demonstration of diffusion to silicon and graph of dopant concentration depending on depth [25].

The gas method is quite straightforward. Wafers are placed in a diffusion furnace, similar to the oxidation furnace from section 1.2.1.c. The main difference is the use of carrier gasses instead of precursor gases.

Method using a layer of dopant material is more complicated. Diffusion can be divided into two steps. The first diffusion is done with a layer of material applied on top of the substrate to create a shallow diffusion with a high concentration. Afterward, the dopant layer is removed, high temperature is applied again, and the original profile diffuses and creates a new, deeper, and less concentrated profile.

### Ion Implantation

This method is using ionized dopants that are being "shoot" in the wafer at high speed. These dopants create a thin layer with a high concentration inside the substrate; depth depends on dopants' kinetic energy (speed).

The principle is shown on the diagram in Fig. 1.14. The linear accelerator accelerates ions emitted by the ion source. A magnetic field is normal to the ion beam, and thanks to the Lorentz force, are ions forced to a circular trajectory whose

diameter differs by the mass. The electrical field of capacitor plates controls the final location of ion impact.

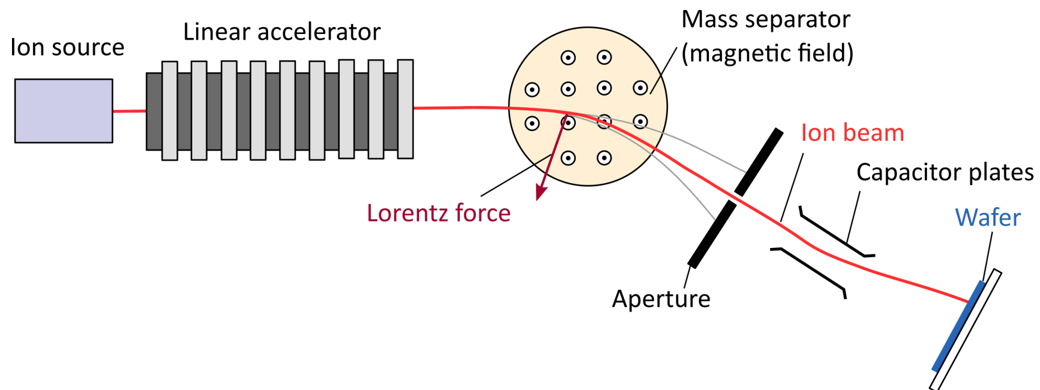



Figure 1.14: Diagram of an ion implanter; [25] and [26] used as template.

The ion implantation can be used to create more precise diffusion with a suitable profile and smaller outdiffusion. Ion implantation is the first step of this process. When is layer of doped material prepared (Fig. 1.15a), the substrate is heated (800 °C – 1000 °C) and dopant diffuses into the substrate (Fig. 1.15b).

Comparing of graphs in figures 1.13 and 1.15 shows benefits of the ion implantation. Using the ion implantation with diffusion can reach deeper and is also faster than the diffusion itself.

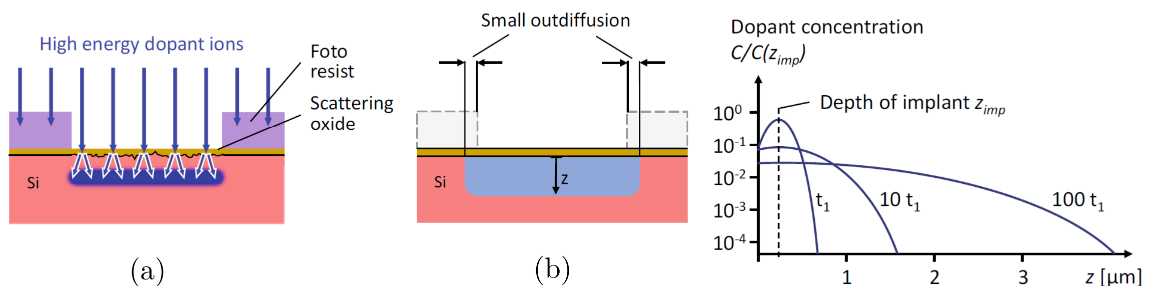


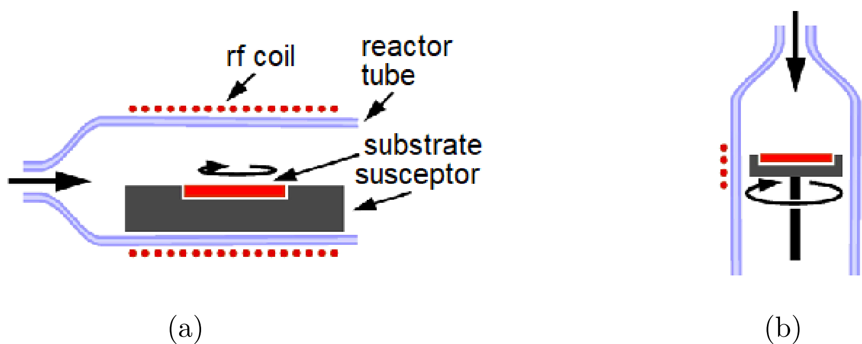

Figure 1.15: Demonstration of ion doping to silicon and graph of depth dependency on concentration and time; (a) visualization of ion implantation to substrate; (b) visualization of final diffusion area [25].

### e) Methods of Epitaxy

The process of growing a layer of material is called the epitaxy; a short overview of different methods in this section (1.2.1.e) is based on [30] and [31]. When the layer is created from the same material as the substrate, it takes over the crystallographic properties of a substrate; the process using a single substance is called homoepitaxy, but most of the time, it is just called epitaxy. The process, when multiple substances are used, is called heteroepitaxy. It creates its s own crystallographic structure

There is a lot of methods for how to grow materials on top of each other. Most of these methods require heat and the presence of a depositing material. The phase of the material is the main difference between the methods; thus, the names of these methods have been inherited from the phase used for the deposition.

The vapor-phase epitaxy can use different transport mechanisms of the gaseous species. Every transport mechanism is suitable for different materials. Method, where the material is chemically unchanged, is called physical-vapor deposition (PVD). It can be evaporated by arc discharge, electron beam, or by other means. The material can also originate from a chemical reaction; the method using this material origin is called chemical-vapor deposition (CVD). These methods have a common principle, and it is the use of a reactor similar to those shown in Fig. 1.16.




(a) (b)

Figure 1.16: Reactor types for metalorganic vapor-phase epitaxy; (a) schematic of horizontal reactor; (b) Schematic of vertical reactor [31].

There are even more epitaxial methods, liquid-phase epitaxy, solid-phase epitaxy, and many other special types based on those already mentioned. A special type of PVD deserves to be mentioned, and it is a molecular beam epitaxy. It takes longer than other epitaxy methods, but it is one of the most precise technologies.

### f) Material Removal

In the last section are mentioned methods of material generation. To create a complete integrated circuit is important to remove materials or their parts. There are two basic methods etching and chemical-mechanical polishing (CMP).

#### Etching

Methods of etching are described in section 1.2.1.c. The main difference is in a wet etching method. It uses different etchers (chemicals used for etching) that can selectively remove the required material.

#### Chemical-Mechanical Polishing

Information about this process are from [32]. The method of polishing is known for centuries. It has been used to make metal objects nice and shiny by creating its surface smooth as possible. While smoothening is top of material ground away. To make CMP work even better, chemical compounds that can be used for etching or for improving polishing are added. Diagram of CMP (Fig. 1.17) shows a principle of this method.

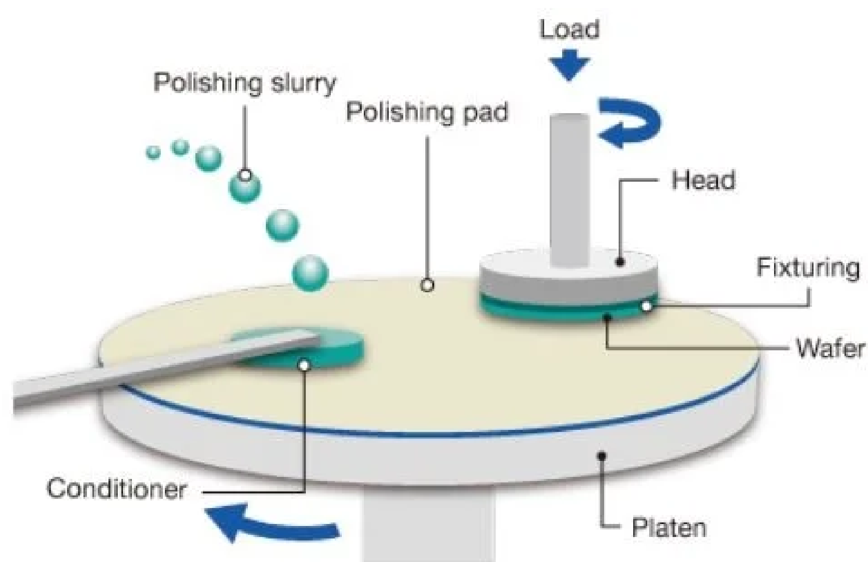



Figure 1.17: Schematic diagram of chemical-mechanical polishing [33].

CMP has its downsides when it is not done correctly. The bad setting of this method can cause not enough material ground away or grinding more material than intended. One of the unwanted outcomes can be dishing (Fig. 1.18); it appears mostly when there are larger continuous areas of some (metal) material.

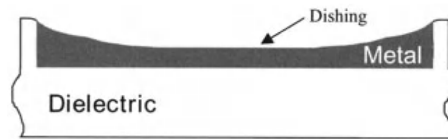



Figure 1.18: Visualization of dishing caused by CMP [32].

### g) Silicon Layers Structuring

Silicon layers epitaxy can help create quite many structures that help with the integration density, such as buried layers and trench insulations. Creating polysilicon, often used as the gate, is also an important step. This section will overview mentioned structures.

#### Buried Layer

A buried layer is created when the silicon substrate has a thin diffusion with a high concentration. When is epitaxy of silicon applied on this substrate, it creates a new silicon layer and since it is a high-temperature process, the doped area diffunds into it (Fig. 1.19).

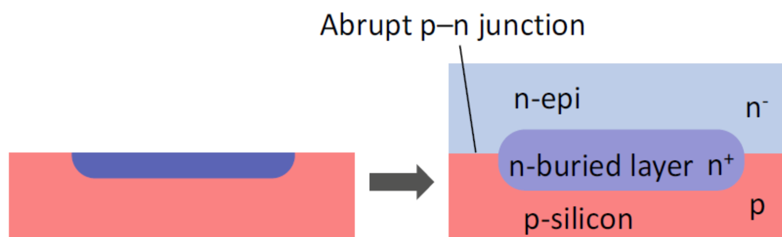



Figure 1.19: Buried layer generation [25].

#### Trench Isolation

A field oxide (FOX) created between components of IC is necessary for placing components as close as possible. It also helps to use higher voltages. Trench isolation can be divided by depth to shallow trench isolation (STI) deep trench

isolation (DTI). In this chapter are described basic steps. The actual processes can be different and far more complicated.

A bit easier trench to create is STI. It doesn't reach so deep as DTI. The process can be divided into three steps shown in Fig. 1.20. First is needed to remove part of the substrate. An applied photoresist works as a mask for etching. On the top of the etched substrate with removed photoresist is grown a layer of oxide. The oxide is meant to be only in trenches, so CMP is used to grind away surplus oxide.

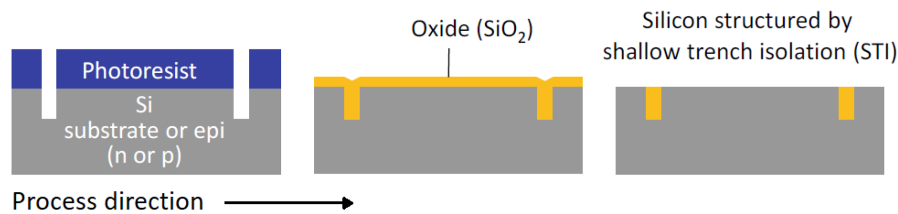



Figure 1.20: STI process visualization [25].

A DTI method is similar to STI. It has even similar steps, as shown in the visualization (Fig. 1.21). The trench in this process can be much wider. An oxide does not fill it up. Polysilicon is used to fill this space. Finishing is done with CMP to create an even surface.

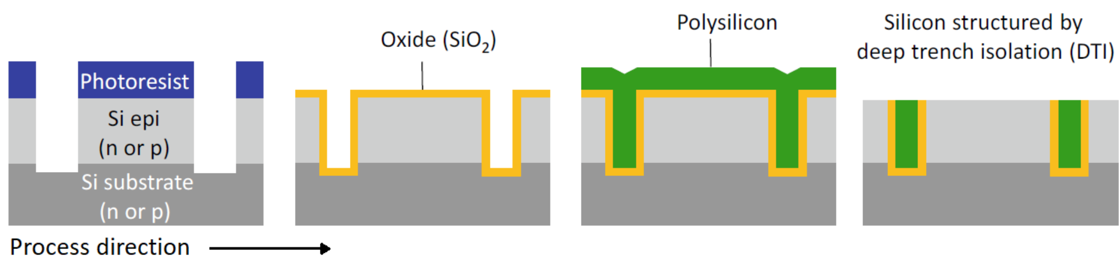



Figure 1.21: DTI process visualization [25].

### Polysilicon

The polysilicon is a special structure, a product of heteroepitaxy called polycrystalline silicon. It consists of grains and has different physical properties than silicon monocrystalline silicon. It can be used as a gate, ohmic resistor, or electrodes for capacitors.

Polysilicon is insulated from monocrystalline silicon with a layer of oxide. In the case of the gate, there is only a thin layer and is called gate oxide (GOX).

A polysilicon deposition and structuring on FOX and GOX have the same steps (Fig. 1.22). The first step is creating a layer of polysilicon. Using an etching with a photoresist mask keeps just desired structures.

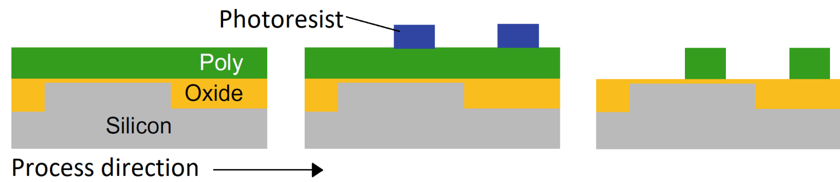



Figure 1.22: Structuring of polysilicon [25].

