Universitat Autònoma de Barcelona
Departament de Microelectrònica i Sistemes Electrònics

From characterization strategies to PDK & EDA Tools for Printed Electronics

Francesc Vila Garcia

Memòria de Tesi
presentada per optar al títol de

Doctor en Microelectrònica i Sistemes Electrònics

Setembre 2015

Universitat Autònoma de Barcelona

CSIC

Dr. Lluís Terés Terés, Científico Titular del Consejo Superior de Investigaciones Científicas i Professor Associat del Departament de Microelectrònica i Sistemes Electrònics,

Certifica

que la Memòria de Tesi *From characterization strategies to PDK & EDA Tools for Printed Electronics* presentada per Francesc Vila Garcia per optar al títol de Doctor en Microelectrònica i Sistemes Electrònics s'ha realitzat sota la seva direcció a l'Institut de Microelectrònica de Barcelona pertanyent al Centro Nacional de Microelectrónica del Consejo Superior de Investigaciones Científicas i ha estat tutoritzada en el Departament de Microelectrònica i Sistemes Electrònics de la Universitat Autònoma de Barcelona.

Director | Dr. Lluís Terés Terés ....

........................., a ...... de .... de ....

If we knew what it was that we were doing,
it would not be called research, would it?

– Albert Einstein

Resum

Durant els últims anys, les tecnologies d'electrònica impresa (PE) estan atraient molta atenció, principalment degut a que es poden fabricar grans àrees, i són una alternativa de baix cost a la microelectrònica tradicional. D'entre totes les tecnologies disponibles, la fabricació emprant impresores d'injecció de tinta (inkjet) resulta particularment interessant, al ser un mètode d'impressió digital (reduint els residus generats al fabricar), i no tenir contacte amb el substrat (per tant permet la utilització de molts tipus diferents de substrats).

La tecnologia inkjet encara està patint un gran desenvolupament, cosa que fa difícil que es puguin dissenyar circuits i sistemes sense tenir un gran coneixement sobre els processos que hi ha al darrere. A més a més, la mancança d'eines específicament dissenyades per a inkjet crea un gran distància entre els dissenyadors i els tecnòlecs responsables de desenvolupar la tecnologia, dificultant així una adopció generalitzada de la tecnologia inkjet.

Aquesta tèsi contribueix a apropar els dissenyadors a la tecnologia, proposant i adaptant fluxes i kits de disseny existents i basats en microelectrònica, a les tecnologies inkjet, complementant-los amb eines específiques per adaptar-los a les peculiaritats de l'inkjet. D'aquesta manera aconseguim un camí directe des del disseny a la fabricació, abstraient els detalls tecnològics del disseny.

A més a més, per tancar el camí entre disseny i la fabricació, aquesta tèsi proposa un entorn semi-automàtic de caracterització, que es fa servir per analitzar el comportament de la tinta dipositada, inferint quines correccions són necessàries per a què el resultat fabricat correspongui tant com sigui possible al disseny. El coneixement extret d'aquest pas s'incorporarà en una eina EDA específica que analitza i aplica automàticament les correccions extretes a un disseny.

Abstract

During last years, Printed Electronics technologies have attracted a great deal of attention due to being a low-cost, large area electronics manufacturing process. From all available technologies, inkjet printing is of special interest, because of its digital nature, which reduces material waste; and being a non-contact process, which allows printing on a great variety of substrates.

Inkjet printing is still on heavy development, thus making designing for it difficult without an in-depth knowledge of how the manufacturing process works. In addition, currently there is a lack of specific tools aiding to design for it, creating a large gap between designers and technology developers and difficulting a wide adoption of this particular technologies.

The work presented on this thesis contributes to bridge the existing gap between designers and technology developers by proposing and adapting existing microelectronics-based design flows and kits, while complementing them with custom, PE specific Electronic Design Automation tools; to achieve a direct path from design to manufacturing, and abstract technology specific details from the design stages. This is achieved by combining a design flow with a PE Process/Physical Design Kit, and a set of EDA tools adapted to PE.

In addition, to finally bridge design and manufacturing, this thesis proposes a semi-automated characterization methodology, used to analyze the deposited ink behavior, and infer all necessary corrections needed to ensure that the final fabricated result corresponds as much as possible to the intended design. This knowledge is then integrated into an specific EDA framework which will perform the aformentioned corrections automatically.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor, Dr. Luís Terés Terés, for continuous support of my PhD study and related research. His guidance helped me in all the time of the research and writing of this thesis. In addition, I would like to thank Dr. Jordi Carrabina, for his insightful comments and, together with Lluís, giving me the oportunity to work in this thesis.

Besides, I would also like to thank my colleagues at ICAS, specially Jofre Pallarès, for his great support and knowledge, giving advice and guiding on the difficult parts of this research. Thanks all for the great time spent working on IMB. Special thanks also to Eloi, Carme, Adrià, and all the people working in the TDK4PE project. It has been a pleasure to work with you in this great and ambicious project, allowing me to learn from you. Special thanks for Arjen, Niels, Remco(s), and Jan, from PhoeniX for allowing me to do a short stay in Enschede, giving me the oportunity to work with you after this thesis, and starting a new exciting adventure.

In addition, I would like to thank my former university classmates, some still remaining in the university pursuing their PhD's. Thank you very much to Albert Saa and Xicola, Ferran, Toni, Miguel, Arturo. Thanks for the discussions during lunch, the everyday "creative Friday", and all of your support. You have helped me to deal with the pressure of doing a PhD, and provided me with very useful directions on topics that I was not familiar with.

My sincere thanks to Ricardo, Salva, and Ciscu, my flatmates during all of these years. It has been a pleasure living with you, enjoying great times and discussions.

In this agreements I cannot forget to thank Cristina, Jordi, Mònica, Mimar, and

all of my friends from Lloret. You have been a huge support during this time, and helped me on my most tough moments. A huge and most sincere thank you.

Last but not least, I want to thank my family, specially my parents. Without their support and education I would not be on the place I am today. Thanks for all the effort and for giving me the oportunity to pursue, first an Engineering bachelor, and then my PhD.

Contents

1 Introduction **1**

1.1 PE Technologies 1

1.2 State of the art and motivation 3

1.3 Thesis context 7

1.4 Aims and objectives 8

2 Design Flows, Tools and Kits **11**

2.1 Abstraction levels and design flows 11

2.2 PE design flows 13

2.3 Process Design Kits 15

2.3.1 Necessary process information 16

2.3.2 XML interface 17

2.4 Characterization strategies and setup 18

2.5 Conclusions 19

3 Specific Design Kits and EDA tools for PE **21**

3.1 EDA tools selection . . . 21
3.1.1 FOSS tool set . . . 22
3.1.2 Commercial tools . . . 26
3.2 EDA tool customizations . . . 26
3.2.1 Free/Open source tool selection . . . 26
3.2.2 Commercial package . . . 30
3.3 XML database . . . 30
3.3.1 XML representation . . . 30
3.3.2 FOSS tool set . . . 41
3.3.3 Commercial tool set . . . 46
3.4 Conclusions . . . 52

4 Image based technology characterization 55
4.1 Experimental approach . . . 55
4.1.1 Experiment generation . . . 56
4.2 Analysis . . . 63
4.2.1 Image processing operations . . . 63
4.2.2 Misalignment . . . 68
4.2.3 Score calculation . . . 71
4.2.4 Statistical procedures . . . 74
4.2.5 Electrical tests . . . 80
4.3 Conclusions . . . 85

5 Automatic Compensations 89
5.1 *Layout2Bitmap* . . . 89
5.1.1 Tool description . . . 89
5.1.2 Design loading . . . 90
5.1.3 Design compensation . . . 91

5.2 *Layout2Bitmap* into design flows 102
5.2.1 Free/OSS flow integration: Glade 104
5.2.2 Commercial flow integration: *MaskEngineer* 104
5.3 Results and conclusions 105

6 Global conclusions and further activities 109

List of Figures

1.1 Inkjet printing printhead diagram. 3

1.2 Different printing imperfections. 4

1.3 OpenPDK use cases. 6

2.1 Evolution of technology and design abstraction levels. 12

2.2 Diagram of the proposed design flow for PE. 14

2.3 PDK development and characterization, and EDA customization loop. 15

2.4 PDK diagram with an abstraction layer. 17

2.5 Characterization environment setup. The diagram contemplates both optical and electrical characterization. 18

3.1 Complete design flow, containing all used tools and intermediate formats. The diagram contains both the FOSS and the commercial flow. This diagram can be seen as a detailed version of Fig. 2.2 . . . 27

3.2 General information and manipulation strategy. For each part, the proposed file formats are shown. 31

3.3 Fundamental set of design rules and basic manipulation operations. . 36

3.4 Sample of complex design rules defined combining several layers. . . 37

3.5 XML usage strategies. Shows direct usage by the tools or usage via custom converter. 42

3.6 *Glade* customizations. 45

3.7 Comparison of the integrated model in ngspice and actual device characterization measurements. 46

3.8 PDK generation for PDAFlow API tools. The arrows represent the information flow. Dashed lines represent the added implementation into PDA API code. 47

3.9 *MaskEngineer* cross-section/layer mapping philosophy, w is the shape width. The layer names/colors come from the TDK4PE technology. 48

3.10 Result of loading the design kit into *MaskEngineer*, and the resulting technology layers. 49

4.1 General flow diagram of characterization procedure. This diagram contemplates both optical and electrical characterization. 57

4.2 Generated experiment layout. The resulting design comprises 1) the substrate alignment marks, 2) the camera alignment structure, and 3) the experiment samples. 58

4.3 Set of different test structures with their parameters. 61

4.4 Set of compensations applied to a L test structure. The *None* corresponds to the control group. 62

4.5 Example of morphological operations using a disk of radius r as a structuring element. The light gray shaded region is the original shape, the black is the structuring element, and the medium gray is the resulting shape. 65

4.6 Different heuristics for connected components labelling. 66

4.7 Results of the cleaning process step by performed operations: a) captured image, b) after thresholding, and opening/closing, and d) after small features removal. 67

4.8 a) Removed map of a whole DIN A5 foil. The printing direction is indicated by the white arrows (right to left and top to bottom), b) Mean values across the x axis, and c) Autocorrelation plot. 67

4.9 Example of image misalignment. On the cross-correlation contour plot, the maximum is marked with a black × (central dot on (c)). . 69

4.10 Misalignment across a whole DIN A5 foil. The vector streamlines indicate the substrate dilation directions, and the color the amount. 70

4.11 Resulting distance map transform on the regions of interest. 72

4.12 Score median depending on test template extension. The diagram shows the drop placement with the appropriate measurements. . . . 73

4.13 Initial data distribution and profile log-Likelihood for Box-Cox transform. Note: The shaded regions correspond to σ, 2σ, and 3σ intervals. 76

4.14 General distribution of transformed score results. 77

4.15 Initial estimated means and prediction intervals for outlier filtering . 78

4.16 Prediction intervals and filtered new foil data. 79

4.17 Tukey HSD post-hoc test results. 80

4.18 Comparison of the same structure with different compensations. . . 81

4.19 OTFT structure as designed and resulting cross-section. Blue parts are the source and drain electrodes, red is the gate. The channel forms between fingers. 82

4.20 Electrical test results of the finger structures. The separation is not the actual designed separation, but the effective separation accounting to ink spreading. 83

4.21 Resulting finger structures without and with compensation. 85

4.22 3D interferometer profile and finger section cut for a compensated fingered structure. The three cuts correspond to different positions inside the structure. 86

5.1 Available primitive element types defined in GDS specification. . . . 92

5.2 Representation of the two available polygon boundaries. 93

5.3 Representation of the two available polygon boundaries. 93

5.4 Polygon with holes example. 94

5.5 Regularized set of boolean operations on polygons. 95

5.6 Sweep line approach to segment intersections. 96

5.7 Overlay of two planar subdivisions. 97

5.8 Polygon offsetting examples. The shaded region corresponds to the amount offseted (δ_1 for exterior offset, δ_2 for interior offset). 98

5.9 Winding number calculation for region containing point q. The resulting number is the same regarding which ray is shot. 99

5.10 Offset curves calculation. The shaded region corresponds to the final calculated shape. 100

5.11 Pattern matrix definition and resulting drop positions on the grid. . 101

5.12 Sweep line fill procedure. The shaded area is already filled, and the red highlighted scanline is the active region. 102