### h) Metallization

The planar process can be split into two parts, the front-end-of-line (FEOL) and the back-end-of-line (BEOL). FEOL is focused on the fabrication of individual devices and their patterns. Electrical connections are created in the BEOL part. This part of the process connects prepared components into the functional device.

Most of the time, a routing of components cannot be done with one layer. That is the reason why metal layers have to be stacked on each other. A simplified view of metal routing is in Fig. 1.23.

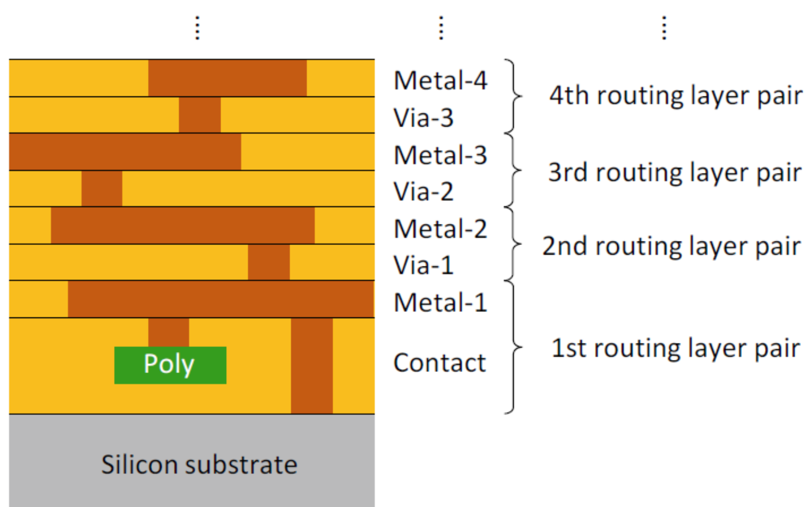



Figure 1.23: Typical view of the routing layers [25].

The used metal has to have high electrical conductivity, current-carrying capacity, mechanical stability, and good contact with the rest of the materials, e.g., aluminum and copper. To avoid Schottky contact can be used a thin layer of self-aligned silicide (salicide) on top of the silicon.

### Planarization

The state-of-the-art metallization uses planarization. One of the reasons for its use is bad connections in integrated circuits without planarization. In sub-micron structures, it can resolve in disconnected or shorted circuits. A difference between both variants of metallization is shown in Fig. 1.24.

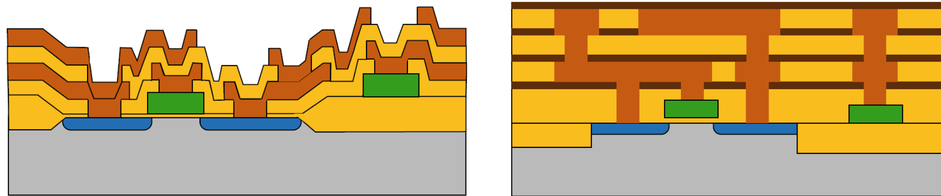


Figure 1.24: Difference between metallization without (left) and with (right) planarization [25].

The process used for the planarization of metallization is called the damascene technique. A creation of copper metallization using this technique is described in Fig. 1.25. The deposit barrier there is for the prevention of diffusion of copper. CMP is used for every layer to create even surfaces.

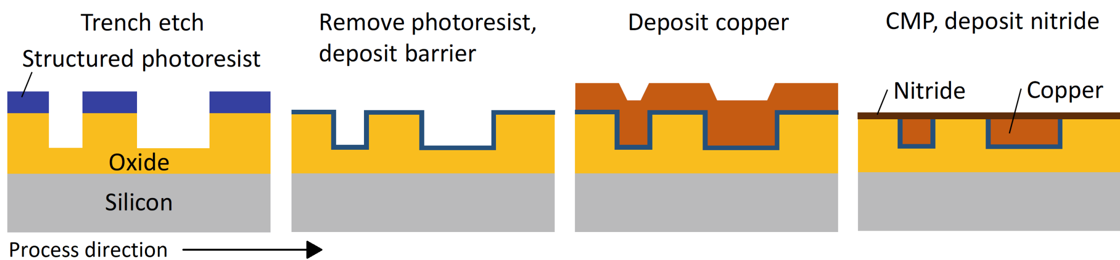



Figure 1.25: Damascene technique used to create copper interconnections [25].

### 1.2.2 CMOS Process

In section 1.2.1 are explained the steps of the planar process. These steps are the basic foundations of the CMOS process. The information in the section 1.2.2 are from [25], and [26], even more detailed information are in [34]. There is no actual standard for CMOS manufacture because it is a dynamically changing industry.

The basic structure of this process is the field-effect transistor (FET). The acronym CMOS stands for "complementary metal-oxide-semiconductor", which means this process can create both NMOS-FET and PMOS-FET. The MOS part is most interesting; this name is just a remnant of the past because the metal has been replaced with polysilicon. NMOS-FET is sometimes called NMOS transistor or just NMOS for short. Its name comes from the main carriers of the main type of conductivity, which are, in this case, electrons (n-type semiconductors). For transistor that uses holes (p-type semiconductor) is used name PMOS-FET, PMOS transistor, or just PMOS for short.

The cross-section of the basic NMOS transistor (Fig. 1.26) can serve as an illustration of its operation. An environment of the transistor is called backgate or bulk. The field-effect transistors are turned on using a control electrode, also known as the gate. This electrode is placed between two doped areas – source and drain, and it is insulated from bulk by GOX. To avoid the body effect is used contact on bulk that is connected to the source.

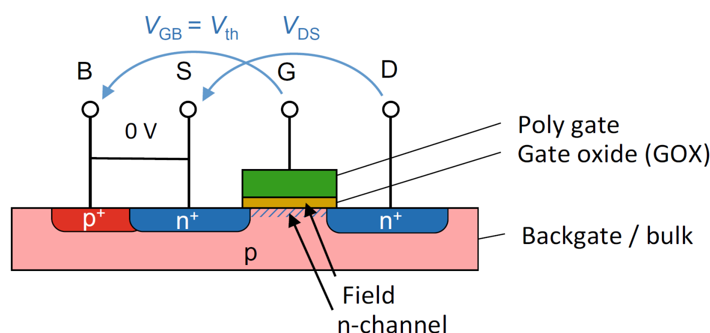



Figure 1.26: NMOS transistor cross-section [25].

Reference potential for bulk and source is selected as 0 V. When is applied voltage $V_{GB} > 0$ between gate and bulk, they start acting like a capacitor. It creates an area of material with a high concentration of minor carriers underneath the GOX. This channel works as a connection between drain and source, and when it is applied,

$V_{DS} > 0$ current can flow through this channel. This principle is "normally off," meaning it is not conducting voltage when the gate is not being powered.

There are various options for the final realization. The basic processes are the single-well process, twin-well process, and triple-well process (Fig. 1.27). These processes with few changes and additional structures are the foundation of the CMOS process.

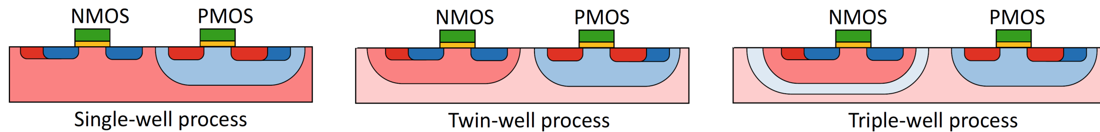



Figure 1.27: CMOS process options (p-doped substrates) [25].

To complete the process are used basically all operations from section 1.2.1. The smart use of mask layers, doping, and growing operations can lead to desired structure (Fig. 1.28). By using structures such as lightly doped drain (LDD) can be achieved better voltage capabilities.

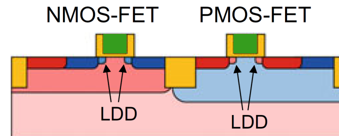



Figure 1.28: CMOS process options with LDD [25].

Using CMOS technology can be created various components, such as resistors, capacitors, diodes, and even antennas. This means this technology process is great for circuit integration.

The final chip is sealed from getting moisture inside it, using a passivation layer ($Si_3N_4$). Depending on the packaging, bond wires are connected to bond pads, or the bond pads can be soldered directly to the chip carrier.

### 1.2.3 BCD Process

This process is one of the "mixed technologies"; information for its overview are from [35], and [36]. BCD combines analog precision of Bipolar transistors, digital design of CMOS, and power and high voltage capabilities of DMOS. This combination allows

creating various products like high-voltage, high-power, and high-density BCD while being able to keep small chip area and reduced electromagnetic interference.

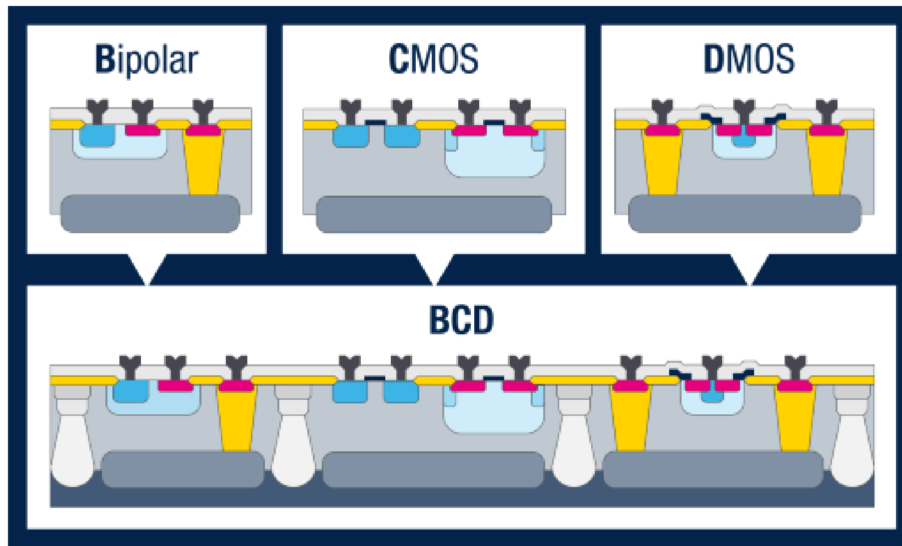



Figure 1.29: Visualization of BCD composition [36].

Products of this process have good reliability and are often used in the automotive and aerospace industry. BCD technology has the capability to manage various peripherals and its modularity. That is the reason why is BCD suitable for use in multimedia techniques.

### 1.2.4 Parasitic Phenomena

Both CMOS and BCD can create complicated structures that can be easily affected by many factors. There is quite a lot of failure mechanisms [26]. This section is focused on the part of these factors called parasitic phenomena and how to avoid them.

#### a) Latchup

CMOS circuit can trigger to low resistance and high current state, i.e., latchup, of the parasitic bipolar transistors and p-n-p-n paths inside of it [37]. The information about latchup are from [37], [38], and [39].

The latchup model has been developed to predict the latchup characteristics accurately. It also has an experimental background to support it. The circuit of this

model is shown in the CMOS structure (Fig. 1.30). The main trigger mechanisms are current across a reverse-biased junction and lateral current in the substrate.

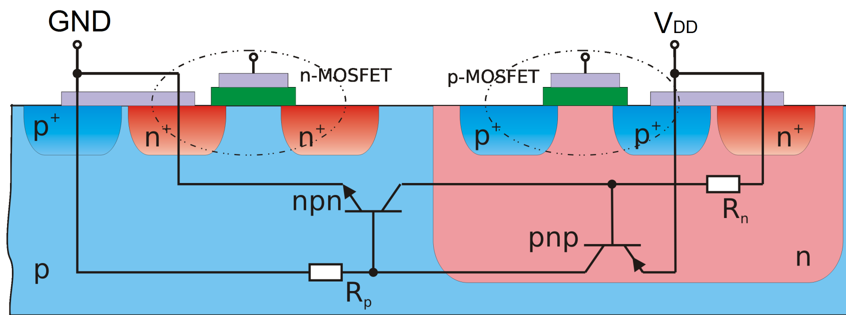



Figure 1.30: Latchup model inside of CMOS invertor [39].

There are many ways how to suppress the latchup. By increasing spacing between source/drain to the well borders or using a buried layer can reduce the gain of the product $\beta_1 \cdot \beta_2$ of parasitic bipolar transistors. Using deep trenches separates device groups. It can greatly suppress the latchup. In combination with the technology of "silicon-on-insulator" (SOI) can prevent latchup almost completely. There are many other ways, such as protective rings or multiple well structures, to suppress the latchup effectively.

### b) Parasitic Capacitances

The information for this section is from the science paper focused on parasitic capacitances [40]. The integrated circuits have capacitance practically everywhere. The main parasitic capacitances are originated in the BEOL part.

The metallization, even one single line of metal, creates capacitance. With the use of multiple lines, interconnections between them, parallel lines, and more complicated metal structures can be really challenging to understand the total capacitance of the circuit. The small circuits (only a few components) can be solved using the models for single parts of the metallization. Bigger circuits would take a long time to solve manually. Smart layout design can prevent high parasitic capacitance, and in case of high capacitance influencing circuit functionality, simulations or 3D visualization can help to reveal the issue.

#### c) Parasitic channel

The description of the parasitic channel is based on [37]. The parasitic channel is a structure created by a conductor with high potential above $p_+$-n-$p_+$ or $n_+$-p-$n_+$ structure. This may lead to induction of inverted channel by this unintentional control electrode.

The parasitic channel can be prevented by shielding vulnerable parts of the structure. Shielding can be done by metal connected to the substrate, STI, and doping ($p^+$-n-$p^+$ to $p^+$-n-$n^+$-n-$p^+$ or $n^+$-p-$n^+$ to $n^+$-p-$p^+$-p-$n^+$).

#### d) Parasitic Resistance

Parasitic resistance can be a problematic issue, although it probably is mostly an issue only for modern (FinFET) technologies. This issue is dependent on the technology size. It is primarily described in the nanometer processes, e.g., [41]–[43], and many others. The parasitic resistance can cause voltage drop on analog signals' interconnections and increase the delay of digital signals, which can desynchronize the clock.