5.13 Flow diagram of the general *L2B* operation. 103

5.14 General structure of the *L2B* framework. 104

5.15 Schematic to generated bitmaps process: a) symbol, b) schematic, c) layout, d) set of generated bitmaps, and e) close view on performed compensations. *Layout2Bitmap* processes the layout on c) to generate a set of masks on d). 107

List of Tables

3.1 Summary of the main characteristics for each layout editor evaluated. 23

3.2 Summary of the main characteristics for each schematic capture program evaluated. 24

3.3 Summary of the main characteristics for each electrical simulator program evaluated. 25

4.1 Different image region identifiers. 71

4.2 Measures of skewness and kurtosis of $log(S_p)$ for different compensations. It approximates the normal distribution values $(\gamma_1 = 0, \gamma_2 = 3)$ 76

4.3 Approximated $\alpha = 0.95$ confidence intervals for proportion of working fingers. 84

List of Listings

3.1 Excerpt of the layer XSD definition. 32
3.2 Excerpt of the layer XML definition 33
3.3 Excerpt of the operation defintion schema. 33
3.4 Example of operation definition in the XML database. 34
3.5 DRC rule definition schema. 34
3.6 DRC rule definition rule definition. 35
3.7 Device model schema definition. 38
3.8 Device model XML contents. 39
3.9 Device model SPICE include line. 39
3.10 Compensation operations schema. 40
3.11 Compensation operation XML contents. 41
3.12 Layers in *Glade IC* technology file format. 43
3.13 Design Rule Check (DRC) values autogenerated file. 44
3.14 DRC script. 44
3.15 Example of PDAFlow technology base class. 50

3.16 Example of PDAFlow technology base class. 50
3.17 Example of inverter design and SPICE simulation. 51
3.18 Example of inverter code with *Layout2Bitmap* export. 52

4.1 Generation script extract. 60
4.2 General PCell structure. 61

5.1 Full listing of the export function inside the design kit definition for PDAFlow design kits, and its usage. 105

Publication list

This thesis is based in part on (and led to) the following published articles and conferences:

[1] Francesc Vila et al. *Layout to bitmap: A layout design compensator and bitmap converter for Printed Electronics*. In *LOPE-C*. 2011.

[2] Eloi Ramon et al. *Inkjet printed electronics at uab-cnm: Technology and design flow developments*. In *Proceedings of the 38th International research conference of iarigai*. The International Association of Research Organizations for the Information, Media and Graphic Arts Industries, Buadpest, Hungary, September 2011. ISBN 978-3-9812704-4-0.

[3] Jofre Pallarès et al. *PE TDK Development Methodology: a nexus between Technology & Design*. In *ECME*. 2011.

[4] Francesc Vila et al. *A complete suite of open/free EDA tools for PE physical design*. In *Design, Automation and Test in Europe*. 2013.

[5] Francesc Vila et al. *PDK development for Printed/Organic Electronics based on XML*. In *DCIS'2013: XXVIII Conference on Design of Circuits and Integrated Systems*. 2013.

[6] Francesc Vila et al. *Physical design flow for Printed Electronics based on open/free EDA tools*. In *International Conference on Organic Electronics (ICOE)*. 2013.

[7] Francesc Vila et al. *A complete suite of open/free EDA tools for PE physical design*. In *Design, Automation and Test in Europe*. 2013.

[8] Jordi Carrabina et al. *Design rules checking for printing electronic devices: concept, generation and usage.* In *IARIGAI Conference*. 2013.

[9] Stepan Sutula et al. *Teaching Mixed-Mode Full-Custom VLSI Design with gaf, SpiceOpus and Glade*. In *Proceedings of the 10th European Workshop on Microelectronics Education (EWME)*. 2014.

[10] Jofre Pallarès et al. *A freeware eda framework for teaching mixed-mode full-custom vlsi design*. In *Proceedings of the XXIX Conference on Design of Circuits and Integrated Systems*. DCIS, Madrid, Spain, November 2014.

[11] Francesc Vila and Jofre Pallarès. *CSEM_Tech1 Design Tutorial*, June 2014.

[12] Francesc Vila et al. *Semi-automatic PE Technology & Device Library Characterization for PDK Development*. In *Large-area, Organic, & Printed Electronics Convention (LOPEC)*. 2014.

[13] Francesc Vila et al. *A fully-automated methodology and system for printed electronics foil characterization*. In *Microelectronic Test Structures (ICMTS), 2015 International Conference on*, pages 188–192. March 2015. ISSN 1071-9032. doi:10.1109/ICMTS.2015.7106138.

[14] Francesc Vila et al. *A systematic study of pattern compensation methods for all-inkjet functional printing processes*. Components, Packaging and Manufacturing Technology, IEEE Transactions on. Accepted. Pending revisions.

List of Acronyms

ASPEC-TDK Application Specific Printed Electronic Circuits - Technology Design Kits . . . 7

API Application Programming Interface . . . 7

ASIC Application Specific Integrated Circuit . . . 13

CAD Computer Aided Design . . . 15

CAM Computer Aided Manufacturing . . . 97

CIF Caltech Intermediate Format

CIJ Continous Inkjet . . . 2

CSV Comma-Separated Values . . . 19

DLL Dynamic-link Library . . . 47

DoD Drop on Demand 2

DRC Design Rule Check xxi

DT Distance Transform 71

DXF Drawing Interchange Format 100

EDA Electronic Design Automation 5

ERC Electrical Rule Check 13

FEC Fluidic Effects Correction 8

FOSS Free/Open Source Software 22

GDS Graphic Database System 34

GUI Graphical User Interface 104

HDL Hardware Description Language 12

ILT Inverse Lithography Technology 13

IoT Internet of Things 1

IP Intellectual Property 11

LVS Layout Versus Schematic 8

MCS Mask Cross Sections 47

MLE Maximum Likelihood Estimator 75

OASIS Open Artwork System Interchange Standard 90

OE Organic Electronics 1

OPC Optical Proximity Correction 13

OTFT Organic Thin Film Transistor 80

PCB Printed Circuit Board 13

PCell Parameterizable Cell 9

PDF Portable Document Format 39

PDK Process/Physical Design Kit 5

PE Printed Electronics 1

RET Resolution Enhancement Techniques 13

SO Shared Object 47

TDK4PE Technology and Design Kits for Printed Electronics 7

TVD Test Vehicle Description 19

XML eXtensible Markup Language 5

XPath XML Path Language 42

XSD XML Schema Definition 32

XSL eXtensible Stylesheet Language 41

XSL-FO XSL Formatting Objects 42

XSLT XSL Transformations 42

Introduction | 1

Nowadays there is an increasing interest in the field of Organic Electronics (OE) and Printed Electronics (PE), in order to have low-cost, flexible and large area electronics for several consumer applications [1], such as wearable technologies and healthcare applications [2, 3], Internet of Things (IoT) applications [4], energy harvesting and storage [5, 6], or lighting [7].

The preference for PE over traditional silicon-based microelectronics in this area is due to the necessity of having large area, thin, and flexible systems, characteristics which PE excels in. In addition to this advantages, PE gives a dramatic reduction on production costs, because both the equipment and the materials used are several orders of magnitude cheaper than the ones used in silicon-based microelectronics. The use of organic materials makes it more environmental friendly.

For these reasons, this research tries to push forward the evolution of these technologies by providing characterization and correction procedures to aid the move from design to manufacturing, and making it more accessible to designers coming from the electronic world by bridging the gap between design and technology.

1.1 PE Technologies

PE (or Plastic Electronics, Organic Electronics, or Flexible Electronics) is a term that comprises a set of different printing technologies which are used to produce electrical circuits by creating electronic devices and systems by depositing inorganic and organic functional inks, usually on plastic and paper substrates. This allows the fabrication of electronic components, such as resistors, capacitors, diodes, or transistors.

The printing techniques are based on the ones used on the graphics industry, and

nearly all industrial printing methods can be used for PE: screen printing, gravure, flexography, inkjet printing, etc. The manufacturing approach is also similar to conventional printing, because the circuits are fabricated on a layer-by-layer basis, depositing them one on top of the other.

Despite the similarities, the requirements for each domain are very different. On conventional printing, the maximum resolution is determined by the human eye, but for electronic device manufacturing higher resolutions are desirable. Specially for transistor devices, whose lower dimensions usually imply higher performances.

From all printing techniques, inkjet printing is postulated as one of the more promising. This technique is a digital, mask-less, low-cost, non-contact deposition method. Being an additive process (only depositing material on specified areas) reduces material waste, and makes inkjet printing a more environmental friendly technique. In addition, its digital mask-less nature makes inkjet printing a great prototyping tool, allowing an easy printing pattern customization.

There are two main approaches on inkjet printing: Drop on Demand (DoD) printers, and Continous Inkjet (CIJ) printers [8]. On CIJ printers, there is a continous stream of ink stimulated to break it in drops, which are deflected as necessary to form the required pattern. DoD printers on the other hand form the drops as needed. DoD printers have typically more resolution, and can eject smaller droplets than CIJ printers, although the later usually are capable of higher printing speeds.

Looking at DoD printers, the ejection mechanism is accomplished mainly by two different technologies: thermal-bubble jets, and piezoelectric jets [9]. Thermal-bubble nozzles achieve ejection by applying an electrical current through a heating element. This causes a vaporization of the ink in the reservoir, making a bubble and a large pressure increase, which ejects a droplet. These nozzles have a low fabrication cost, and are not affected on air trapped on the ink chamber. On the other hand, the ink needs to resist this rapid thermal change, making it unsuitable for the deposition of organic materials or materials with low curing temperatures (the nozzle usually heats up to 400 °C, while most inks have a curing temperature around 150 °C).

Piezoelectric nozzles do not apply a thermal load to the ink in order to eject a droplet and have longer lifespans. Fig. 1.1 contains the basic diagram of a piezoelectric nozzle system. The nozzle ejects the droplets by applying an electrical driving signal to a piezoelectric transducer. The application of a waveform on the transducers creates pressure differences on the nozzle chamber, making it to eject an ink droplet [10]. This ejection methodology is the preferred method for deposition of functional materials for PE, as it does not create thermal stress to the ink. In addition it can eject faster than the thermal nozzles because it does not require the cooling down step of the heating element.

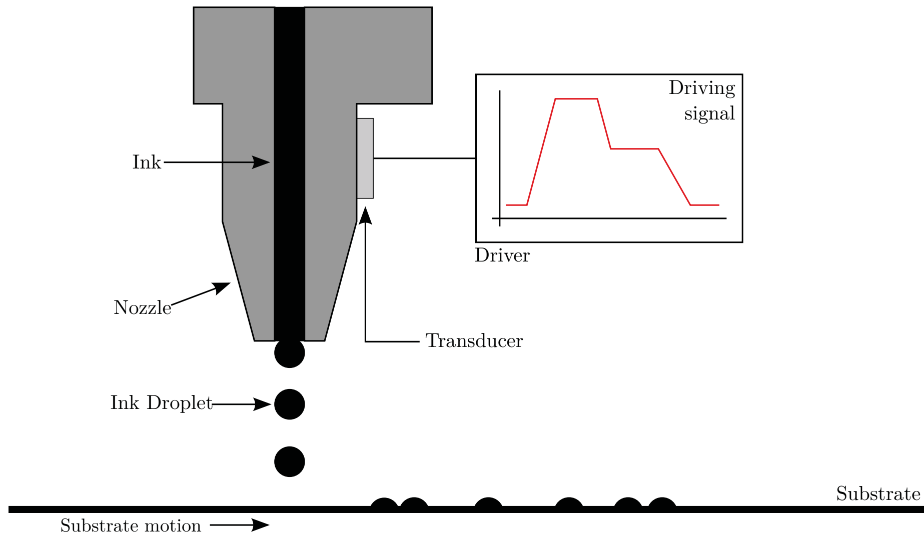


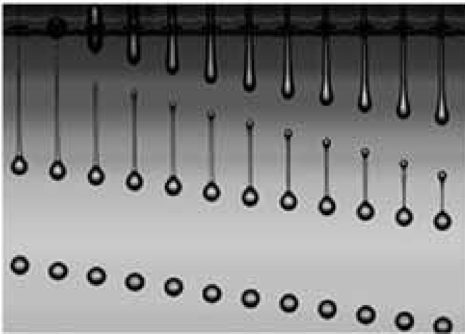
Figure 1.1 | Inkjet printing printhead diagram.

Despite all the aforementioned advantages, inkjet PE still has some shortcomings making the development of reliable devices and circuits difficult. In order to achieve a working printing system, all of the printing head, ink, and substrate must be compatible between each other. From the device fabrication perspective, the ink-substrate interaction is of vital importance to achieve a correct deposited film.

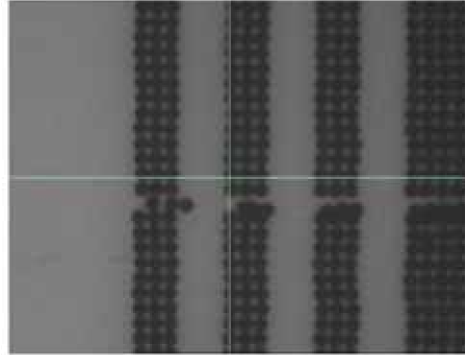
During the printing process several different imperfections may appear. Fig. 1.2 contains some examples of those errors. We can observe that some ejection failures (extra satellite droplets being ejected as on Fig. 1.2a, non-matched ejection velocities on Fig. 1.2b, or a clogged non-ejecting nozzle as in Fig. 1.2c) can produce some printing errors. In addition, some imperfections appear due to ink coalescence. Fig. 1.2d illustrates deformation errors on the intended printed shapes because of this effect.

1.2 State of the art and motivation

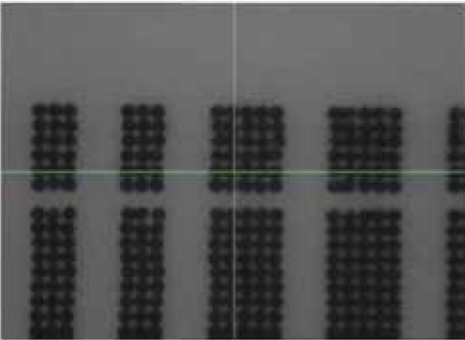
We have seen that inkjet printing is a promising technology which enables a wide new range of applications. Its low cost means a low entry barrier, therefore increasing its adoption. Nevertheless, it is constantly evolving and under heavy development,



(a) Satellite droplets. Source: Mark-Jan van der Meulen. University of Twente.



(b) Non-matched ejection velocity. Source Fujifilm Dimatix Inc.



(c) Non ejecting nozzles. Source Fujifilm Dimatix.

(d) Coalescence effects.

Figure 1.2 Different printing imperfections.

making it less stable for system fabrication. Another consequence of this is that a high knowledge of materials and printing methods is required to design for this technology. In addition, there are still some important printing stability and reliability issues which need attention.

The majority of research areas for inkjet printing involve patterning and material improvements for obtaining better performing devices [11–13], but a lot of less work is focused on circuit and system design [14, 15]. Increasing efforts on this research area and bringing the sillicon-based microelectronics know-how would bridge the gap between circuit design and technology manufacturing, thus reducing the interdependence of these two areas and the needed knowledge to design circuits. These reasonings have their microelectronics counterparts, when C. Mead and L. Conway [16] introduced the concept of the transition from the *tall-skinny* engineers, meaning an engineer capable of all stages of the design process, to the *short-fat* engineers, specialized on each design process step. Adapting this analogy to the status of inkjet PE, we need some clean interfaces between design and manufacturing [17], thus enabling engineers to design without and in-depth knowledge of the underlying technology, and technology developers focus on processes improvements.

In order to accomplish this, we need a set of tools specifically tailored for PE, and introduce the concept of Process/Physical Design Kit (PDK), which will contain the necessary information of PE processes allowing to abstract design from manufacturing. These parts are covered on chapters 2 and 3 respectively.

The idea of adapting current Electronic Design Automation (EDA) tools is not new, as the VLSI group from the University of Minessota proposed an Organic PDK [18]. This PDK based on FreePDK from NCSU [19] is only available to EDA tools supporting Open Access (Cadence, Synopsys, and Mentor Graphics). The main drawback is the high cost of the EDA tools, and that although it includes some information on the used technology, it is still focused to traditional microelectronics processes, so the PDK only contains very basic functions.

Starting on 2009, the Si2 foundation first presented OpenPDK coalition [21], although it did not start presenting specifications until mid-2010. This alliance of EDA vendors proposed an unique design kit specification (OPDK), based on an eXtensible Markup Language (XML) database. Fig. 1.3 shows the general use cases for this PDK. The whole specification defines the basic representation of the necessary information necessary for circuit design (the interchange area on the figure), and the mechanisms used to use this information in the EDA tools (the transclude, parse, and check areas on the figure). Although this is not specific to PE, it could be adapted and extended for it. On the other hand, the main disadvantage is that is still under development (changes could affect the extensions for PE technologies), is not fully public, and still there is not a wide adoption

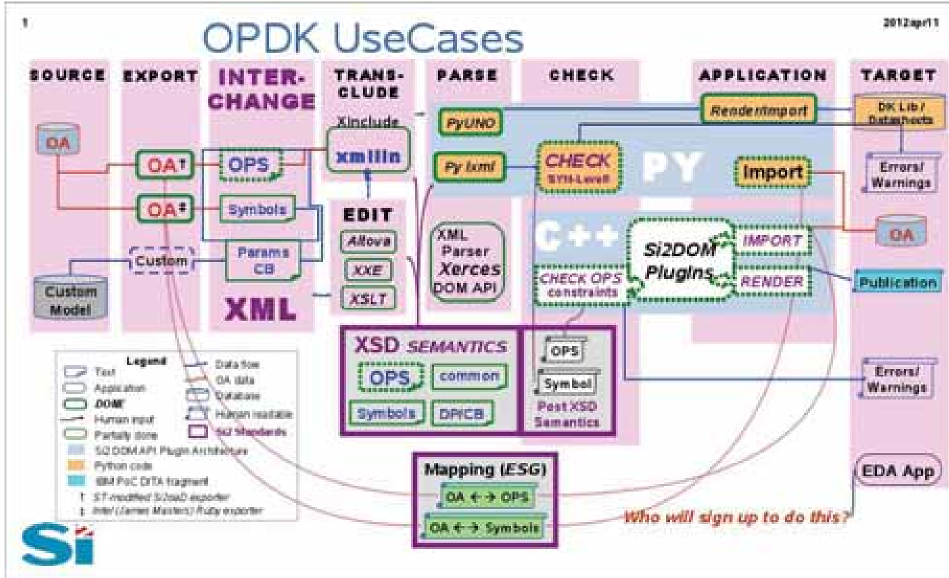


Figure 1.3 | OpenPDK use cases. Image source: [20]

.

across different EDA vendors. In addition, the first proposal was made on 2012, well beyond the beggining of this thesis, so while being interesting to adopt some of their design desitions, it is not suitable to adopt OpenPDK as the PDK for PE.

Concerning printing reliablity we need an in-depth understanding of the underlying printing processes. At the time of this thesis, the main efforts on PE research still remain on device/material fabrication and improvements [22, 23], or in fine tuning processes to reduce printing issues [24–26]. Numerical modelling of printing processes focus more on droplet ejection simulations on the printheads [27, 28], rather than simulating deposited droplets and films behavior [29]. These works usually focus on single droplet modelling, or interaction between two droplets, making difficult adapting these techniques to a full circuit, mainly due to computational costs. Nevertheless, Soltman et al. made an initial approach to modelling [24, 30] full films, extracting from there a set of possible corrections needed to obtain the desired printed output, focusing on an approximation model of deposited lines and films.

This extracted information on the deposited ink behavior can lead to having a post-processing tool, which modifies the design to overcome undesired results and maintain the fidelity with the intended design. In addition, having such a tool

will finally bridge the step going from design to manufacturing. The design of the characterization procedure needed to extract this knowledge and the implementation of this tool are covered on chapters 4, and 5 respectively.

1.3 Thesis context

This work is framed inside two research projects: The Technology and Design Kits for Printed Electronics (TDK4PE) project, European project [31], and the Application Specific Printed Electronic Circuits - Technology Design Kits (ASPEC-TDK) project, funded by Spanish government [32].

One of the main objectives of these projects is to provide a methodology which enables PE circuit implementation, based on (and extending) current silicon microelectronics methodologies. This will achieve a reduction of costs and time to market of circuits based on PE technologies.

Inside these projects, the work of this thesis can be divided in two differentiated parts, one focused on developing and adapting design flows and EDA tools to work with PE, and another focused on providing a characterization methodology for modelling the printing processes while extracting key printing information to develop compensation operations, and the associated tools used to convert designs to something directly manufacturable.

To cover the first part, this work proposes a common database containing all PE information needed to customize existing EDA tools. Then, each different EDA tools will access this database by means of an Application Programming Interface (API), or a set of specific converters. The design of this database has been performed under the TDK4PE project, in collaboration with one of the project partners: PhoeniX software [33].

In order to cover the last part, we need extensive knowledge of the underlying printing procedure, specially the ink behaviour when deposited. Modelling this we can compensate and correct some effects, like ink spreading, line bulging, or shape changes due to coalescense, to increase the printing fidelity and obtain final morphologies resembling as much as possible to the designed ones.

While previous works propose an initial approach to numerically model printed lines [24, 30], and other works focus on substrate-droplet or droplet-droplet simulations [27, 28], this approach is not scalable to full circuits, mainly due to the computational complexity of these techniques. This work proposes a different approach to extract compensation information than the one proposed by Soltman et al. [30]. Building up from the results of different expected printing effects [34, 35],

instead of numerically model the film deposition, we create a semi-automated characterization setup based on both optical and electrical characterization. This allows the mass-characterization of printed samples, so by fabricating a large amount of test samples with different corrections, and by applying different image processing algorithms and statistical procedures, we can evaluate which correction produces a better improvement with statistical confidence, thus minimizing the effects produced by printing issues not caused by the correction itself (e.g. nozzles clogged or non-matching ejection velocities).

1.4 Aims and objectives

This thesis aims to provide a general design flow and kit, specifically tailored for PE. All the needed information for such kit will be backed up by a database, making it accessible to several different EDA tools. In addition, a custom EDA component will be developed to cover the final conversion step to obtain manufacturable designs. This component will perform all necessary Fluidic Effects Correction (FEC) techniques to increase the design fidelity to the fabricated circuits.

To be able to cover these aims, the objectives of this thesis are twofold. The first main objective comprises the design of a suitable design flow for PE, based on current microelectronics design flows. This design flow will cover the steps for basic full-custom circuit design (schematic and layout entry, electrical simulation, and verification tools, such as DRC and Layout Versus Schematic (LVS)). In addition, all relevant technology information needed will be stored in a database, easy to access from different EDA tools. Therefore the detailed steps necessary to achieve it are:

- Construct a suitable database to represent all information related to PE processes, being easily accessible to EDA:
  - Develop the database representation. Use an extensible structured format, while being easily parseable.
  - Define an API to manipulate the data from EDA tools, and create a set of standalone generators for extracting useful information from it. This step allows the connection of distinct EDA tools.
- Customize a full-custom design environment oriented to PE circuit designs:
  - Customize a schematic entry program, containing the available PE devices along their parameters.

  - Integrate specific PE device models into an electrical simulator (SPICE-like). Together with schematic capture will allow easy simulation of PE circuits without requiring full layout and fabrication.
  - Customize a physical layout tool. This tool will act as the core of the design process, allowing the drawing of the final design to manufacture. To ease the design steps, a set of Parameterizable Cells (PCells) will be provided, reducing design time and containing layout pieces which are correct by construction.
  - Create a set of verification procedures to ensure design validity. This includes DRC checks for layout, and a netlist extraction script to perform LVS checks.

The second main objective is the creation of a tape-out tool for PE. This tool will convert designed layouts to files directly understandable by the printer. This conversion will incorporate the FEC rules which will correct the printing issues, and obtain a final fabricated result which reproduces the intended design with more fidelity. To achieve this objective we need to:

- Study and quantify how the ink behaves when deposited, in order to extract which corrections are needed. To accomplish this is necessary to:
  - Build a semi-automated characterization setup to characterize printed structures optically and electrically. A semi-automated setup will allow mass-characterizatinon of test structures, therefore increasing the statistical significance of the extracted results. This allows the reduction of the noise caused by spurious printing defects.
  - Design a set of different test structures and possible compensation operations to extract necessary corrections. The tested structures will be shapes commonly appearing in circuit and devices layouts, thus ensuring that compensation operations also apply to bigger circuits. The proposed compensations are built upon previous results describing expected printing defects.
  - Define a metric which evaluates each different compensation operations so they can be evaluated. This metric scoring the performance of each different compensation is essential for statistical analysis.
  - Implement a set of image processing algorithms, able to analyze each captured image, align and superpose against a test template containing the expected results, and using the previous defined score metric evaluate the performance of the compensation. In addition, this analysis will perform basic image pre-conditioning like removing satellite drops, providing additional printing quality information.