## 1.3 File Formats

The thesis is mainly focused on the technical part such as 1) GDSII data format conversion for the 2D and 3D visualization, 2) possibility to simulation critical part on silicon by a device simulator. Therefore, there is a need to understand file formats that can be used along with the data processing. It means to be able to operate with them and be able to choose between multiple formats.

### 1.3.1 Graphic Design System

A brief overview of the Graphic Design System, also known as GDSII, is based on information from [44]. GDSII is used as the industry-standard database for IC layout. This format is used for over 35 years, thanks to its simplicity and elegant architecture.

The GDSII database is binary by default. Binary databases can achieve better compactness. Many companies have created a binary-ASCII converters. The main

motivation behind converting is to be able to work with the raw data.

A set of records represents raw data. Every record has a number in brackets, and the hierarchy of records can put together the whole structure. All of the records are nicely organized in [45].

### 1.3.2 2D Formats

There is quite a lot of graphic formats, which are in everyday use, e.g., [46]. This work will overview only in the vector graphic formats. These formats use coordinates for visualization, and they can be easily scaled (zoomed) without loss of quality [46]. Some of the commonly used formats are SVG, DXF, EPS, AI, and PDF [46], [47]. Some of these formats are proprietary, and their inner structure can be copyrighted.

Two file formats are documented enough, and no special program license is needed to work with them. One of those formats is SVG, the second one is DXF.

#### Scalable Vector Graphic

Scalable vector graphic, also known as SVG is nicely described in [48], even more detailed information can be found in [49]. SVG is a language in XML describing two-dimensional graphics. This format has an open-source viewer and can be embedded into websites.

#### Drawing Exchange Format

The DXF format has been developed by Autodesk and is used as a format for the exchange of data between various CAD software [50]. This format uses "group code" to which are associated variables [51]. Syntax and specifications are in [51].

### 1.3.3 3D Formats

According to [52] there are hundreds of 3D formats. In this section will be mentioned only a few of popular formats. These formats can be used for visualization, animation, and for 3D printing.

#### Wavefront OBJ

One of the most popular 3D formats, developed by Wavefront Technologies [52]. There are two variants of this format, ASCII and binary. The ASCII format is open-source. This format is mostly used for visualization, but 3D printing can use it too.

#### Blender

CAD manufacturers often use proprietary file formats [52]. One of those proprietary formats is the BLEND format. This format belongs to Blender, which enables scripting graphics using python [53].

#### Stereolitography

The most popular stereolithography format is STL [52]. Its main domain is 3D printing. The use of this format in rapid prototyping is almost a matter of course. STL is a simple format that uses only surface geometry, the rest of the information it ignores.

#### Universal 3D

A bit of special format in terms of documents is U3D [54]. This format is not listed among the most popular in [52]. As mentioned, U3D is special in term of documents and that is because U3D can be embedded in PDF (Fig. 1.31). The 3D view has been tested in Adobe Acrobat (ver. 2021.001.21050).

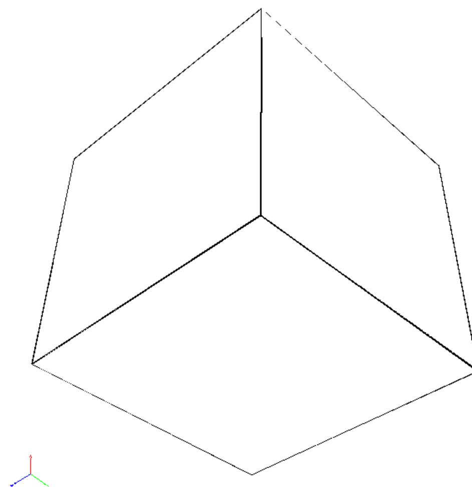


Figure 1.31: Cube embedded using U3D format (click-on 3D activation).

# 2. Methods

This chapter is focused on methods and ideas used in the process of developing the diploma thesis program. The name of this program is **G-Visit**, it is an acronym for "GDSII visualization and simulation tool". The idea that G-Visit is a gate to the world of GDSII visualization and simulations is behind the draft of its icon.

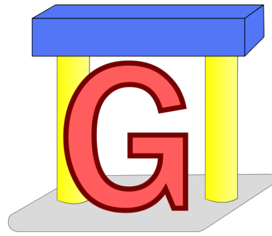



Figure 2.1: Draft of G-Visit icon.

The workflow of G-Visit (Fig. 2.2) shows the basic principle of the whole program functionality and formats it uses. This diagram can be parted into three parts by its main focus. These parts are a 2D part, a 3D part, and a TCAD part marked by different colors. The process of development has been done gradually, part by part from start to end. In the beginning, there was a similar but different diagram. Changes including used formats and the structure itself have been incorporated in the process of development.

The module hierarchy shows all modules used by G-Visit (Fig. 2.3). This diagram helps to understand the inner structure better. G-Visit is developed the modular way, and in case of need, modules can be edited or even replaced in the future. The reason for splitting operations into more modules is to make them less complicated and easily editable. The same colors mark the 2D part, the 3D part, and the TCAD part as in Fig. 2.2. Also, the imported modules and operationalize parts are marked by their own colors. The imported modules support the functionality of newly created modules. The operationalization part is focused on algorithmization and making things work together.

Further information about modules, their structures, and used file formats is reviewed in upcoming sections. The structure of sections is divided into the same parts as the workflow and module hierarchy. This should make it easier to understand and interconnect information.

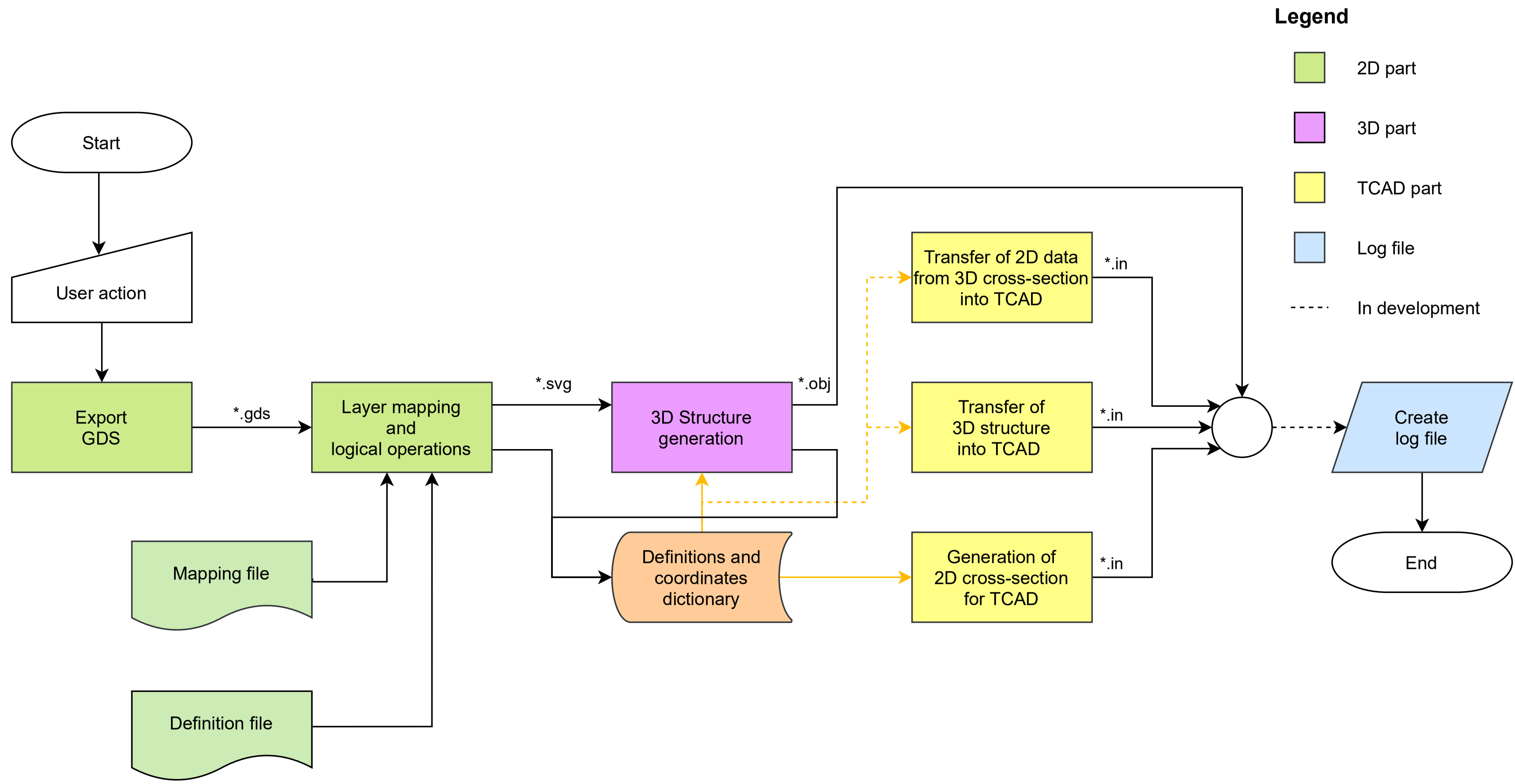



Figure 2.2: Flowchart of G-Visit workflow.

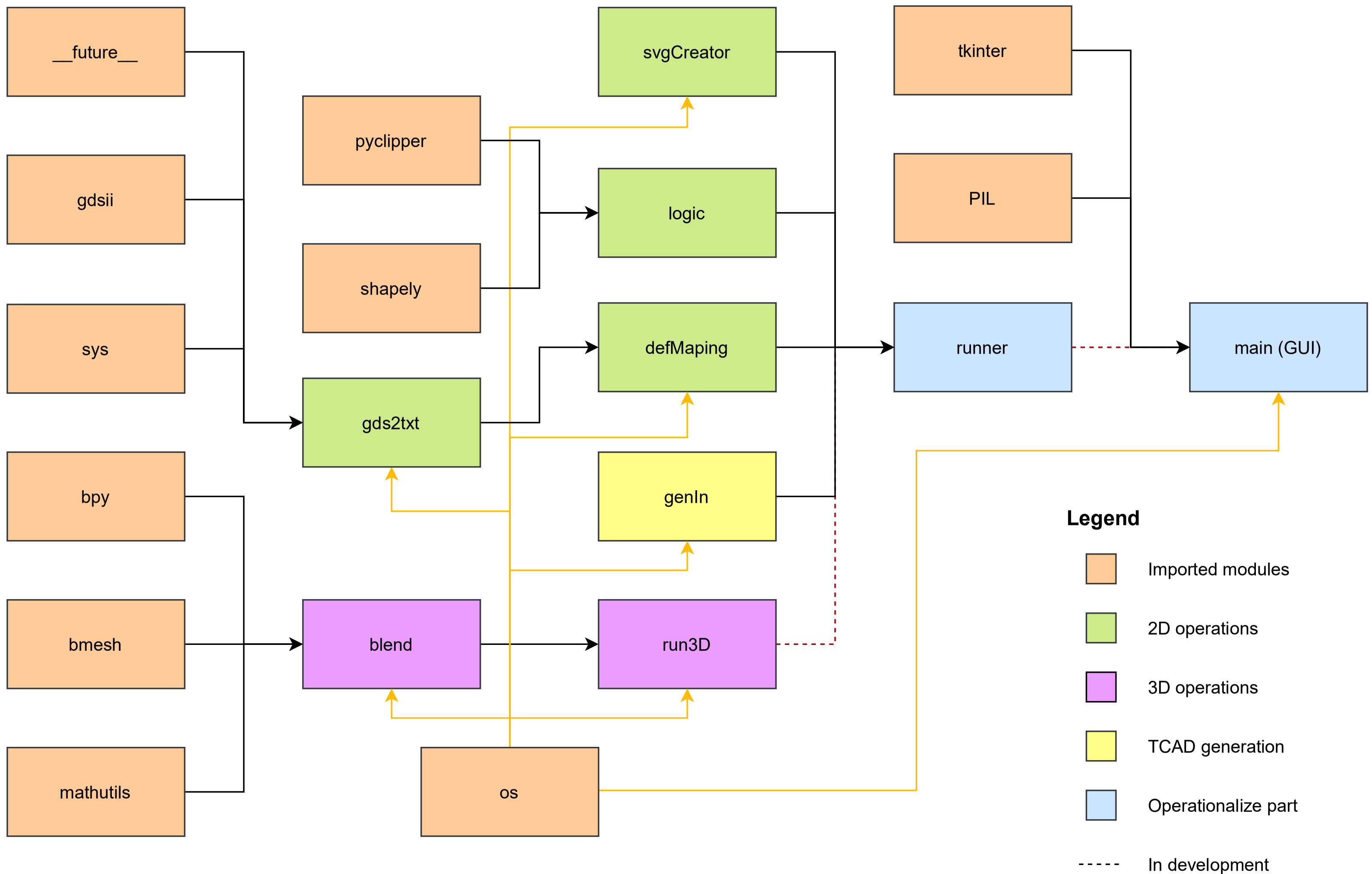



Figure 2.3: Diagram of Python modules hierarchy.

## 2.1 Definition File

The module **defMaping** loads data from definition file. Although this module belongs in 2D part, it loads all definitions to the dictionary, which is available through the whole process. It means that correct definition is important for 3D and TCAD part, as well.

There is few commands, that are supposed to make loading of definitions easier. For commenting is used the hash symbol "#". The whole file is splitted into parts with use of \part command.

The first part is for the layer definitions (List. 2.1). There are defined layers using the *.layerprops file or manual input. The *.layerprops file has a great advantage because it contains color information. The manual input consists of LayerName, LayerPurpose, GDSName, and GDSPurpose. The color file has to be added with the manual definition, if not, as the default is used a color file from Cadence Virtuoso's library (GPDK090).

```
1 \part                                      # Part for layer definitions command
2 # Automatic definition from file
3 \defined_in(INVENTOR_1s.gds.layerprops)
4 # Manual definition
5 LayerName: active                          # Layer name and begin of layer definition (mandatory)
6 LayerPurpose: drawing                      # Layer purpose
7 GDSName: 1                                 # GDS layer number
8 GDSPurpose: 0                              # GDS layer purpose number
```

Listing 2.1: Part for layer definitions.