- Using computational geometry algorithms, implement the extracted compenstaion operations on the tape-out EDA component. These algorithms will analyze the design layout, detect areas of interest where compensations should be applied, perform all needed transformations, and generate a set of files understandable by the printing equipment.

Covering all these objectives we will achieve a complete design environment adapted to inkjet PE technologies, and achieving the abstraction between the different steps.

This thesis is outlined as follows: Chapter 2 describes design flows and kits, and introduces the characterization environment. On chapter 3 contains the description of the information database, and how is used to customize two sets of EDA tools. Chapter 4 explains the characterization procedures to analyze the printed shapes and extract suitable compensations, along to the statistical analysis performed, and chapter 5 describes the tape-out tool implementation. Finally, chapter 6 draws the final conclusions and gathers the future work and open ressearch lines.

Design Flows, Tools and Kits | 2

This chapter covers a review of the design kit, design flow and EDA tools concepts, and their application in PE.

First we will see the different abstraction levels from traditional microelectronics technologies, and the status of PE. Then, we will review the different design flows available. Last, we will cover the characterization setup and how the resulting technology information will feed the design kits for their further EDA tools customization within the design flow.

2.1 Abstraction levels and design flows

There has been an increase of the level of abstraction during the evolution of the EDA tools for traditional silicon-based microelectronics. This raise of the abstraction has followed the development of the technology, and the increment of the complexity of the designs [36, 37]. We can see this evolution on Fig. 2.1.

Following the technology maturity evolution, the first abstraction level kept on working at transistor level design with electrical simulation. The designs were captured at layout or schematic level, and simulations performed with SPICE-like simulators [38, 39] with basic device models and capacity loads for timing analysis. As technology and design complexity increases, the abstraction level has to increase too. Therefore, the abstraction contemplates the concept of cells, as a cluster of transistors with a certain functionality, and introduces logical gate-level models and behavioral simulators. As technology evolves, gates are clustered into cells, and cells into Intellectual Property (IP) blocks; and design entry is shifted to hardware

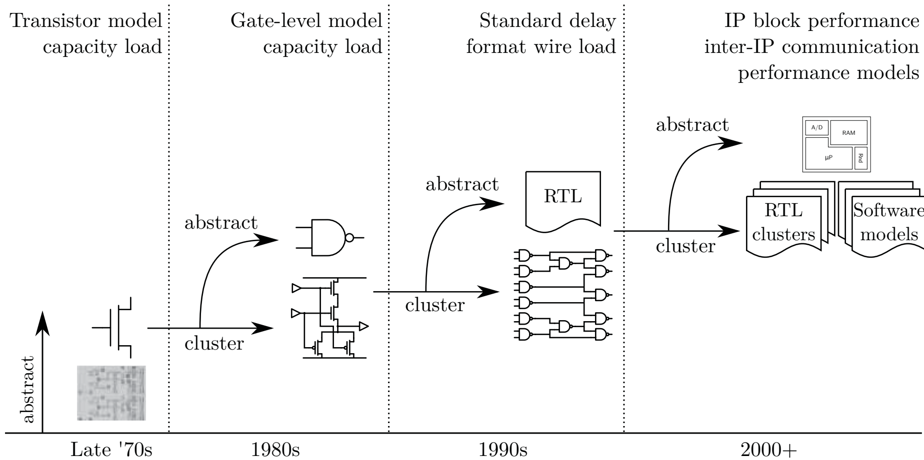


Figure 2.1 | Evolution of technology and design abstraction levels. Image credit to [36].

description languages.

We see that there is a lot of difference between designing at transistor level and producing a full-custom layout, or use a Hardware Description Language (HDL) language to produce a high level description of the behavior of a system. Independently of the design entry point, for each abstraction level there are a set of necessary steps to produce a system from design to manufacturing. These steps are defined by a design flow.

Therefore, a design flow is composed by a combination of a set of EDA tools with a methodology used to get a circuit design ready for fabrication.

In the case of PE, the technology and its automation necessities share many points with traditional microelectronics, so we can take advantage of the expertise and developments in this field and adapt them to the peculiarities of those emerging technologies. But unlike modern silicon-based technologies, PE processes are still on early stages of development, hence we should focus on transistor-based abstraction levels (i.e. electrical and layout design and simulation activities), and therefore use and adapt full-custom design flows.

2.2 PE design flows

Taking current microelectronics design flows as an starting point, and considering the maturity of PE technologies, we can consider adapting two different approaches: a Printed Circuit Board (PCB) based design flow, or a full-custom Application Specific Integrated Circuit (ASIC) design flow.

PCB design flows are based on schematic capture, using existing libraries of discrete devices, and their associate footprints to produce the board layouts. For this technology, the development and prototyping costs are cheap compared to silicon-based circuits, therefore the main effort leans towards prototyping and characterization, although there are logical and electrical simulators. In addition, these flows have to include DRC and Electrical Rule Check (ERC) tools to verify the final layouts. Due to prototyping costs, these processes are less critical than their ASIC counterparts.

Full-custom ASIC design flows are also based on schematic capture, using basic device libraries and the associated electrical models. Contrary to the PCB flows, these are strongly based on simulation, because the prototyping costs are much higher. The next step after capturing the schematic and verifying the simulations is producing a geometric layout, which the fabrication masks will derive from. This layout is checked using DRC, ERC, and LVS tools, to ensure that it works as intended, and it corresponds to the previously captured schematic.

After looking at the available options, we ended in a compromise between the two flows. We can see the proposal on Fig. 2.2. At first glance, the proposed flow resembles the ASIC design flow, although due to the maturity and prototyping cost of PE technologies, there is less effort on performing verification on design level rather than characterizing prototypes. This implies that although the proposed design flow will cover DRC, LVS checks, and electrical simulations, they will not be as thorough as they are on ASIC design flows. On Fig. 2.2 we can also see the concept of PDK, which will contain all the needed technology information for the design flow.

PE is a very different technology than microelectronics, hence the post-layout tape-out procedures will be different, although needed in both cases. For silicon-based microelectronics, these steps are the responsible of mask creation from the design layout, and often include a set of correction steps, such as Optical Proximity Correction (OPC) and Resolution Enhancement Techniques (RET) techniques, so the fabricated design resembles as much as possible to the intended one [40, chap. 19], or Inverse Lithography Technology (ILT), which determines mask shapes that produce specific results [41]. In addition, those corrections help to the manufacturability of the design and increase the yield. PE fabrication has similar imperfections when printing, albeit caused by very different physical effects.

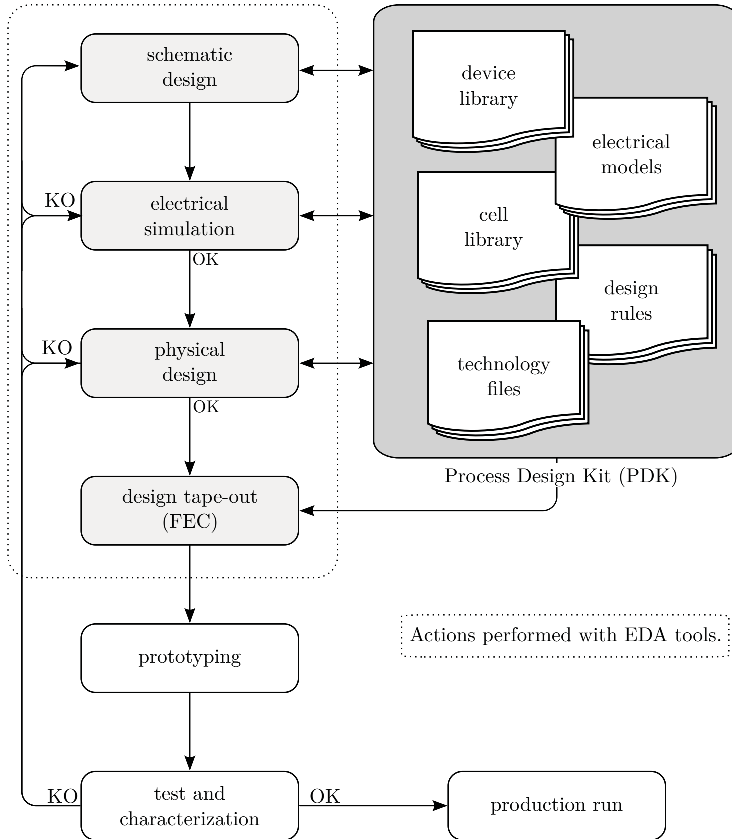


Figure 2.2 | Diagram of the proposed design flow for PE.

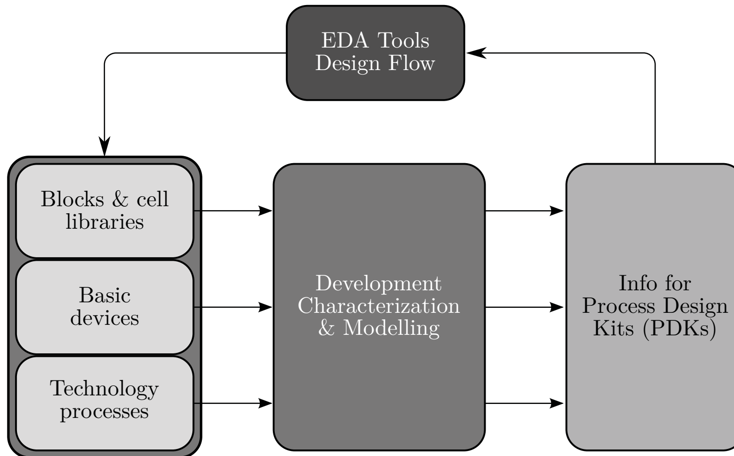


Figure 2.3 PDK development and characterization, and EDA customization loop.

Therefore we need a last step on the flow which will convert the design to a set of printable files. This conversion step includes the necessary FEC corrections, which act as the RET counterparts for PE. The FEC techniques and the tool performing the tape-out procedure are covered in detail on chapters 4, and 5.

2.3 Process Design Kits

Process/Physical Design Kits (PDKs) consist of a set of files containing all the needed information to customize a specific Computer Aided Design (CAD) design flow enabling design of electronic systems for a particular technology. A representation appears on the right shaded area of Fig. 2.2. Therefore, it is the design flow which dictates which information is necessary on the PDK.

Fig. 2.3 represents the process needed to create a PDK. In traditional silicon-based microelectronics, the PDK information is extracted using characterization and modelling methodologies. This information is then packaged into a PDK which will feed a set of tools allowing the design for this technology. On the other hand, because PE processes are still in heavy development and there is much less knowledge on the processes, we need another approach: evolving the PDK with technology developments by performing incremental characterization.

This evolutive approach allows having a very basic PDK early, therefore being able to customize a set of EDA tools. Taking advantage of the automation facilities of these tools, we can use them to assist on test vehicle and generating large amount of test structures, improving the technology characterization and PDK refinement. Hence we will start by basic characterization of print processes and go up to basic devices and cells modelling, generating an updated PDK at each iteration. To be able to cope with the characterization of large amount of samples we need a semi-automatic characterization setup, described on section 2.4.

2.3.1 Necessary process information

Depending on the design flow step, the tool covering it will need some technology related information. This section will outline which information is needed on the different steps of the design flow.

Schematic design and Electrical simulation

Because it is a netlist description specifying the connections of basic devices, schematic entry itself is not a technology dependent step. Therefore, schematic capture programs need a basic library of all the available devices on a particular technology. Each device is comprised of the device symbol with the related customizable design parameters (p.e. W/L values for transistors), and a reference to its model.

In order to perform electrical simulation, the design kit needs access to the electrical models for the available devices. Therefore, for each device in the library, the design kit has to include its electrical model.

Physical design

To be able to represent physical layouts, the EDA tools need to know which layers are available in the technology. In addition to their graphical properties, the layout editor needs to be aware of connectivity information, so it can extract devices and nets to build a netlist file, as well as the extracted DRC rules used for verification purposes.

Design tape-out and FEC tools

These tools are the most dependent on technology processes of the design flow. In order to perform the design conversion and correction, they need to know specific parameters for each process step, either the physical parameters or some appropriate empirical abstractions.

2.3.2 XML interface

As we have seen, PDKs collect all the information potentially needed to customize all the EDA tools covering a design flow, hence allowing their usage to design circuits for the target technology. This implies that every change in the technology produces changes on every affected part of the PDK. Having unstable technology processes means that the PDK creation and/or update process has to be performed regularly, or even frequently until the technology is mature enough and its parameters well characterized.

To facilitate the PDK generation, we propose a common database for all the technology related information. Hence, when technology updates, we only have to change this intermediate representation, which will feed all different PDKs and EDA tools customization files. Fig. 2.4 contains a representation of this strategy.

We will use XML as this intermediate representation, and then adapt the EDA tools or create a data translator which will generate the needed customization. Chapter 3 discusses this solution in depth.

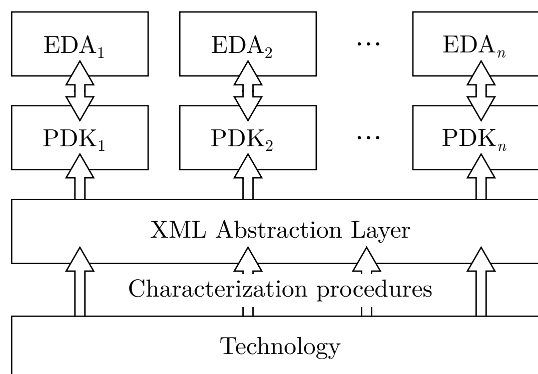


Figure 2.4 | PDK diagram with an abstraction layer.

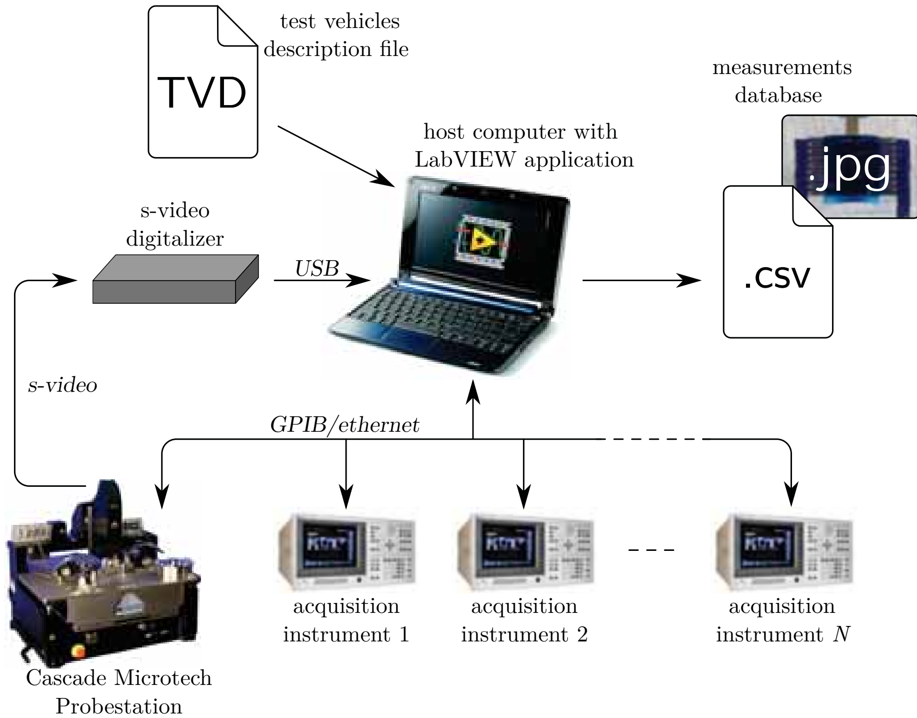


Figure 2.5 Characterization environment setup. The diagram contemplates both optical and electrical characterization.

2.4 Characterization strategies and setup

As commented, in order to extract a PDK for a PE technology following the procedure outlined on Fig. 2.3, we need to automate the characterization step to avoid it being the bottleneck of technology evolution. This progressive refinement and the maturity of the technology means that a large amount of samples will be characterized, therefore needing a semi-automatic procedure able to cope with the samples volume.

The characterization setup is built around a host computer controlling Cascade Microtech Summit Series 12000 probe station in combination with all the required measurement instruments, and is shown on Fig. 2.5.

This probe station has been adapted to work with flexible plastic substrates, and is

equipped with a Leica S6D microscope and a video capture device. Using a set of LabVIEW 2013 programs we can control all the setup equipment, thus being able to configure, control the actions, query the status, and exchange data. This allows smart characterization protocols which performs different actions depending on the measured data and capture still images of the devices under test.

The whole characterization procedure starts with the Test Vehicle Description (TVD) file, which is automatically generated using the EDA tools scripting facilities. This file contains all the relative position of the samples under test, and each sample parameters and identification for tracking and posterior analysis purposes. With this information, the LabVIEW procedures on the host computer drive the probe station movements and all connected equipments, performing the corresponding measurements, and capturing an image.

After the characterization of the foil finishes, the host computer generates and exports a Comma-Separated Values (CSV) file with the results, and saves it with the captured images.

2.5 Conclusions

In a general sense, PE processes share a lot of characteristics with their microelectronics counterparts. Therefore, it makes sense to adapt existing microelectronics design flows to PE technologies, hence taking advantage of the expertise of this area. But there are particular needs that those already established flows do not cover, hence requiring some specific in-depth adaptations.

Because of the still low maturity of the technology, all the processes are constantly evolving, so all the information included inside the PDK is prone to change. To be able to cope with this evolution, we need some representation that will help changing it without much effort, hopefully decreasing the necessary work, thus reducing possible error. We developed an specific XML database representation specifically tailored to printing PE technologies to collect and exploit PDK information.

This XML intermediate representation, in addition to concentrating all the information in a common place, facilitates PDK generation for several different EDA tools, therefore obtaining a twofold advantage. First, once a tool supports this representation, any technology change will be reflected automatically on the PDK, and secondly, by having a complete representation it is possible to describe a completely new technology and generate all the PDKs automatically, without making any change to the supported EDA tools. In addition, each time the parameters change, just updating the XML information and all the PDKs will be automatically up to date.

The semiautomatic characterization system developed at ICAS group provides an automated probe station and equipment controlling, having tracking information (i.e. position and parameters) for each tested sample, and automatic data data extraction and analysis. The whole setup is driven by a set of LabVIEW control subroutines which read the generated TVD file, therefore providing an automated path from test vehicle generation to characterization, thus making a less error prone system reducing human interaction, and speeding up technology characterization.

The proposed intermediate representation and its usage is discussed thoroughly on chapter 3, together with two use cases for two different set of tools and different technologies. Chapter 5 describes the framework used to perform the FEC and conversion for design post-processing.

Specific Design Kits and EDA tools for PE | 3

This chapter covers the specific details of the generated PDKs for PE. First of all, the chapter contains a revision of the EDA tools used to cover the design flow, leading to the discussion of the design kit intermediate representation and its usage.

As a concluding section, we will show two different technologies adapted to this set of tools: an inkjet process, and a gravure process.

3.1 EDA tools selection

On this section we will discuss which set of tools will be used to cover the proposed design flow. First we will discuss a set of free (and if possible open source) tool set, and then a commercial tool package.

The main characteristics we search for are that the tool is easy to install, customize, and maintain, while being cross-platform software. In addition, the layout editor, and associated tools, should support arbitrary geometries and contain basic verification tools such as DRC, netlist extraction, and LVS.

All the tools should be able to import and export to different file formats, following the industry standards, and if possible, contain scripting facilities to ease the customization steps, allow procedural layout development using PCells, and simplify tool linking and sistematic steps.

A part from the set of EDA tools evaluated, the design flow will be mapped on top an available commercial EDA package already fixed in the framework of the TDK4PE European project.

3.1.1 FOSS tool set

This section contains the discussion of the Free/Open Source Software (FOSS) tool set. In order to cover the flow, we need a layout editor, a schematics editor and an electrical simulator.

Along this chapter, the usage of FOSS will not be precise because it also will include some tools which are free (as in gratis) but not open source.

Layout editor

The first tested layout editor has been *Toped* [42]. *Toped* is a multi-platform layout editor, supporting GDS, OASIS, and CIF. It is an open source software which is still under development. As major advantages, the editor is free and supports standard input/output file formats. In addition, it can import *Cadence*® technology files. On the other hand, the editor does not come with its own verification tools, although it can import *Calibre* errors, and the PCell support is limited.

Glade IC [43] is a free (not open source) layout editor able to read and write a wide range of file formats. The program is scriptable using *Python* and is under active development. In addition, it has basic DRC, extractor, and LVS tools (this last one using *Gemini* [44]), and supports the usage of PCells. A part from its own technology file format it can import several other EDA formats such as *Cadence*® technology files.

Electric VLSI [45] is a schematic and layout editor, free and open source, and multi-platform. It is under active development and also supports a wide range of input/output file formats. It contains verification tools, such as DRC, and ERC. The design entry is very different from traditional EDA tools, because is strongly based on connectivity. This makes LVS checks immediate, as both schematic and layouts are captured placing nodes (the components themselves) and arcs (the connectors or wires), therefore have connectivity information. As a drawback, this encumbers designers when they want to draw arbitrary geometry shapes, and can lead to inconsistent results, as touching shapes are not necessary actually connected.

Magic VLSI [46] is a venerable layout editor quite popular in universities and small companies. It is free and open source, and under active development. It is capable of importing and exporting to standard file formats and has several verification tools for LVS and DRC. On the other hand, it is a complex and unintuitive software (operated using commands), and it only supports "Manhattan geometries". In addition, *Magic* is not a cross-platform program (without using an emulator).

Table 3.1 contains a summary of the reviewed characteristics for each layout editor

	Toped	*Glade*	*Electric*	*Magic*
Support reading multiple input output design formats (at least GDS).	✓	✓	✓	✓
Support Cadence® technology files.	✗	✓	✗	✗
Arbitrary geometry and drawing support.	✓	✓	✓[1]	✗
Easily scriptable (for customization).	✓[2]	✓	✓	✓
Creation of PCells.	✓[2]	✓	✓	✓
DRC verification.	✗[3]	✓	✓	✓[3]
LVS verification.	✗	✓	✓	✓[4]
Netlist extraction.	✗	✓	✓	✓[4]
Cross platform support.	✓	✓	✓	✗
Ease of use (and similarity to other VLSI layout editors).	✓	✓	✗	✗
Opensource software	✓	✗	✓	✓

[1]While allowing drawing arbitrary shapes, it is based on connectivity, not on arbitrary shapes drawing.

[2]Using a not widely used, proprietary language.

[3]Can import DRC results from *Calibre*.

[4]Only for manhattan geometries.

Table 3.1 Summary of the main characteristics for each layout editor evaluated.

	XCircuit	*KiCad*	*gschem*
Scriptable for inter-tool communication.	✓	✗	✓
Standalone program (not part of other non-suitable EDA suite.	✓	✗	✓
Custom symbol with custom attributes.	✓	✓	✓
SPICE simulator connection/netlist extraction.	✓	✓	✓
Cross platform support.	✗	✓	✓

Table 3.2 Summary of the main characteristics for each schematic capture program evaluated.

program. After the review, the most suitable tool seems to be *Glade IC*. Although it is not open source, its API provides access to all the inner parts, thus making a very extensible program. It is also extended using *Python*, which is a popular language, aiding with the customization and integration with other tools.

Schematic editor

XCircuit is the schematics capture part by the same author as *Magic VLSI* [47]. Although it has a small library of standard symbols and can connect with electrical simulators, it is thought more as a general purpose drawing program. The usage is similar as *Magic*, therefore it is not very intuitive nor simple to use. The main issue is the same as *Magic*, this software is not cross-platform.

KiCad [48] is an EDA suite for schematics capture and PCB board design. It is an extensible, cross-platform solution, although is highly oriented to a PCB flow.

gEDA project [49], as does *KiCad*, provides a set of open source, cross-platform EDA tools. They also provide a full PCB flow, with schematics capture and simulation, but unlike *KiCad*, all the different tools are loosely coupled, thus facilitating using only part of the suite. All the different tools are built around a *scheme* API, therefore the whole set of EDA components are fully customizable.