The second part defines logic operations (List. 2.2). These operations are stored in the dictionary, and afterward, they are used to alter existing layers or create new ones. Its definition consists of LayerName, LogicOperation, and RGB.

```
1 \part                                      # Part for logical operations command
2
3 LayerName: salicide_down                   # Layer name (mandatory)
4 LogicOperation: geomOr(nplus pplus)        # Logic layers operation
5                                            # - nested function not implemented
6 RGB: [0,255,255]                           # Color setting by RGB or keep_old
```

Listing 2.2: Part for definition of logic operations.

The third part is used for definition of 3D operations (List. 2.3). In this part can be used the \sti command, which allows the generation of STI, the parameter of this command is the masking layer. There are also defined operations, the possibility of using bevel, visibility, and height of operation. The special parameter is "trapezoid," which applies the bevel operation with only one face.

```
\part                                            # Part for 3D operations command
\sti active                                      # STI layer generation + masking layer
# First are sinking operations --> from smaller sinks to bigger sinks
LayerName: pplus                                 # Layer name and begin of layer definition (mandatory)
Operation: sink                                  # operations - sink, grow, growPlane (default grow)
Bevel: 1                                         # 0 [False] or 1 [True] (default 0)
Trapezoid: 0                                     # Special Bevel - 0 [False] or 1 [True] (default 0)
Visibility: 1                                    # 0 [False] or 1 [True] (default 1)
Height: 1                                        # Height in micrometers (if growPlane used - planarization
                                                 #  above actual maximal height) - last parameter!!!
```

Listing 2.3: Part for 3D definitions.

The last part defines materials and operations needed for TCAD *.in file creation (List. 2.4). Materials such as silicon need to have defined their doping. It is an optional parameter, which consists of doping type, concentration, and doping profile. In the TCAD part, there are not defined heights because they are already defined in 3D definitions.

```
\part                                            #Part for TCAD definitions
# automatically generates oxide and substrate
# materials --> atlas user's manual p. 811

\substrate

LayerName: pwell                                 # Layer name and begin of layer definition (mandatory)
Material: silicon                                # TCAD name for material
DopingType: p                                    # Doping type parameter
Concentration: 1e18                              # Concetration in cm^{-3} format 1e##
DopingProfile: uniform                           # Uniform or gaussian (default uniform)
Operation: sink                                  # Operation sink or grow (on the previous layer)
PreviousLayer: substrate                         # Previous layer - last parameter!
```

Listing 2.4: Part for TCAD definitions.

The defMaping module use for its function gds2txt (overviewed in section 2.2.1) and the os module. Many others modules use the **os** module. It is used for getting paths in the operating system.

## 2.2 2D Part

This section focuses on operation with 2D data, i.e., reading input data, editing them with the help of definition files, and exporting them. The data exported to SVG have great visualization capability.

### 2.2.1 Definitions and GDSII Mapping

The **defMaping** module does not load just definitions stored in the definition file, but it loads coordinates from GDSII and has many functions that help to orient in the stored data.

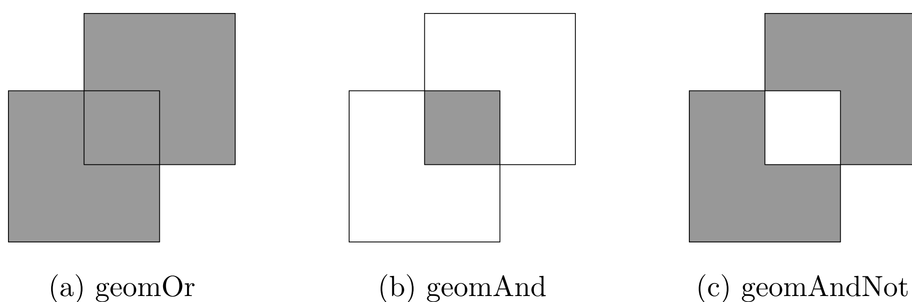
The GDSII format is commonly used for layout design, as mentioned in section 1.3.1. This is also the default format of Cadence Virtuoso Layout Suite, available in one of the laboratories in our faculty. The first thing needed to do is to translate data from its binary format. This is done by module **gds2txt**, based on [56]. Translated data are processed with information from [45] and stored into dictionary.

### 2.2.2 2D Logic Operations

The operations from definitions are scripted in the **logic** module. There are also operations used internally that helps with GDSII coordinates processing, and other processing steps. These functions can for example change format of coordinates so it can be used by imported modules, such as pyclipper (boolean operations) and shapely (graphical operations).

**Boolean**

Pyclipper (ver. 1.2.1) is used for boolean operations (Fig. 2.4). It uses one object as a clip (mask for the operation) and second one as a subject. When are used complicated polygons, that intersects on multiple places, it can create multiple outputs. G-Visit uses boolean operations with these function names: geomOr (Fig. 2.4a), geomAnd (Fig. 2.4b), and geomAndNot (Fig. 2.4c).



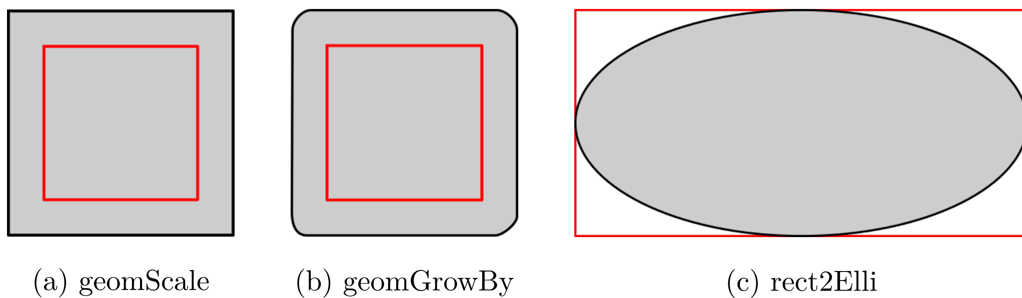
(a) geomOr (b) geomAnd (c) geomAndNot

Figure 2.4: Representation of 2D boolean operations with names of their functions.

The definition of boolean operations takes only one operation argument, and when it is needed to use more, it has to be split into more single operations. Implementing of nested function is not priority.

#### Geometric

Shapely (ver. 1.7.1) is a graphic tool, used for some graphic operations (Fig. 2.5). The operations called geomScale and geomGrowBy seem familiar on the first look, but geomScale (Fig. 2.5a) uses multiplication for change of the original (red) rectangle, and geomGrowBy (Fig. 2.5b) uses addition. There can also be seen a secondary property of geomGrowBy; it rounds the corners. Another important function is rect2Elli (Fig. 2.5c), which changes a rectangle to ellipse, in the case of the square to circle.



(a) geomScale (b) geomGrowBy (c) rect2Elli

Figure 2.5: Representation of 2D operations with names of their functions.

### 2.2.3 Generation of SVG

The SVG format has been chosen for this work over DXF for its better handling with holes inside of objects, which have been tested. SVG also has an advantage in graphic visualization possibilities.

The module that create the *.svg file is **svgCreator**. There are many modules available, but they didn't suit the application. They were unnecessarily complicated, or missing crucial functions, like grouping by layers.

## 2.3 3D Structure

For 3D structures, there are many tools for Python that have been taken into consideration, e.g., bpy, trimesh, pygame, panda3d, openpyscad, etc... Every tool has its pros and cons, and finally, the Blender (bpy) was chosen for its versatility, ability to work with various formats, extensive documentation, and active community.

### 2.3.1 Blender Connector

Blender uses two main modules, bpy (ver. 2.82.1) and bmesh, that is part of bpy's dependencies. The module **blend** creates usable functions from sets of bpy and bmesh commands. It also uses mathutils modules for vector operations.

### 2.3.2 Generator of 3D View

Nowadays, the **run3D** is more or less the main file for 3D operations; it generates the whole 3D view from processed 2D data. Bugs in boolean operations cause that the whole process can not be fully automated yet. Once the bugs will be resolved in future versions of bpy, automation will occur in the runner module using the run3D functions.

#### Extrude

The function that adds the third dimension is in Blender called extrude. Using this function, there two functions are created, grow, and sink (Fig. 2.7). Although the outcome of these functions looks different, they are quite similar to each other. A simplified flowchart (Fig. 2.6) shows a principle of the grow function.

This flowchart works with profile.blend, and layer.svg. The profile changes with every layer's growth; the output profile is the original profile that is changed by the union function with the growing layer. To avoid issues with boolean operations, there are added overlapses that ensure the correct location of objects.

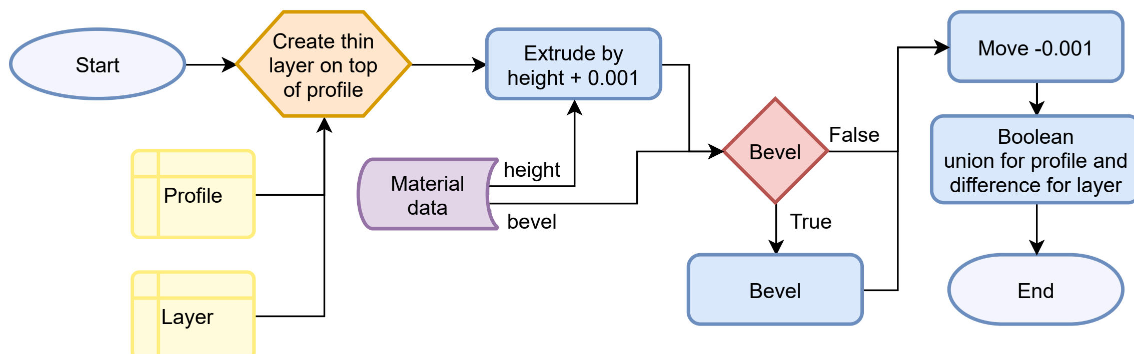
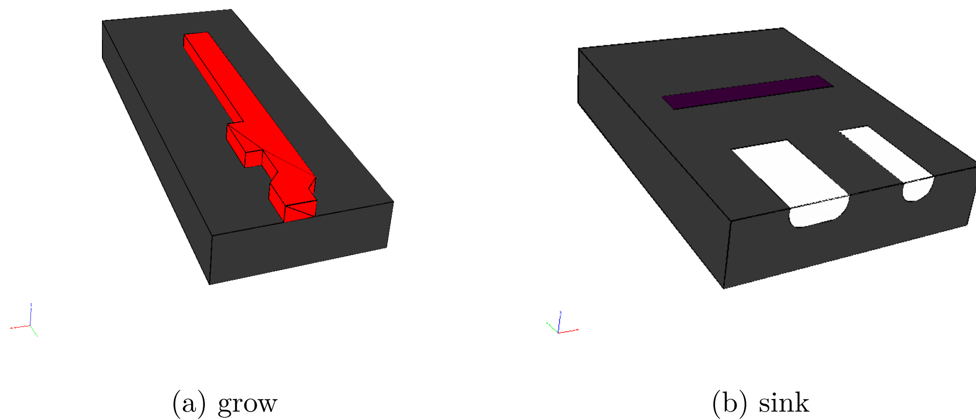



Figure 2.6: Simplified flowchart of "grow" function.



(a) grow (b) sink

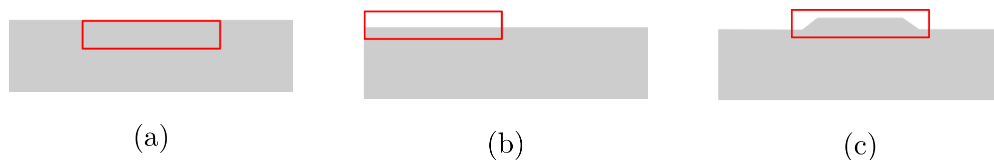
Figure 2.7: 3D operations using extrude method with name of functions (click-on 3D activation).

### Bevel

One of the parameters of grow and sink function is a bevel. This parameter is for rounding of desired edges. Bevel with the grow function rounds edges on top of the object and sink on the bottom (Fig. 2.7b). The important parameter for the bevel is the number of edges it splits.

### Boolean

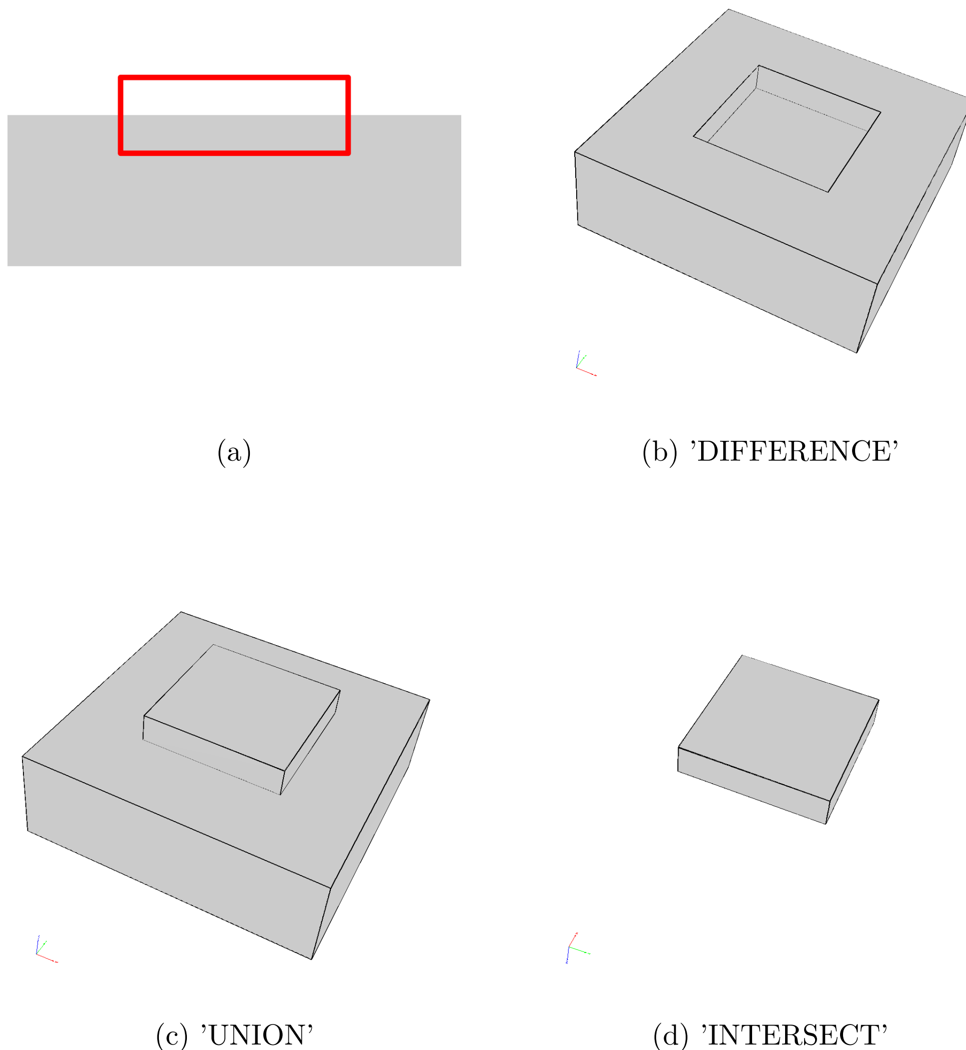
The most problematic part of the whole G-Visit development is boolean operations. There is quite a lot of bugs in bpy version 2.82.1 (Fig. 2.8). Since boolean operations are one of the most important 3D operations, it has been quite challenging to avoid these issues.