Table 3.2 contains the summary of the evaluated characteristics for the shcematic capture program. We choose *gschem* and *gnetlist* components of the *gEDA* suite because they are less coupled than its *KiCad* counterparts, maintaining similar capabilities. All the *gEDA* suite is built around the *gEDA* library which is totally

	ngspice	*gnucap*
SPICE like simulator.	✓	✓
Support of custom models using Verilog-A.	✓	✗[1]
Industrial strength simulator.	✓	✗

[1] Although custom models are supported, at the time of the evaluation the only option was to use the model compiler and not Verilog-A.

Table 3.3 Summary of the main characteristics for each electrical simulator program evaluated.

accessible by the scripting API. This facilitates the extension, adding extra functionalities, and connecting the tool with the other ones which compose the design flow.

Electrical Simulator

PE produced devices usually are not modelled using traditional device models coming from microelectronic technologies. Hence, the electrical simulator has to support some way of custom model inclusion. The industry standard for coding device models is to use Verilog-A [50], along with the proposed extensions for device modelling [51].

Ngspice [52, 53] is an open source port of the traditional Berkeley SPICE3f5 engine. In addition to implementing all SPICE functionality, it also integrates XSPICE extensions from Georgia Tech [54] for mixed signal simulation, and basic Verilog-A model loading through *ADMS* [55], which parses a subset of Verilog-A standard and transforms to C code following SPICE simulator interfaces.

Gnucap [56, 57], is the GNU Circuit Analysis package, consisting in a general purpose circuit simulator. It is not based on SPICE, although some models have been ported from their SPICE implementations. The engine is designed to be truly mixed mode, supporting event driven simulations. At the time of this evaluation, custom models had to be implemented using its model compiler [58], and it did not support loading Verilog-A compact models.

Therefore, we chose the ngspice simulator, because it is a industrial strength implementation of the SPICE simulator, with several standard models compiled in, and the ability to integrate custom models using Verilog-A with *ADMS*.

3.1.2 Commercial tools

The commercial package covering the design flow is *MaskEngineer* together with *CleWin*, from PhoeniX^{TM} software. *MaskEngineer* is a professional, object oriented, and parametric layout package. This tool expects that all layouts are described parametrically using the provided scripting language (PhoeniX script). Describing it in a mathematical way increases design reuse, and just by changing parameters, *MaskEngineer* generates the layout. *CleWin* is a hierarchical layout editor containing basic drawing primitives and a scripting engine.

MaskEngineeer engine contains a boolean package supporting geometric manipulation operations, and a built-in DRC tool.

This tool set covers the layout and verification steps. To cover the schematic capture, *CleWin* has been extended to be more integrated to the common PDAFlow API [59]. This integration allows direct access to the design kit components, and is capable of placing an icon view of the component. Effectively this provides schematic capture capabilities to this layout editor, with the advantage of being able to generate the layout directly from the same software.

As for electrical simulator, the commercial EDA package is capable of exporting to SPICE format, therefore *ngspice* is also used.

3.2 EDA tool customizations

After the discussion, we can see the whole design flow and the tools that cover it on Fig. 3.1. The figure contains the proposed set of FOSS tools covering the flow, and an existing commercial package. This integration will demonstrate further the extensibility of the proposed design kit structure.

3.2.1 Free/Open source tool selection

After the evaluation of the FOSS tools, and concerning all options available on 2010 (the time when the evaluation and analysis was performed) we choose to cover the whole flow with:

- *Glade IC* (and integrated tools) for layout, DRC, LVS, and extraction.
- *gschem* and *gnetlist* from *gEDA* for schematics capture and netlist extraction.
- *ngspice* for electrical simulation.

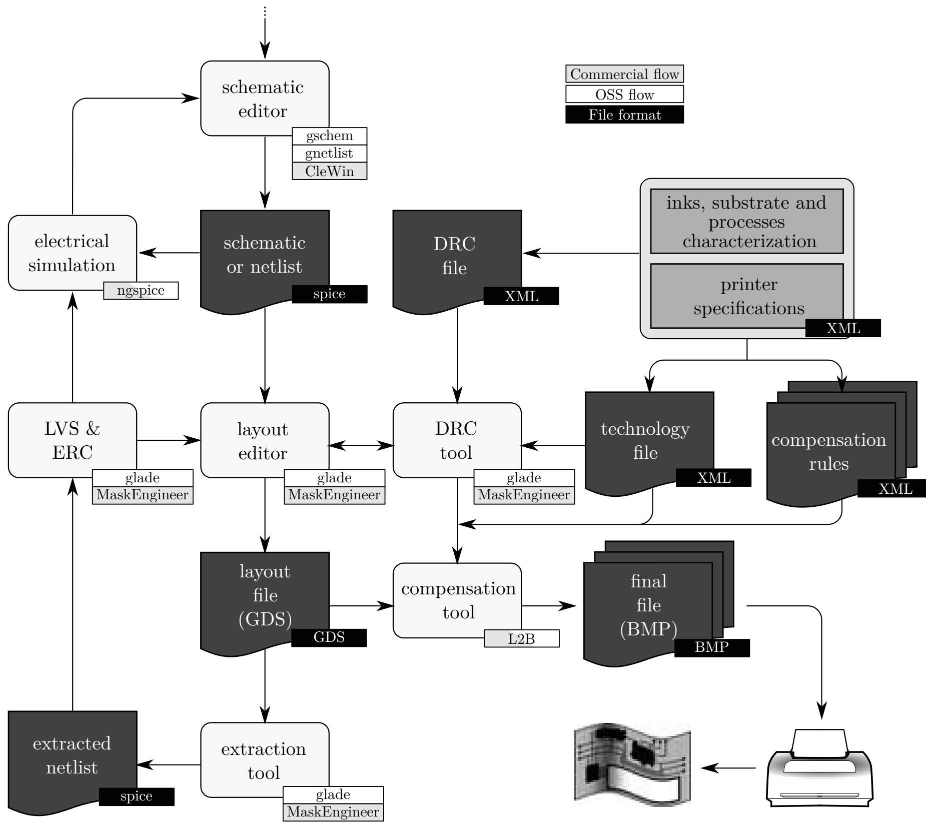


Figure 3.1 Complete design flow, containing all used tools and intermediate formats. The diagram contains both the FOSS and the commercial flow. This diagram can be seen as a detailed version of Fig. 2.2

This set of tools covers all steps needed to produce a design from schematic capture to layout, but there are still some missing features that need an in-depth attention.

The first one is the model inclusion inside the electrical simulator. PE transistors are usually modelled using UMEM [60, 61] into the Unified Compact Model. This model is not integrated into the simulators. In order to include the model inside the simulator we performed the following steps:

- Modify *ADMS* source code support inside the *ngspice* simulator to support additional Verilog-A functionalities: support for *analog functions* constructs.
- According to *ngspice* manual [53, chap. 13], modify device parsing and loading code to support the custom Verilog-A model.
- Modification of transistor evaluation code to support 3-terminal devices (transistors without bulk).
- During the evolution of this work, more devices other than transistors have been modelled using Verilog-A. Therefore, we extended the simulator resistor and capacitor devices model loading code to support Verilog-A models.

In addition to this work, the contact with *ngspice* developers for support, and later for code patches and modifications led to contributions on the simulator main source code tree.

The schematic editor and netlister needed some adaptations to include the device symbols and their model references, and customizations to automatically launch simulations, read simulation parameters from designs, automatically include device models, and support of hierarchical designs and dependency tracking between parts. Hence, the performed modifications to the *gEDA* suite are:

- A complete new set of symbols for the available devices.
- Scripting procedures to adapt gschem to the proposed design flow:
  - Fixed and updated an auto-numbering script to facilitate schematic capture. Until then all devices were not automatically numbered, leading to errors on netlisting procedures.
  - Custom procedures to automatically create a symbol for the current schematic.

- A new netlisting backend[1] which supports hierarchy (multiple schematics, and automatic detection of input/output ports), inclusion of electrical models, and simulation directives.
- An integrated project structure with a central *Makefile* controlling schematic actions:
  - Automatic dependency tracking between a hierarchical design (regenerate affected netlists when a schematic changes).
  - Automatically launch simulations for the top design.
  - Automatically create a suitable netlist for LVS comparison.

For the layout editor, a part from all the PCell creation and verification scripts, we made some customizations to support the design flow and project structure. It automatically loads the design kit libraries, loads the current design (if any), and has custom menu entries to automatically export the design and convert it to printable files. Therefore, the modification done to *Glade IC* are:

- Custom connectivity extraction script supporting 3-terminal transistors.
- Integration with project directory: At startup automatically detect which design kit is loaded and if there is an existing project.
- Custom menu actions to integrate with the design flow:
  - Flatten current design and load it to *Layout2Bitmap*
  - Initial design post-processing before tape-out: Generate derived layers for manufacturing.
- Although the source code to *Glade IC* is not available, the LVS source is. We adapted the program to compare 3-terminal devices.

The last step of the proposed flow on Fig. 3.1 is the equivalent of design tape-out for PE, the creation of the bitmap files which will be sent to the printer. There are not any existing tool which will perform this procedures because they are technology specific. For this purpose, we developed a custom EDA framework (*Layout2Bitmap*) responsible to convert the designs to the appropriate formats, and apply the necessary corrections, transformations, and compensations. This tool is detailed in chapter 5.

[1] The netlister program treats backends as programs which transform an input schematic to a certain netlist program. This provides a highly flexible approach, being able to generate different netlist formats, and hence supporting several simulators.

3.2.2 Commercial package

All the commercial tools are built around the PDAFlow API [59], so by adapting the XML intermediate representation of the design kit to this API we will achieve integration into both *MaskEngineer* and *CleWin*.

MaskEngineer software was intended mainly to photonics applications and it did not have a direct connection with an electrical simulator. Also, it was not possible to seamlessly connect the *MaskEngineer* software with the custom EDA framework used to perform the tape-out procedure. To solve these topics, and during a research stage in PhoeniX Software, we extended both *MaskEngineer* and *Layout2Bitmap* to manage this seamless integration:

- *Layout2Bitmap* plugin code to read design from PDAFlow API designs.
- Connecting code between PDAFlow API and the XML database described on section 3.3 below.
- New *MaskEngineer* functions to export designs:
  - Export as SPICE netlist and launch simulator.
  - Export design to *Layout2Bitmap*.
- Initial implementation and support for extra DRC checks.

3.3 XML database

Now we will address the XML database developed in order to collect all the technology information needed for its usage as a PDK for specific tools. We choose this specific format because it gives a structured and human readable representation of information, while being easy to parse by a computer. In addition, it permits easy querying and subsetting, therefore facilitating information retrieval or generation. During the XML description, we will outline which information is needed on each step of the design flow, and how it is handled by the database. This will lead to a suitable abstraction which achieves tool independence.

3.3.1 XML representation

The specific structure of the XML database will be divided into several different parts describing the details of: layer definitions, design rules, compensation rules,

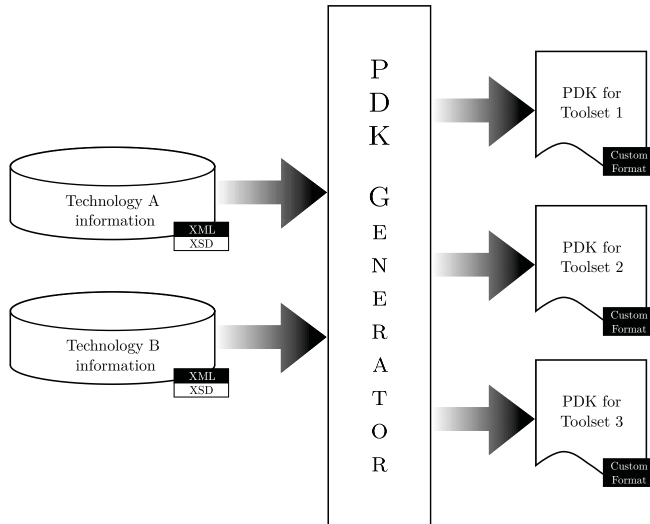


Figure 3.2 | General information and manipulation strategy. For each part, the proposed file formats are shown.

and electrical models information. Fig. 3.2 shows the general strategy. The whole system operation consists on a wrapper on top of the XML database files. This wrapper acts as an abstraction layer, and generates specific configuration files for each target tools.

Along this section we will see how the different needed information for the design kit will be represented inside the XML database.

Layer related information

Before actually representing layers using XML, we need to define the needed information. For each layer, a layout EDA tool needs:

- Layer graphical properties.
  - Layer line and fill color.
  - Layer line and fill stipple pattern.
- Layer name and purpose.

- GDS layer number and datatype, for design import/export.
- Layer electrical properties (layer conductivity and capacity).

All of these properties cover all the information needed to support basic layout design using a layout EDA tool.

The extract of XML Schema Definition (XSD) schema for a layer appears in listing 3.1:

```
...
<xs:complexType  name="dlayer">
  <xs:sequence>
    <xs:element name="parameters" type="parameters"/>
...
  </xs:sequence>
  <xs:attribute name="name" use="required" type="xs:NCName"/>
</xs:complexType>
...
<xs:complexType name="parameters">
  <xs:sequence>
    <xs:element name="short_name"      type="xs:NCName" />
    <xs:element name="purpose"         type="xs:NCName" />
    <xs:element name="gds_num"         type="xs:short"/>
    <xs:element name="gds_dtyp"        type="xs:short"/>
    <xs:element name="print_direction" type="xs:string"/>
    <xs:element name="color"           type="color"/>
    <xs:element name="sel"             type="xs:boolean"/>
    <xs:element name="vis"             type="xs:boolean"/>
...
  </xs:sequence>
</xs:complexType>
...
```

Listing 3.1 | Excerpt of the layer XSD definition.

And an example layer defined following that schema is shown on listing 3.2:

```
...
<dlayers>
  <dlayer name="OrgSC1">
    <parameters>
      <short_name>OSC1</short_name>
      <purpose>drawing</purpose>
      <gds_num>60</gds_num>
      <gds_dtyp>50</gds_dtyp>
      <print_direction>Top-Down</print_direction>
      <grid>20</grid>
```

```
      <color name="" R="0" G="255" B="0" />
      <sel>true</sel>
      <vis>true</vis>
      <fillstyle>
        <name>Slash1</name>
        <bitmapFile>slash1.bmp</bitmapFile>
      </fillstyle>
      <linestyle>Solid</linestyle>
      <function>-</function>
      <printing_layer>1</printing_layer>
      <design_layer>1</design_layer>
    </parameters>
  </dlayer>
...
</dlayers>
```

Listing 3.2 | Excerpt of the layer XML definition

Design Rules information

The case of the DRC rules is more complex than the layer characteristics. To have a complete definition of all available checks, we need an extensible operation definition. XML format allows nested recursive statements, thus is possible to define a type as shown on listing 3.3:

```
...
<xs:complexType name="dlayer_ref">
    <xs:attribute name="gds_num"   type="xs:short"/>
    <xs:attribute name="gds_dtyp"  type="xs:short"/>
</xs:complexType>
...
<xs:complexType name="operation">
  <xs:choice minOccurs="1" maxOccurs="unbounded">
    <xs:element name="operation" type="operation"/>
    <xs:element name="layer"     type="dlayer_ref"/>
  </xs:choice>
  <xs:attribute name="name"  use="required" type="xs:string"/>
  <xs:attribute name="value"                type="xs:double"/>
</xs:complexType>
```

Listing 3.3 | Excerpt of the operation defintion schema.

Having this recursive definition, we can define complex geometrical layout operations acting on single layers or any combination of them. Listing 3.4 shows the implementation of eq. (3.1).

$$(\mathrm{Ly}(60,50) \cap \mathrm{Ly}(70,40)) \cup \mathrm{Ly}(80,30), \tag{3.1}$$

where $\mathrm{Ly}(X, Y)$ specifies a layer selection function selecting the layer with Graphic Database System (GDS) number X and datatype Y. By using this structure, we can refer to a layer or any combination of layers of the design. A part from the DRC rules, the same definition is shared among the whole database sections which need querying layers.

```
<operation name="OR">
  <operation name="AND">
    <layer gds_num="60" gds_dtyp="50" />
    <layer gds_num="70" gds_dtyp="40" />
  </operation>
  <layer gds_num="80" gds_dtyp="30" />
</operation>
```

Listing 3.4 | Example of operation definition in the XML database.

After having this recursive definition of the operations, we can define the proper DRC rules. Listing 3.5 contains an extract of the schema definition:

```
<xs:complexType name="interLayerRule">
  <xs:sequence>
    <xs:element name="name"          type="xs:NCName"/>
    <xs:element name="layerA"        type="layerSelector"/>
    <xs:element name="drcOperation"  type="drcOperation"/>
    <xs:element name="layerB"        type="layerSelector"/>
    <xs:element name="value"         type="xs:double"/>
    <xs:element name="description"   type="xs:string"/>
    <xs:element name="error_msg"     type="xs:string" />
  </xs:sequence>
</xs:complexType>
```

Listing 3.5 | DRC rule definition schema.

And an example rule is defined as shown on listing 3.6:

```
...
<rule>
  <name>GtC1.0</name>
  <layerA>
    <operation name="AND">
      <layer gds_num="100" gds_dtyp="10" />
      <layer gds_num="80" gds_dtyp="40" />
    </operation>
  </layerA>
  <drcOperation>overlap</drcOperation>
  <layerB><layer gds_num="100" gds_dtyp="10" /></layerB>
  <value>25</value>
```

```
  <description>Minimum overlap between gate region and drain/source</description>
  <error_msg>Gate to drain/source overlap is less than 25</error_msg>
</rule>
...
```

Listing 3.6 | DRC rule definition rule definition.

In addition, we need some common mapping between the geometrical operations defined on the XML database and those available in most DRC tools. The following operations act as the base where all the checks will be built upon. They are divided in four categories: boolean operations, region manipulation functions, geometric tests, and edge selectors:

Boolean operations contain the union ($\cup$), intersection ($\cap$), and difference ($\setminus$) operations.

Region manipulation include the growing ($\oplus$) and shrinking ($\ominus$) operations.

Geometric tests are the concavity test (ConcV), and distance function (d).

Edge selectors return the edges facing the outside (e_o), the inside (e_i), enclosed edge (e_e), or abutting (e_b) a given region.

Using these operations then, we can define a fundamental set of rules. Fig. 3.3 shows this basic set. When adapting the XML database we will always use the operations available on the target DRC tool, but in case an operation is not available, we can define it as equations (3.2) show.

$$\text{Width}(A) = d(e_i(A), e_i(A)) \geq v \tag{3.2a}$$

$$\text{Notch}(A) = d(e_o(A), e_o(A)) \geq v \tag{3.2b}$$

$$\text{Spacing}(A, B) = d(e_o(A), e_o(B)) \geq v \tag{3.2c}$$

$$\text{Margin}(A, B) = d(e_i(B), e_o(A \cap B)) \geq v \tag{3.2d}$$

$$\text{Overlap}(A, B) = d(e_b(B, A \cup B)), e_e(A \cup B)) \geq v \tag{3.2e}$$

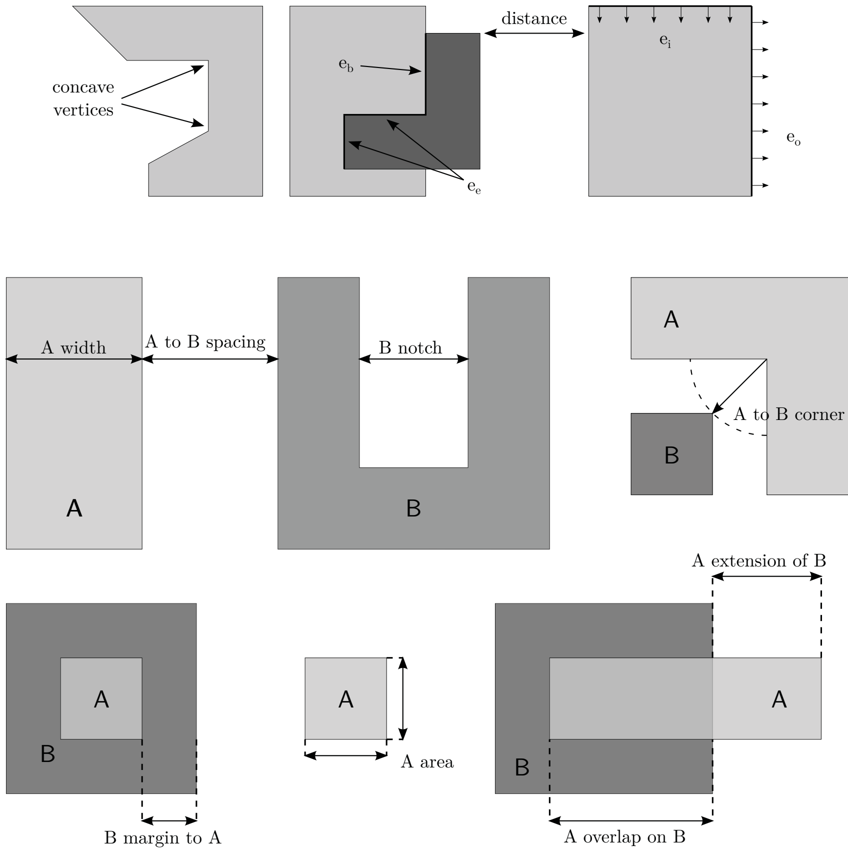


Figure 3.3 Fundamental set of design rules and basic manipulation operations.

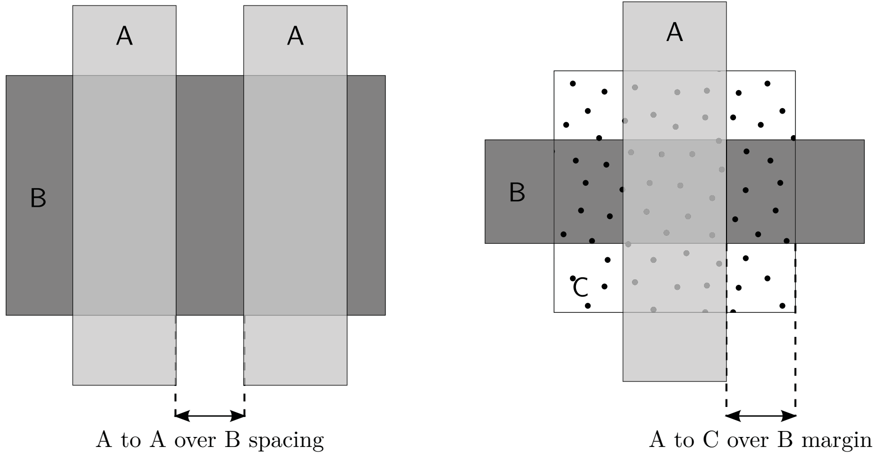


Figure 3.4 Sample of complex design rules defined combining several layers.

$$\text{Extension}(A, B) = d(e_b(B, A \setminus B), e_i(A)) \geq v \tag{3.2f}$$

$$\text{Area}(A) = \frac{1}{2} \sum_{i=0}^{n-1} (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i \geq v \tag{3.2g}$$

$$\text{Corner}(A, B) = d(\text{ConcV}(A), e_o(B)) \geq v \tag{3.2h}$$

On these equations, A, and B regions refer to layout regions which can come from individual layers, or as a result of previous computations combining more layers, and v is the rule value. Therefore we can compute more complex checks, such as Fig. 3.4 shows. Equation (3.3a), and (3.3b) show the resulting design checks.

$$d(A \cap B, A \cap B) \geq v \tag{3.3a}$$

$$A \cap C \setminus B \cap C \setminus [[(A \cap C \setminus B \cap C) \ominus v] \oplus v] \tag{3.3b}$$

Equation (3.3a) does not compare against a rule value, but returns the areas violating the rule.

Electrical models information

Because representing the actual electrical model in XML is impractical due to its complexity, we include only the model metadata. This allows generating small pieces of information needed by the simulators to use the actual model, and generate documentation about the model itself and usage instructions. Listing 3.7 shows the schema part dealing with model information:

```
  <xs:simpleType name="modelLanguage">
    <xs:restriction base="xs:string">
      <xs:enumeration value="verilogA"/>
      <xs:enumeration value="spice"/>
      <xs:enumeration value="vhdl-AMS"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="model">
    <xs:sequence>
      <xs:element name="name"             type="xs:NCName" />
      <xs:element name="modeldescription" type="xs:string" />
      <xs:element name="type"             type="xs:NCName" />
      <xs:element name="lang"             type="modelLanguage" />
      <xs:element name="level"            type="xs:short" />
      <xs:element name="filename"         type="xs:string" />
      <xs:element name="parameter"        type="parameter" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="parameter">
    <xs:sequence>
      <xs:element name="name"        type="xs:NCName" />
      <xs:element name="type"        type="xs:NCName" />
      <xs:element name="value"       type="xs:double" />
      <xs:element name="description" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
```

Listing 3.7 | Device model schema definition.