(a) (b) (c)

Figure 2.8: Cross-section of problematic positions for boolean operations in blender library - bpy (2.82.1); (a) faces on the same level; (b) touching faces; (c) complicated profile of face.

The 3D boolean operations (Fig. 2.9) are quite similar to 2D boolean operations (Fig. 2.4). The operation can be done by function bool3D with operators 'DIFFERENCE'(Fig. 2.9b), 'UNION'(Fig. 2.9c), and 'INTERSECT'(Fig. 2.9d). A cross-section view for this examples is in Fig. 2.9a.



(a) (b) 'DIFFERENCE'

(c) 'UNION' (d) 'INTERSECT'

Figure 2.9: Boolean operations overview (click-on 3D activation); (a) cross-section of substrate and operator object; (b-d)function bool3D operation with different inputs.

### Special operations

Special operations, except "planar," are created to avoid bugs of boolean operations. They are used to create layers where cannot be avoided one of the mentioned issues (Fig. 2.8).

## 2.4 TCAD Export

The final part of this program is creating input for TCAD. There is available Silvaco TCAD in a laboratory at our university, so its format is used for G-Visit. The data are generated into *.in file, which can run in DeckBuild and generate the mesh.

#### Input from 2D

This process is done by splitting polyline into lines. These lines are one by one used with and operation on other layers. In this process, coordinates converse from the X-Y coordinate system to the X coordinate system. The Y data are generated from definitions.

#### Input from 3D

This part is not yet implemented. The 2D data should be generated from the cross-section, and for beveled objects is a planned approximation method. There also should be a 3D input for TCAD.

### 2.4.1 Generation of TCAD Input

The principle of TCAD *.in file generation by the **genIn** module is shown in Fig. 2.10. It only generates regions, mesh, and dopings. The electrodes have to be added manually. The user also has to prepare the simulations.

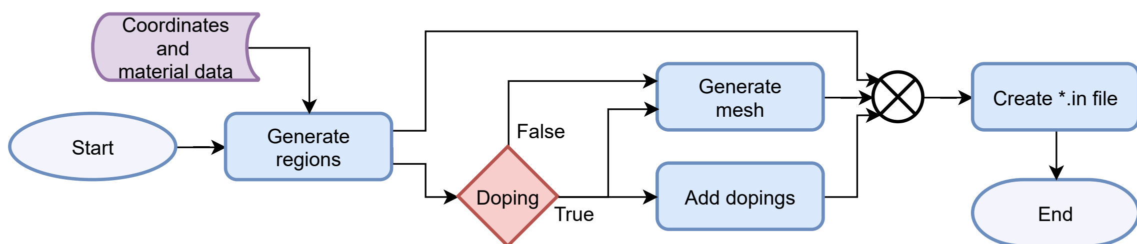
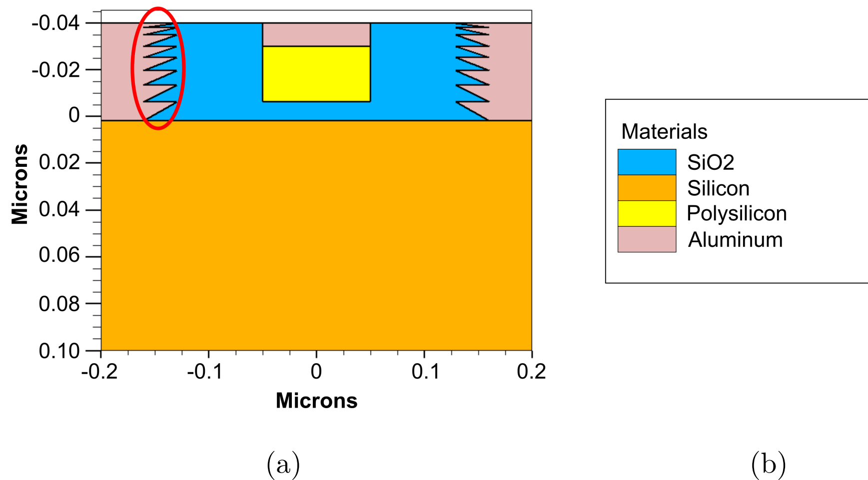



Figure 2.10: Simplified flowchart of genIn module principle.

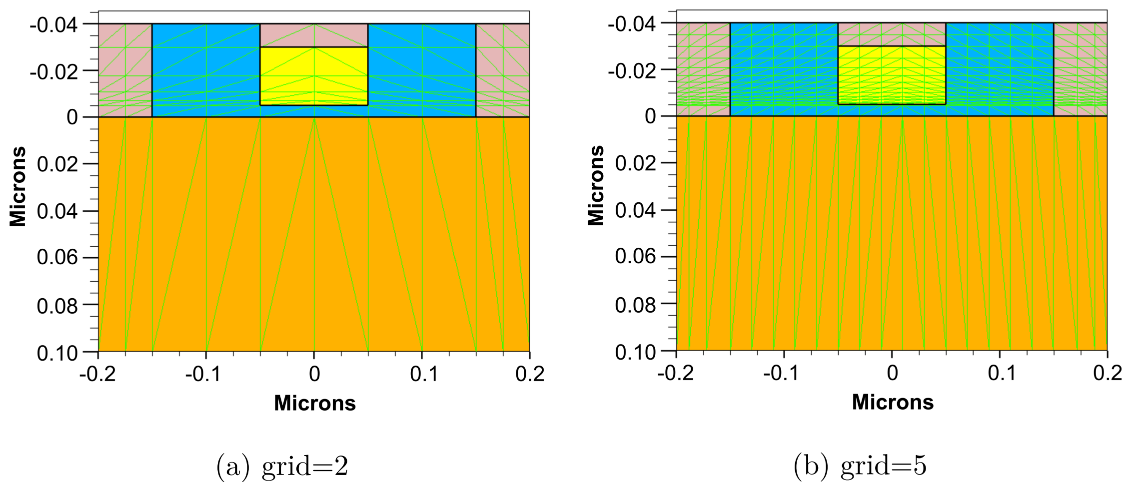
It is necessary to generate regions as first, to be able to match mesh with generated regions. If not done properly, it can resolve in badly generated regions (Fig. 2.11a). This inaccuracy leads to a dysfunctional model.




(a) (b)

Figure 2.11: Bad meshing.

Meshing is very important for simulations: the more precise and detailed mesh, the more precise simulation. In Fig. 2.12 there is the comparison of mesh with two different grid parameters. This figure also shows substrate and oxide generation; it is by default generated with minimal mesh precision (noticeable on the substrate). The precision is achieved by adding more layers to the model, which can be seen on the oxide part.




(a) grid=2 (b) grid=5

Figure 2.12: Visualization of mesh precision dependence on the grid parameter.

# 3. Results

This chapter contains results of created visualization files, TCAD input with its simulation, and an overview of time complexity. For testing of G-Visit has been used the GDSII file with a layout of an inverter (Fig. 3.1).

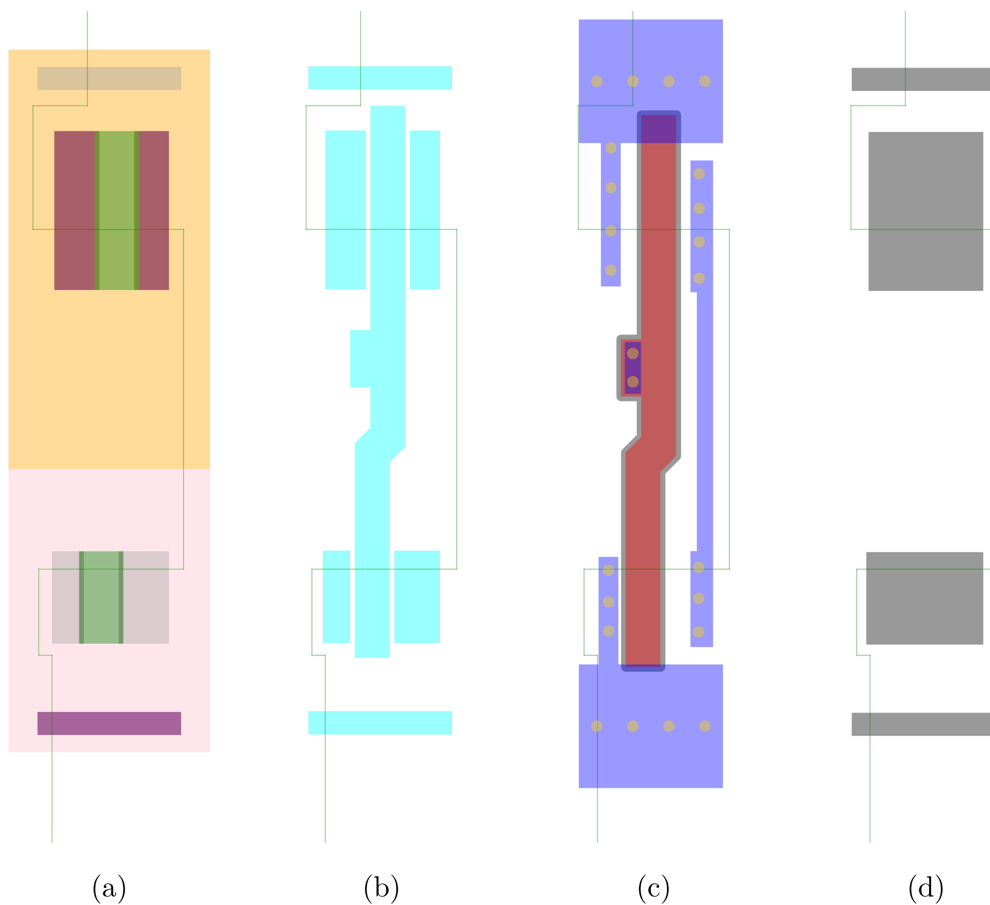
Figure 3.1: Screenshot of input GDS file layout from KLayout (legend added).

## 3.1 Visualization

This section shows the visualization of the input file (Fig. 3.1) using G-Visit. It shows possibilities for 2D layout and 3D structure views.

### 3.1.1 Layout View

The original layout makes layers really indistinct (Fig. 3.1). Its format is not great for visualization purposes. G-Visit edits and generates new layers to really show the layout of the final product (Fig. 3.2) and legend (Fig. 3.3). The SVG format makes it really easy to decompose and arrange layers into the synoptical complexes manually. GDSII can turn on and off the layers; creating a similar output is possible but difficult, and it does not contain all layers of IC.



(a) (b) (c) (d)

Figure 3.2: Layout generated by G-Visit; (a-c) decomposed layout; (d) STI mask.

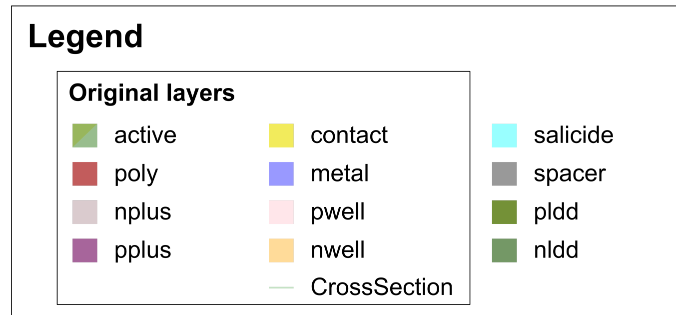



Figure 3.3: Legend for figures (3.2a-3.2c).

### 3.1.2 Generated 3D Structure

The 3D view (Fig. 3.4) has really clean look. Although this view has wrong colors caused by export to OBJ format the legend in Fig. 3.3 does mostly correspond to the 3D view. The spacer has yellow color but it should be grey.

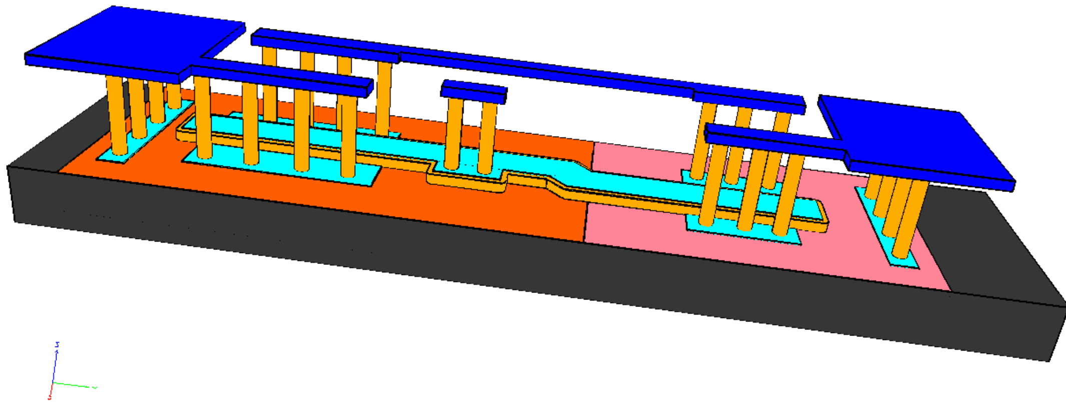


Figure 3.4: Final view of generated IC (click-on 3D activation).

The STI layer is visualized separately (Fig. 3.5). The STI layer (black) is sunk in the substrate. Visualization of STI together with the rest of the layers breaks them. Reasons and solutions will be discussed in chapter 5.

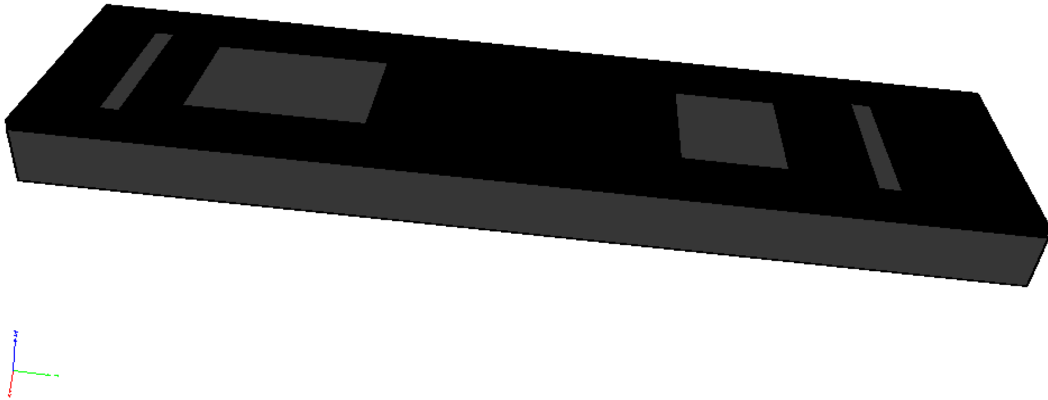


Figure 3.5: Generated STI layer (click-on 3D activation).

Blender file is the native output of G-Visit. This format keeps the correct colors of objects, and the IC can be there edited for visualization. Example is shown in Fig. 3.6. It also shows the possibility of layer decomposes to show better the inner structure. There can also be done cross-section or other graphical editing.

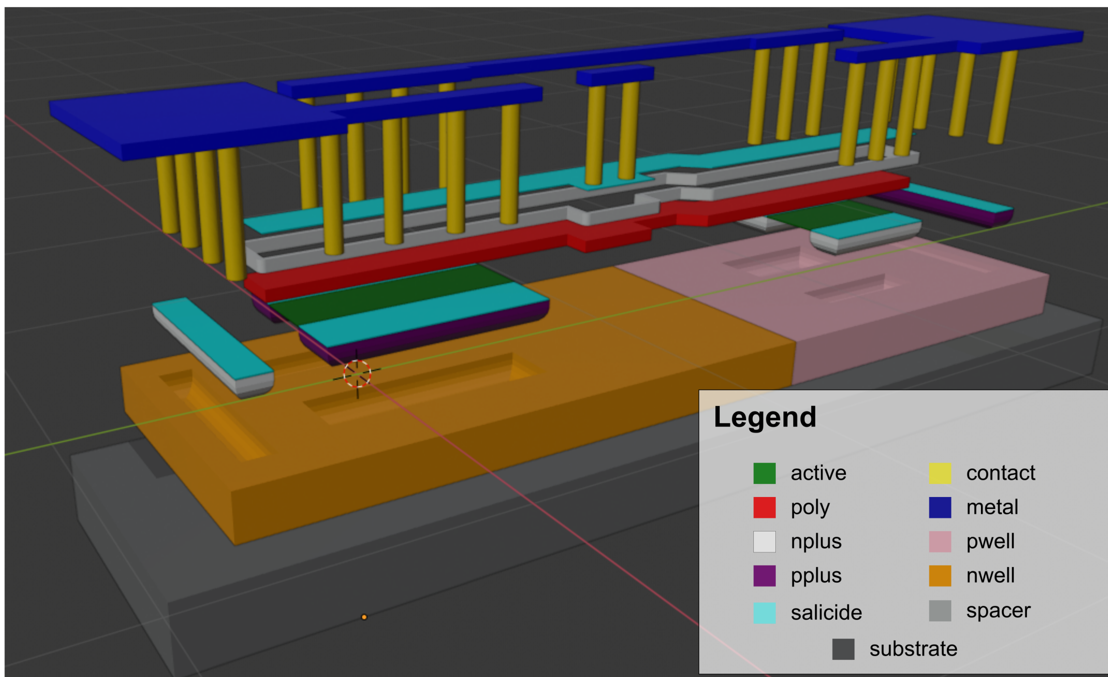



Figure 3.6: Screenshot of decomposed inverter in Blender (legend added).

## 3.2 TCAD Data

Generated file from G-Visit is executable in Silvaco TCAD Deckbuild. It has set regions, their doping, and mesh. Electrodes and simulations have to be set manually. Layers as the gate oxide and salicide are about 10× thicker for visualization purposes.

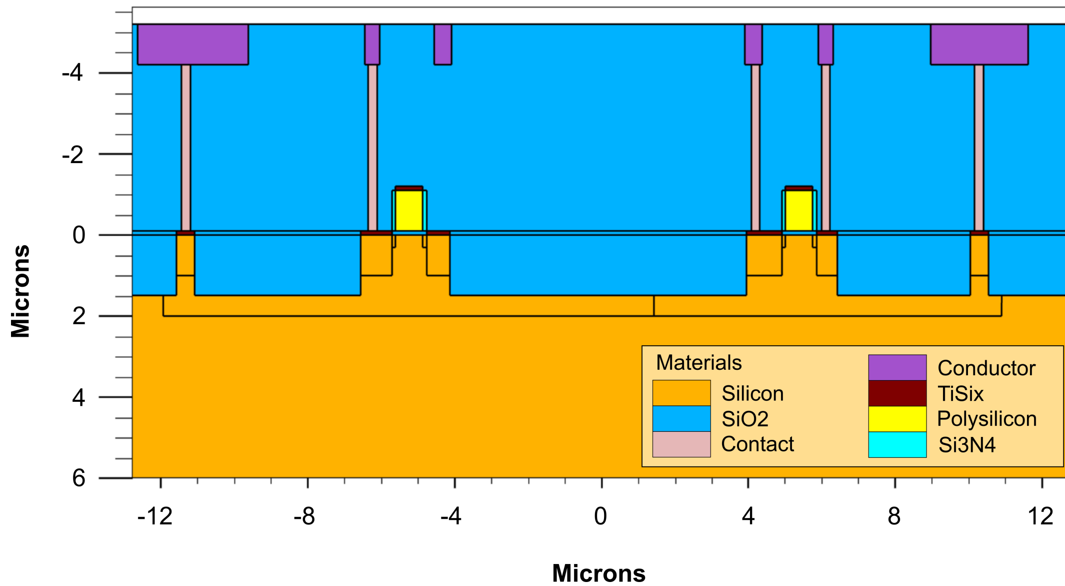



Figure 3.7: TCAD output structure.

For simulations has been used the NMOS transistor from the generated inverter with gate oxide thickness 8 nm. From the transfer characteristic can be estimated threshold value about 1 V.

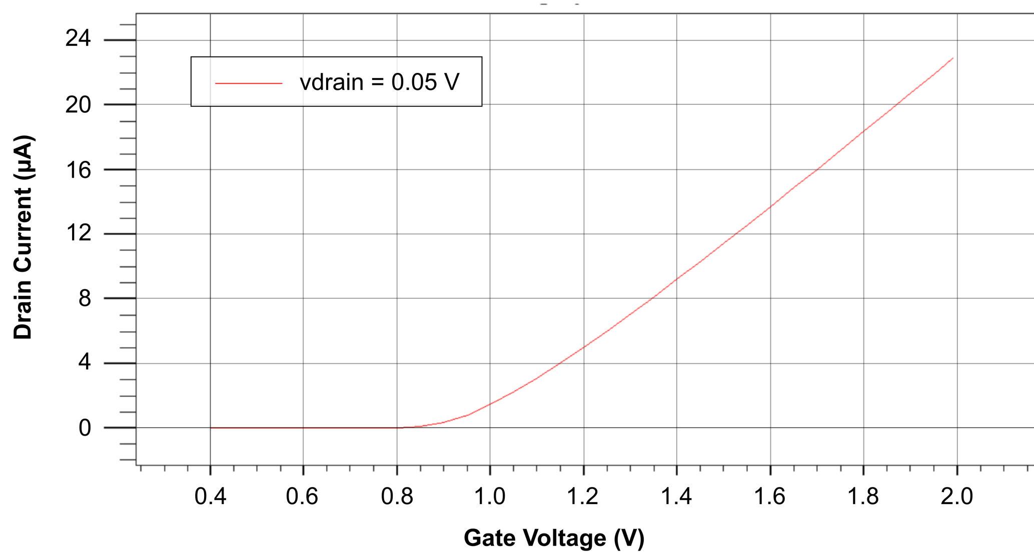



Figure 3.8: Transfer characteristics of the transistor.

The effective channel length of this transistor is 1 µm, the LDD regions length is 0.1 µm and width of the transistor is 2 µm. Doping of p-type substrate is $10^{16}$ cm$^{-3}$. The n$^{+}$ region doping is $10^{23}$ cm$^{-3}$ and for LDD it is $10^{20}$ cm$^{-3}$.

The standard simulation of transistors is an output I-V characteristic. This simulation has been done for the same transistor. The I-V characteristic has been measured for 4 gate voltages, 1.0 V, 1.1 V, 1.2 V, and 1.3 V. The characteristic has been manually split into linear and saturation regions.

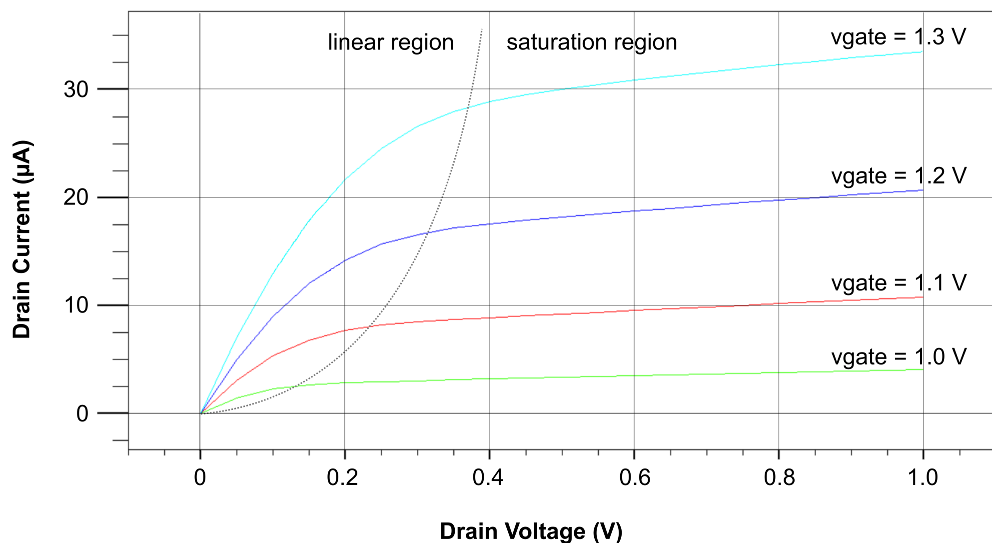



Figure 3.9: Output I-V characteristics of NMOS.

## 3.3 Time Complexity

Time complexity is important attribute. It can show advantage that can G-Visit bring to its users.

### 3.3.1 Comparison with Manual Work

Manual work is not a normalized parameter, and it can vary with users' proficiency. The measured data are just for demonstrative purposes.

Manual creation of Silvaco TCAD *.in file has not been measured. Many coordinates users have to measure in GDS manually and translate them from X-Y coordinates to X coordinates and set the Y parameter. Another complication would be adding layers that GDS misses and are applied in the manufacturing process.

A manual visualization in Blender has been done approximately without the measurement of GDS sizes. This took me about 7 hours of work. Using generated SVG from G-Visit took 1 hour and 10 minutes of work. These results are compared with G-Visit run time in Tab. 3.1.

Table 3.1: Overview of inverter 3D visualization methods.

| Method | Time (s) |
|---|---|
| manual | 25 200 |
| manual with generated SVGs | 4 200 |
| G-Visit | 25 |

### 3.3.2 Layout Size Influence

With bigger layouts, the run time of G-Visit is growing. A graph shows the time dependency on transistor count (Fig. 3.10); data have been measured by the cProfile module in Python on a laptop with the processor i5-8250U and 8 GB RAMs.

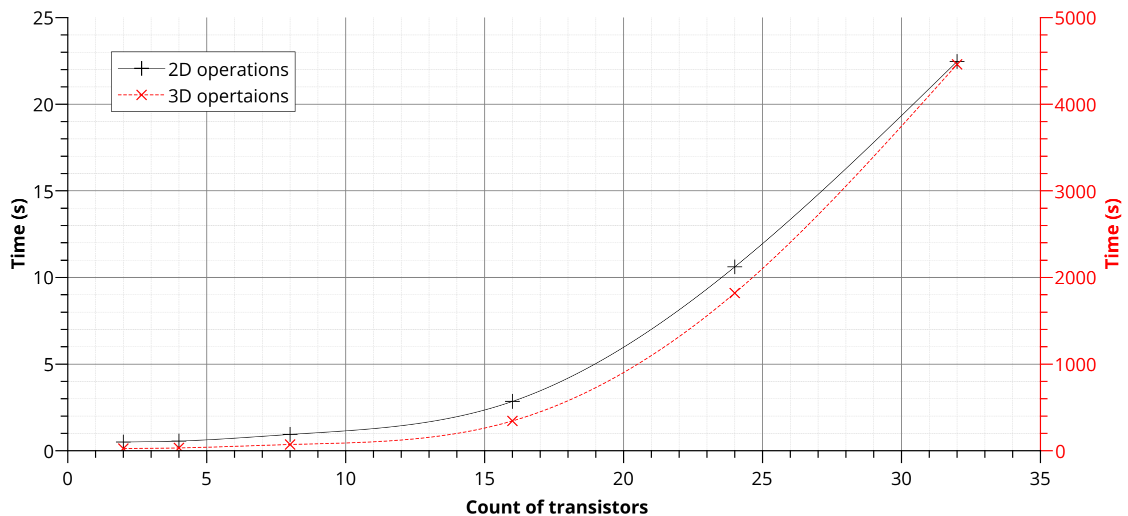



Figure 3.10: Graph of time dependency on transistor count (dual Y-axis).

# 4. Future Work

The development of G-Visit will be continued. One of things that can be improved is optimization and transition to newer version (2.91) of Blender, that is supposed to have less bugs in boolean operations.

## 4.1 3D TCAD

The biggest step planned for the future is the possibility of getting the 3D object for Silvaco TCAD and be able to run simulations. This step can bring G-Visit to professional tools from companies focused on IC simulations, such as Victory Process, Gds2Mesh, and Sentaurus TCAD.

There are two ways, how to do that. Generation of 3D from 2D layout data, or using generated 3D object in Blender.

The first method may be easier to do since X-Y coordinates are already mapped and are needed to add the Z coordinates. This method should also be easier to set doping profiles.

The existing 3D object from Blender should be closer to the actual IC from manufacturing. That may be motivation for using the method of importing data from Blender. The operation of data transition from Blender can be really challenging. It has to be correctly mapped so that could be correctly transferred to TCAD. This can be divided into two basic steps of data transition to TCAD. At first, it should be done without a bevel; afterward, the bevel could be approximated.

All mentioned presumptions may be verified by comparing cross-sections generated from the 3D objects with cross-sections from 2D data in TCAD. The generation from 2D data is already implemented. The generation from 3D could also be a good way to better understanding mapping for planned 3D TCAD.

# 5. Discussion

This chapter analyzes the results of the whole work (chapter 3). It also briefly overviews methods and mechanisms necessary for a deeper understanding of G-Visit.

Processing 2D data and creating 3D objects in this work is a complex issue. During this process, real manufacturing processes are taken into account, making the process even more complicated. There are tools with a similar focus, and only a few can do what G-Visit can (Tab. 5.1).

Table 5.1: Tools capabilities comparison.

| Tool | 2D view | Cross-section | 2D sim. | 3D view | Adv 3D view | 3D sim. |
|---|---|---|---|---|---|---|
| **G-Visit** | ✓ | ✓ | ✓ | ✓ | ✓ | planned |
| Victory Process | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Gds2Mesh | × | ✓ | ✓ | ✓ | ○ | ✓ |
| Sentaurus TCAD | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| GDS2POV | ✓ | × | × | ✓ | × | × |
| GDS3D | × | × | × | ✓ | × | × |
| ShapeshifteR | × | × | × | ✓ | × | × |
| Qckvu3 | × | × | × | ✓ | × | × |

This table (Tab. 5.1) is showing tools with similar focus and comparing their capabilities. It compares 2D capabilities, such as 2D view of the full layout (with layers added during manufacturing), the possibility of the cross-section visualization, and 2D simulation. It also compares a 3D visualizations and simulations. The column for "Adv 3D view" compares the possibility of the advanced 3D view that takes into account real manufacturing processes.

Standard symbols used in the table are, ✓ for confirmation, × for rejection of statements. There is one special symbol ○; in this case, the tool is capable of advanced 3D visualization, but it is recommended not to use that for bigger structures. It is a compromise that enables generating larger structures in a shorter time.

## 5.1 Visualization

The visualization is important to better understanding layers used in IC manufacture. It may also reveal potential issues of manufactured structures. The visualization also can serve for debugging the G-Visit functionality. It is a first output, where can be actually observed inaccuracies without going trough variables in Python.

Visualization provided by G-Visit shows all layers used in manufacturing, which IC layout design tools do not show. These tools contain those layers that are necessary for IC mapping. Other layers are automatically added during the manufacturing process.

Adding those layers is done by the correct definition file and 2D operations. The visualization is in SVG format. This format based on the vector graphic can be viewed and edited by free tools. This means that the layers (which are grouped by material) can be shown separated for higher clarity (Fig. 3.2).

The 3D visualization can achieve better insight into the IC structure. G-Visit uses Blender for its operations, so the default file format is *.blend. This tool also supports export to OBJ format, one of the most used formats in 3D visualization. This export has coloration issues. This is probably caused by Blender itself. While testing, the manual export in Blender of the same structure exported using G-Visit script had more issues, even though the same version of bpy was used.

The bugs of boolean operations in bpy, created a difficulties to overcome. One of the biggest challenges is STI layer. This layer can be generated (Fig. 3.5), but it fails when it has to do boolean operations with other layers. This may be related to the issue shown in Fig. 2.8c.

Mentioned issues does not outbalance positives this work brings. During 3D visualization materials can be beveled, nested in each other and even create layers theirs faces touches. The bug that is causing issue (Fig. 2.8a, 2.8b) in most cases has been successfully bypassed.

Using a generated *.blend file has its advantages. The user can freely turn on and off layers, move with them freely and even create cross-sections. This can be really helpful for demonstrative purposes.

## 5.2 TCAD

The generation of TCAD makes from G-Visit not only visual tool, but a technological tool that enables import of semiconductor structures in TCAD. The mesh is calculated from prepared regions to ensure correct rendering. Electrodes are only addressed as the metal during the whole process, so they cannot be really scripted as TCAD electrodes. They have to be set manually, same as simulations.

Shown visualization does not show contact to PMOS source. It is caused by a cross-section line missing the contact. If something like this happens, the user can easily add this contact manually for TCAD simulations. The visualization also uses thicker salicide, and GOX to make these layers visible.

The possibility of setting mesh density can help with the balance between the speed and precision of simulations. The denser is the mesh, the more precise and time-consuming the simulation is. When is the mesh too dense, it resolves in Deckbuild error.

Simulations are beneficial and can show the parameters of the ICs. An NMOS transistor, which is part of the generated inverter, is used to demonstrate simulations. GOX of this NMOS is set on 8 nm. The threshold of this transistor is about 1 V. This value has been estimated from the transfer characteristic (Fig. 3.8). The second simulation shows the output I-V characteristic of this NMOS transistor (Fig. 3.9). This characteristic can be used for the determination of the transistor's operation region. An approximate curve of this interface (pinch off voltage) has been manually added.

Those were only exemplary simulations. There could be more simulations of this NMOS or even simulations of the inverter, but it could be really time-consuming.

## 5.3 Time Complexity

Time complexity is an important factor. Valuable would time comparison with other tools, but this information has not been found. Anyway the time comparison with manual work is really illustrative and shows the advantages of G-Visit (Tab. 3.1).

The mentioned table is missing one important parameter, and it is time for

setting the definition file. Since it is one-time work, it can be evenly split for every single use of these definitions (Fig. 5.1). The assumption is that this definition file can be created even faster than the 3D visualization itself. Even when manual 3D visualization time is exceeded, the final time investment to definitions will be worth it after few uses. The repeated use of the definition file can suppress the time towards zero.

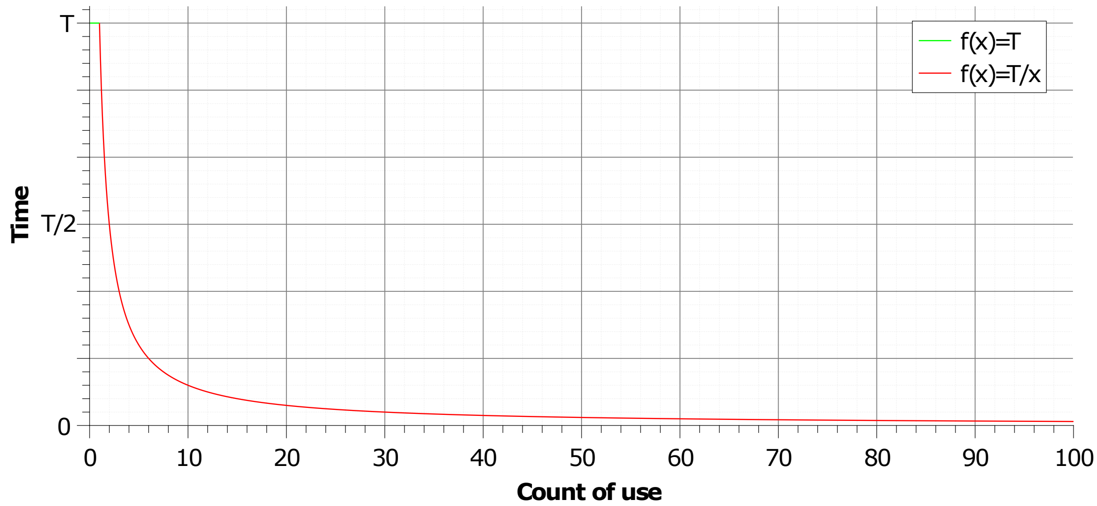



Figure 5.1: Graph of time amortization.

# Conclusion

During the work on master's thesis has been developed tool G-Visit. This tool was created to help the development of IC by providing visualization and input data for TCAD. These capabilities are compared with other similarly focused tools in Tab. 5.1. It shows that G-Visit holds on with offered professional tools, such as Victory Process (Silvaco), Gds2Mesh (Cogenda), and Sentaurus TCAD (Synopsys).

New Python modules were specifically created for G-Visit. A good example is the svgCreator, a module for SVG generation that allows grouping layers. The module genIn generates the input file for TCAD. These two modules have been created because other available modules did not provide the necessary functionality. Both of them use only the os module for paths in the operating system. G-Visit uses even more created custom created modules, but they are dependent on other modules' functions (Fig. 2.3).

This tool can accelerate the work of its users. It can generate the TCAD input file together with 2D visualization in 0.5 seconds and 3D visualization in 25 seconds. The time of 3D generation compared to the one done manually is more than 1 000 faster (Tab. 3.1). The graph of time dependency on transistors illustrates the time complexity (Fig. 3.10). The time investment for new technology definitions is amortized by use of these definitions (Fig. 5.1).

Section 1.2 introduces semiconductor structures used in IC manufacture, and it also explains basic physical principles of IC and their parasitic effects. With that in mind has been developed G-Visit. The development has been done in these steps:

- processing of GDSII data,
- 2D logical operations and their definitions,
- advanced 3D structure generation and its definition,
- import of data to TCAD together with material properties, which creates a structure, that can be used for user defined simulations.

The development of G-Visit proved that the visualization and TCAD export is a complex matter. There will always be the possibility of further development and enhancements. Some of them are described in chapter 4.

# Bibliography


[1] R. Light, *GDS2POV*, (2013). GitHub, Inc.. Accessed: May 17, 2020. [Online]. Available: https://github.com/ralight/gds2pov

[2] Persistence of Vision Raytracer Pty. Ltd, *Persistence of Vision Raytracer*, (2013). Accessed: May 17, 2020. [Online]. Available: http://www.povray.org/

[3] R. Light. *GDS2POV*. Accessed: May 17, 2020. [Online]. Available: https://atchoo.org/gds2pov/

[4] University of Twente – Integrated Circuit Design Group, "3D GDSII Viewer Goes Open Source," *University of Twente*, 2012. Accessed: Oct. 15, 2019. [Online]. Available: https://www.utwente.nl/en/eemcs/icd/news/2012/3/59583/3d-gdsii-viewer-goes-open-source

[5] *GDS3D*, (2017). GitHub, Inc.. Accessed: Oct. 15, 2019. [Online]. Available: https://github.com/skuep/GDS3D

[6] D. Carron, "3D GDSII – Out of the Box," 2004. Accessed: Oct. 18, 2019. [Online]. Available: http://shapeshifter.free.fr/index.htm

[7] Artwork Conversion Software, Inc., "Layout Viewer for GDSII and OASIS," *Artwork Conversion Software, Inc.*. Accessed: Jun. 3, 2020. [Online]. Available: https://www.artwork.com/gdsii/qckvu3/.

[8] Artwork Conversion Software, Inc., "Extract 3D – 3D Geometry Extraction Plugin for Qckvu," *Artwork Conversion Software, Inc.*. Accessed: Jun. 3, 2020. [Online]. Available: https://www.artwork.com/gdsii/qckvu3/plugin/3d/index.htm

[9] Silvaco, Inc.. *Victory Process*, (2019). Accessed: May 20, 2020. [Online]. Available: https://silvaco.com/products/tcad/process_simulation/victory_process/victory_process_19.pdf

[10] Silvaco, Inc.. *Victory Process – Full Physical 3D Semiconductor Simulator*, (Nov. 2012). Accessed: May 20, 2020. [Online]. Available: https://www.silvaco.com/content/kbase/VICTORY_Presentation_Nov2012.pdf?20140113

[11] Silvaco, Inc., "RC Extractor for Realistic 3D Structures," *Silvaco, Inc.*. Accessed: May 27, 2020. [Online]. Available: https://www.silvaco.com/products/interconnect_modeling/rc_extraction/clever.html

[12] Cogenda, Pte Ltd., "Gds2Mesh," *Cogenda, Pte Ltd.*. Accessed: Jun. 2, 2020. [Online]. Available: https://www.cogenda.com/article/Gds2Mesh

[13] Cogenda, Pte Ltd.. *Gds2Mesh – 3D TCAD Model Constructor*, (2008). Accessed: Jun. 2, 2020. [Online]. Available: http://www.i-vis.co.jp/pdf/cogenda/gds2mesh.pdf

[14] Cogenda, Pte Ltd., "Products" *Cogenda, Pte Ltd.*. Accessed: Jun. 2, 2020. [Online]. Available: https://www.cogenda.com/article/products

[15] Synopsys, Inc.. *Sentaurus TCAD – Industry-Standard Process and Device Simulators*, (2012). Accessed: Jun. 5, 2020. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/silicon/datasheets/sentaurus_ds.pdf

[16] Synopsys, Inc.. *Sentaurus™ Process User Guide*, (2015). Accessed: Jun. 15, 2020. [Online]. Available: http://www.sentaurus.dsod.pl/manuals/data/sprocess_ug.pdf

[17] C. Kernstock, Z. Stanojević, O. Baumgartner and M. Karner, "Layout-Based TCAD Device Model Generation," in *2015 Int. Conf. on Simulation of Semiconductor Processes and Devices (SISPAD)*, Washington, DC, USA, Sep. 2015, pp. 198 – 201, doi: 10.1109/SISPAD.2015.7292293.

[18] G. E. Moore, "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.," in *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33 – 35, Sep. 2006, doi: 10.1109/N-SSC.2006.4785860.

[19] R. R. Schaller, "Moore's law: past, present and future," in *IEEE Spectrum*, vol. 34, no. 6, pp. 52 – 59, Jun. 1997, doi: 10.1109/6.591665.

[20] D. Rotman, "We're not prepared for the end of Moore's Law," Feb. 2020. Accessed: Jun. 6, 2020. [Online]. Available: https://www.technologyreview.com/2020/02/24/905789/were-not-prepared-for-the-end-of-moores-law/

[21] R. Chau, "Process and Packaging Innovations for Moore's Law Continuation and Beyond," *IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2019, pp. 1.1.1 – 1.1.6, doi: 10.1109/IEDM19573.2019.8993462.

[22] R. S. Williams, "What's Next? [The end of Moore's law]," in *Computing in Science & Engineering*, vol. 19, no. 2, pp. 7 – 13, Mar. – Apr. 2017, doi: 10.1109/MCSE.2017.31.

[23] L. Xiu, "Time Moore: Exploiting Moore's Law From The Perspective of Time," in *IEEE Solid-State Circuits Magazine*, vol. 11, no. 1, pp. 39 – 55, 2019, doi: 10.1109/MSSC.2018.2882285.

[24] Wikipedia, "Transistor count," *Wikipedia - The Free Encyclopedia,*. Accessed: May 6, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Transistor_count#Microprocessors

[25] J. Lienig and J. Scheible *Fundamentals of Layout Design for Electronic Circuits* Cham, Switzerland: Springer Nature Switzerland AG 2020, 2020.

[26] A. Hastings, *The Art of Analog Layout*, 2nd ed. Upper Saddle River, NJ, USA: Pearson, 2006.

[27] Wafer World, Inc., "Silicon Wafer | Silicon Wafer and its Different Diameters," *Wafer World, Inc.*, (Aug. 2019). Accessed: Jun. 6, 2020. [Online]. Available: https://www.waferworld.com/silicon-wafer-different-diameters/

[28] B. El-Kareh and L. N. Hutter, *Silicon Analog Components – Device Design, Process Integration, Characterization, and Reliability*, 2nd ed. Cham, Switzerland: Springer, 2020.

[29] Siltronic AG. *Siltronic Investor Presentation*, (2019). Accessed: Jun. 6, 2020. [Online]. Available: https://www.siltronic.com/fileadmin/investorrelations/Pr%C3%A4sentation/Siltronic_Investor_Presentation_January_2019.pdf

[30] G. J. Schmitz and U. Prahl, *Handbook of Software Solutions for ICME*, Berlin, Germany: Springer, 2016.

[31] U. W. Pohl, "Methods of Epitaxy," in *Epitaxy of Semiconductors – Introduction to Physical Principles*, Berlin, Germany: Springer, 2013, ch. 7, pp. 275 – 313.

[32] M. R. Oliver, "Metal Polishing Processes," in *Chemical-Mechanical Planarization of Semiconductor Materials*, Berlin, Germany: Springer, 2004, ch. 3, pp. 41 – 46.

[33] Sensofar Metrology, "Surface Metrology for In-Situ Pad Monitoring in Chemical Mechanical Planarization (CMP)," *AZO Materials*, (2019). Accessed: May. 4, 2020. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=12527

[34] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 2nd ed. New York, NY, USA: McGrawHill, 2017.

[35] C. Contiero, A. Andreini and P. Galbiati, "A new integrated silicon gate technology combining bipolar linear, CMOS logic, and DMOS power parts," in *IEEE Transactions on Electron Devices*, vol. 33, no. 12, pp. 2025 – 2030, Dec. 1986, doi: 10.1109/T-ED.1986.22862.

[36] STMicroelectronics. "BCD." ST life.augmented. Accessed: Oct. 3, 2019. [Online]. https://www.st.com/content/st_com/en/about/innovation---technology/BCD.html

[37] E. A. Vittoz, "Passive Components and Parasitic Effects," presented at the MEAD course on "Advanced Analog CMOS IC Design," EPFL, Lausanne, Swiss, Aug. 24-28, 2015. doi: 10.13140/RG.2.1.3223.2726.

[38] R. C. -Y. Fang and J. L. Moll, "Latchup Model for the Parasitic p-n-p-n Path in Bulk CMOS," in *IEEE Transactions on Electron Devices*, vol. 31, no. 1, pp. 113 – 120, Jan. 1984, doi: 10.1109/T-ED.1984.21484.

[39] Michael Pinter. "Latchup." Graz University of Technology. Accessed: May. 9, 2021. [Online]. Available: http://lampx.tugraz.at/~hadley/psd/L13/latch-up/Latch-Up.html

[40] A. Brambilla, P. Mafezzoni, L. Bortesi and L. Vendrame, "Measurements and Extractions of Parasitic Capacitances in ULSI Layouts," in *IEEE Transactions*

*on Electron Devices*, vol. 50, no. 11, pp. 2236 – 2247, Nov. 2003, doi: 10.1109/TED.2003.818150.

[41] H. Wu *et al.*, "Parasitic Resistance Reduction Strategies for Advanced CMOS FinFETs Beyond 7nm," in *2018 IEEE International Electron Devices Meeting (IEDM)*, pp. 35.4.1 – 35.4.4, Dec. 1-5, 2018, doi: 10.1109/IEDM.2018.8614661.

[42] P. Jay and A. D. Darji, "Analysis of the source/drain parasitic resistance and capacitance depending on geometry of FinFET," in *2015 11th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp. 298 – 301 Jun. 29-Jul. 2, 2015, doi: 10.1109/PRIME.2015.7251394.

[43] T. Matsukawa *et al.*„ "Variability Origins of Parasitic Resistance in FinFETs With Silicided Source/Drain," in *IEEE Electron Device Letters*, vol. 33, no. 4, pp. 474 – 476, April 2012, doi: 10.1109/LED.2012.2182755.

[44] S. DiBartolomeo. "All About Calma's GDSII Stream Format." Artwork Conversion Software. Accessed: May 10, 2021. [Online]. Available: https://www.artwork.com/gdsii/gdsii/

[45] "GDSII Format." Accessed: Oct. 5, 2019. [Online]. Available: https://boolean.klaasholwerda.nl/interface/bnf/gdsformat.html#rec_bgnstr

[46] M. Duò. "15 Best Image File Types (Pros vs Cons + Use Cases for Each Format)." KINSTA BLOG. Accessed: May 10, 2021. [Online]. Available: https://kinsta.com/blog/image-file-types/

[47] Wolfram. "Vector Graphics Formats." Wolfram Language & System. Accessed: May 10, 2021. [Online]. Available: https://reference.wolfram.com/language/guide/VectorGraphicsFormats.html

[48] J. Trythall. "Pocket Guide to Writing SVG." Pocket Guide to Writing SVG. Accessed: Nov. 3, 2019. [Online]. Available: https://svgpocketguide.com/book/#intro

[49] W3Schools. "SVG Tutorial." W3Schools. Accessed: Feb. 10, 2020. [Online]. Available: https://www.w3schools.com/graphics/svg_intro.asp

[50] Scan2TCAD. "DXF Files." Scan2TCAD. Accessed: Feb. 10, 2020. [Online]. Available: https://www.scan2cad.com/category/dxf/

[51] Victoria. "The DXF File Specification." Scan2TCAD. Accessed: Feb. 10, 2020. [Online]. Available: https://www.scan2cad.com/dxf/file-specification/

[52] D. Chakravorty. "All you need to know about 3D file formats – The Most Common 3D File Formats." All3DP, Aug. 2019. Accessed: Mar. 16, 2020. [Online]. Available: https://all3dp.com/3d-file-format-3d-files-3d-printer-3d-cad-vrml-stl-obj/

[53] C. Conlan, *The Blender Python API: Precision 3D Modeling and Add-on Development*, Berkley, CA, USA: Apress, 2017.

[54] A. Grahn, *The media9 Package*, 2021. Accessed: May 11, 2021. [Online]. Available: https://texdoc.org/serve/media9.pdf/0

[55] Silvaco, Inc.. *Deckbuild User's Manual*, (Version 4.2.0.R). October 2015. Accessed: Oct. 15, 2019. [Online]. Available: https://dynamic.silvaco.com/dynamicweb/jsp/downloads/DownloadManualsAction.do?req=silen-manuals&nm=deckbuild

[56] E. Meshcheryakov, *python-gdsii v0.2.1 documentation*, 2010. Accessed: Oct. 24, 2019. [Online]. Available: https://pythonhosted.org/python-gdsii/

# List of Figures

1.1 GDS3D window look [4]. . . . . . . . . . . . . . . . . . . . . . . . . . 8
1.2 ShapeshifteR Koala visualization [6]. . . . . . . . . . . . . . . . . . . . 9
1.3 Qckvu3 – Extract 3D visualization window [8]. . . . . . . . . . . . . . 9
1.4 Victory process buffered oxidation [10]. . . . . . . . . . . . . . . . . . . 10
1.5 Structure from Victory process [11]. . . . . . . . . . . . . . . . . . . . . 10
1.6 Structure from Gds2Mesh [12]. . . . . . . . . . . . . . . . . . . . . . . . 11
1.7 Simulated mesh and structure in Sentaurus Device [15]. . . . . . . . . 12
1.8 Graphical representation of Moore's Law (data from [24]). . . . . . . 13
1.9 Diagram of oxidation furnace; [26] used as template. . . . . . . . . . . 15
1.10 Comparison of wet (isotropic) and dry (anisotropic) etching [25]. . . . 16
1.11 Schematic representation of the idealized LOCOS process [25]. . . . . 17
1.12 Uneven surfaces caused by oxide process [25]. . . . . . . . . . . . . . . 17
1.13 Demonstration of diffusion to silicon and graph of dopant concentration depending on depth [25]. . . . . . . . . . . . . . . . . . 18
1.14 Diagram of an ion implanter; [25] and [26] used as template. . . . . . 19
1.15 Demonstration of ion doping to silicon and graph of depth dependency on concentration and time; (a) visualization of ion implantation to substrate; (b) visualization of final diffusion area [25]. . . . . . . . . . 19
1.16 Reactor types for metalorganic vapor-phase epitaxy; (a) schematic of horizontal reactor; (b) Schematic of vertical reactor [31]. . . . . . . . 20
1.17 Schematic diagram of chemical-mechanical polishing [33]. . . . . . . . 21
1.18 Visualization of dishing caused by CMP [32]. . . . . . . . . . . . . . . 22
1.19 Buried layer generation [25]. . . . . . . . . . . . . . . . . . . . . . . . 22
1.20 STI process visualization [25]. . . . . . . . . . . . . . . . . . . . . . . 23
1.21 DTI process visualization [25]. . . . . . . . . . . . . . . . . . . . . . . 23
1.22 Structuring of polysilicon [25]. . . . . . . . . . . . . . . . . . . . . . . 24
1.23 Typical view of the routing layers [25]. . . . . . . . . . . . . . . . . . 24
1.24 Difference between metallization without (left) and with (right) planarization [25]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
1.25 Damascene technique used to create copper interconnections [25]. . . 25

1.26 NMOS transistor cross-section [25]. . . . . . . . . . . . . . . . . . . . . . 26
1.27 CMOS process options (p-doped substrates) [25]. . . . . . . . . . . . . 27
1.28 CMOS process options with LDD [25]. . . . . . . . . . . . . . . . . . . 27
1.29 Visualization of BCD composition [36]. . . . . . . . . . . . . . . . . . . 28
1.30 Latchup model inside of CMOS invertor [39]. . . . . . . . . . . . . . . 29
1.31 Cube embedded using U3D format (click-on 3D activation). . . . . . 32

2.1 Draft of G-Visit icon. . . . . . . . . . . . . . . . . . . . . . . . . . . 33
2.2 Flowchart of G-Visit workflow. . . . . . . . . . . . . . . . . . . . . . . 34
2.3 Diagram of Python modules hierarchy. . . . . . . . . . . . . . . . . . . 35
2.4 Representation of 2D boolean operations with names of their functions. 38
2.5 Representation of 2D operations with names of their functions. . . . . 39
2.6 Simplified flowchart of "grow" function. . . . . . . . . . . . . . . . . . 40
2.7 3D operations using extrude method with name of functions (click-on 3D activation). . . . . . . . . . . . . . . . . . . . . . . . . . . 41
2.8 Cross-section of problematic positions for boolean operations in blender library - bpy (2.82.1); (a) faces on the same level; (b) touching faces; (c) complicated profile of face. . . . . . . . . . . . . . 41
2.9 Boolean operations overview (click-on 3D activation); (a) cross-section of substrate and operator object; (b-d)function bool3D operation with different inputs. . . . . . . . . . . . . . . . . . . . . 42
2.10 Simplified flowchart of genIn module principle. . . . . . . . . . . . . . 43
2.11 Bad meshing. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
2.12 Visualization of mesh precision dependence on the grid parameter. . . 44

3.1 Screenshot of input GDS file layout from KLayout (legend added). . . 45
3.2 Layout generated by G-Visit; (a-c) decomposed layout; (d) STI mask. 46
3.3 Legend for figures (3.2a-3.2c). . . . . . . . . . . . . . . . . . . . . . . 47
3.4 Final view of generated IC (click-on 3D activation). . . . . . . . . . . 47
3.5 Generated STI layer (click-on 3D activation). . . . . . . . . . . . . . 48
3.6 Screenshot of decomposed inverter in Blender (legend added). . . . . 48
3.7 TCAD output structure. . . . . . . . . . . . . . . . . . . . . . . . . . 49
3.8 Transfer characteristics of the transistor. . . . . . . . . . . . . . . . . 49

3.9 Output I-V characteristics of NMOS. . . . . . . . . . . . . . . . . . . . . 50
3.10 Graph of time dependency on transistor count (dual Y-axis). . . . . . . 51

5.1 Graph of time amortization. . . . . . . . . . . . . . . . . . . . . . . . . . 58