Then a full model can be defined as shown in listing 3.8:

```
...
<model>
  <name>ofet</name>
  <modeldescription />
  <type>MOS</type>
  <lang>verilogA</lang>
  <level>48</level>
  <parameter>
```

```
    <name>CI</name>
    <type>model</type>
    <value>1.8e-9</value>
    <description/>
  </parameter>
  <parameter>
    <name>VFB</name>
    <type>model</type>
    <value>-2.76</value>
    <description/>
  </parameter>
...
  <parameter>
    <name>L</name>
    <type>instance</type>
    <value>140e-6</value>
    <description/>
  </parameter>
  <parameter>
    <name>W</name>
    <type>instance</type>
    <value>1e-3</value>
    <description/>
  </parameter>
</model>
...
```

Listing 3.8 | Device model XML contents.

As a result of processing this XML information, two documents are generated. The first one is a Portable Document Format (PDF) document (*specific process information* route on Fig. 3.5) that contains a model inclusion and ngspice integration checklist, and the second one, shown on listing 3.9 will be used by the simulator in order to load the compiled model. All netlists including this file will have access to the model and its associated parameters.

```
.MODEL modp PMOS LEVEL=15
+ ALPHASAT=1.327
+ EPS=6.5
+ EPSI=4.3
+ GAMMA=0.119
+ LAMBDA=-1.797e-3
+ M=-3.947
+ TOX=1600e-9
+ VAA=4.763e58
+ VTO=10.452
+ SL=2.3 DVL=0.1 QL=0.7 IOL=1.8e-13
+ SS=4 DVS=3.1 QS=0.6 IOS=8e-13
```

Listing 3.9 | Device model SPICE include line.

The *ngspice* simulator is also used on the commercial tool set, therefore no special action is required for *MaskEngineer/CleWin*.

Compensation information

As in the same case as the DRC rules, compensation operations contain layer operations definitions, and the possible compensation operation or pattern to apply. The applied process is discussed in more detail on chapter 5, which addresses the specific *Layout2Bitmap* compensation tool. Listing 3.10 contains the compensation schema:

```
...
<xs:complexType name="compensationRule">
  <xs:sequence>
    <xs:element name="name"         type="xs:NCName"/>
    <xs:element name="source"       type="layerSelectorOp"/>
    <xs:element name="target"       type="dlayer_ref"/>
    <xs:element name="pattern"      type="xs:string"/>
    <xs:element name="dropDiameter" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
...
<xs:complexType name="patternDefinitions">
  <xs:sequence>
    <xs:element name="patternDefinition" maxOccurs="unbounded" type="patternDefinition"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="patternDefinition" mixed="true">
  <xs:attribute name="file"                type="xs:string"/>
  <xs:attribute name="name" use="required" type="xs:string"/>
</xs:complexType>
...
```

Listing 3.10 | Compensation operations schema.

Listing 3.11 shows the definition of a compensation operation:

```
...
  <compensationRule>
    <name>Metal1-CE</name>
    <source>
      <operation name="OR">
        <operation name="DIFFERENCE">
          <layer gds_num="100" gds_dtyp="10" />
          <operation name="GROW" value="-20">
```

```
            <layer gds_num="100" gds_dtyp="10" />
          </operation>
        </operation>
        <operation name="GROW" value="-40">
          <layer gds_num="100" gds_dtyp="10" />
        </operation>
      </operation>
    </source>
    <target gds_num="100" gds_dtyp="10" />
    <pattern>full</pattern>
    <dropDiameter>0</dropDiameter>
  </compensationRule>
...
```

Listing 3.11 | Compensation operation XML contents.

Following the same approach as the commercial tool set, *Layout2Bitmap* reads directly from the XML database. The compensation operations are handled in the same way as the DRC checks, but in this case, *Layout2Bitmap* does not perform any intermediate result caching. The compensation results are discussed in depth in both chapter 4, and in chapter 5.

3.3.2 FOSS tool set

The purpose of this section is to describe how the set of FOSS tools were adapted to work with the XML database. To reduce the complexity of this integration, and considering that all the selected tools are scriptable, the proposed approach is to build a custom generator for each tool. This generator will be the responsible for generating tool specific customization files.

XML Adaptation

The custom built generator will extract useful information from the XML database, and generate the needed customization files for each separate tool. This is the preferred method, because each different tool has a completely different internal representation. But, because all the tools are scriptable, we can generate specific customization files for them. By performing this approach we take the responsibility out of the tool and into the generator, so we only have to maintain it and we will not need to maintain a source tree for each different tool involved.

To achieve this, the generator exploits different XML facilities. The eXtensible Stylesheet Language (XSL) language is used to perform queries and extract specific information from the database files. XSL provides different methods to manipulate the information:

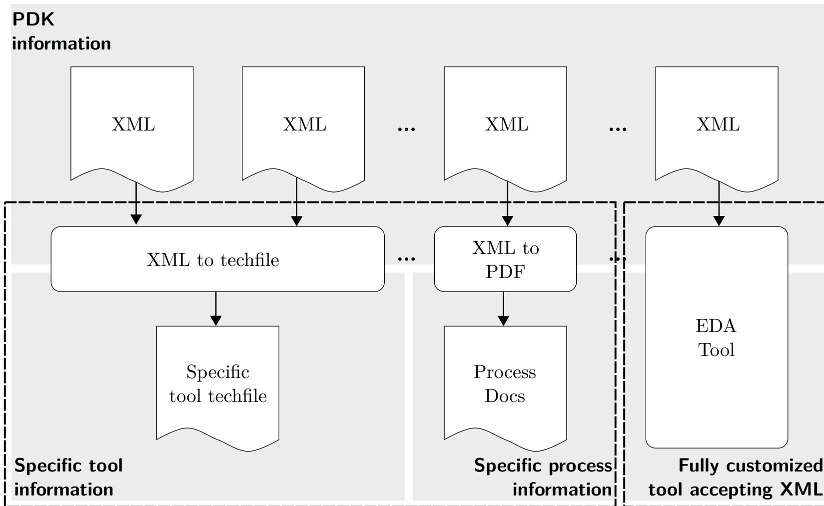


Figure 3.5 XML usage strategies. Shows direct usage by the tools or usage via custom converter.

XSL Transformations (XSLT) provides a language used to transform XML documents to other formats.

XML Path Language (XPath) specifies how different parts of the XML documents are extracted.

XSL Formatting Objects (XSL-FO) is a language to apply formatting to XML files. This language is usually used to generate complex formatted files, such as PDF documents for documentation purposes.

Using these standards, we can process and transform all the technology information into tool usable information.

The left outlined part on Fig. 3.5 represents this strategy. For each different tool technology file, a custom generator will be created. Because both the XML database and the tool custom format will be constant, this effort should only be performed once for each tool, but executed any time the XML database values change.

Layer related information

In order to generate the layer information, the generator is straightforward. It only needs to extract the information and format accordingly. Because the XML database does represent each layer with its needed information, the resulting technology file will be only a list of the available layers with their graphical properties.

Listing 3.12 shows the generated *Glade IC* technology file format:

```
LAYER  OrgSC1       drawing  60  50  (0,255,0)      t t  Slash1     Solid
LAYER  Metal2       drawing  70  40  (0,0,255)      t t  Cross2     Solid
LAYER  Dielectric1  drawing  80  30  (255,255,0)    t t  Empty      Solid
LAYER  Resist1      drawing  90  20  (165,0,150)    t t  Cross2     Solid
LAYER  Metal1       drawing  100 10  (255,0,0)      t t  Backslash  Solid
LAYER  Via12        drawing  75  15  (165,255,255)  t t  cross      thick
```

Listing 3.12 | Layers in *Glade IC* technology file format.

By loading this technology file, *Glade IC* has all the information for the graphical representation of the available technology layers.

Design Rules information

Concerning the DRC rules, the generation becomes trickier. Usually DRC checker tools expect the checks definitions as some sort of script file. This increases the complexity of the file format that needs to be generated. The general procedure used to adapt the XML database to a specific checker is divided into three steps. First, we need a mapping between the available operations defined in the XML database and the target tool. Using this mapping, the generator creates all the layer selectors using the target language. Finally, the generator specifies all the actual rule checks.

Using the XML description and all the defined base operation set, the generator outputs two files for the DRC scripts. The first one, shown on listing 3.13 contains all the rule values:

```
class DR:
  """Design rule class"""
  def __init__(self, mnemonic, description, value, error_msg):
    self.mnemonic = mnemonic
    self.description = description
    self.value = value
    self.error_msg = error_msg
```

```
  def warn(self):
    print "\n *** Warning *** Conflict with rule", self.mnemonic, ":" ,
        self.error_msg , self.value

  def __str__(self):
    return self.mnemonic + ": " + self.error_msg + ' ' + str(self.value)

M1_Angle =  DR(
  "M1.Angle",
  "Minimum angle of Metal1 elements",
  100,
  "The minimum angle of Metal1 elements is less than 100")
...
```

Listing 3.13 | DRC values autogenerated file.

This file is used as a centralized place from which all helper scripts and PCells read the DRC rule values, automatically adapting to rule changes without changing the code. The generated DRC script can be seen on listing 3.14:

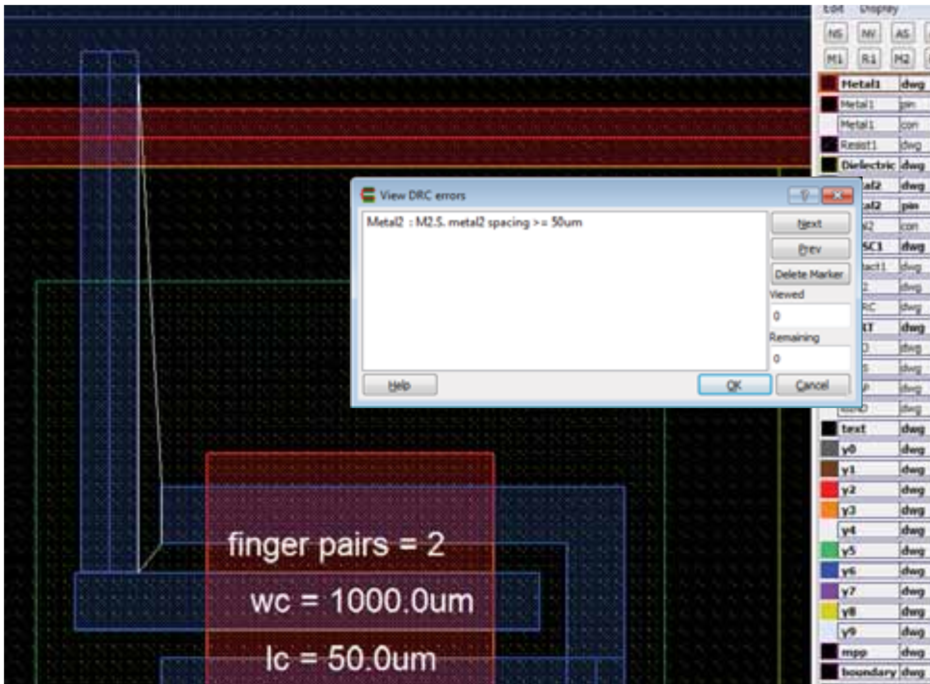
```
...
# Select all the design layers
OSC1_drawing = geomGetShapes("OrgSC1", "drawing")
M2_drawing   = geomGetShapes("Metal2", "drawing")
D1_drawing   = geomGetShapes("Dielectric1", "drawing")
R1_drawing   = geomGetShapes("Resist1", "drawing")
M1_drawing   = geomGetShapes("Metal1", "drawing")
VIA1_drawing = geomGetShapes("Via12", "drawing")

# Generate all derived layers
idp16020080  = geomAnd(M1_drawing, M2_drawing)
idp16157312  = geomOr(geomAnd(OSC1_drawing, M2_drawing), D1_drawing)
...
geomArea(idp16020080, 100, 9e99,
  "The minimum angle of Metal1-Metal2 elements is less than 100")
geomSpace(M1_drawing, samenet, 100,
  "The minimum angle of Metal1-Metal2 elements is less than 100")
```

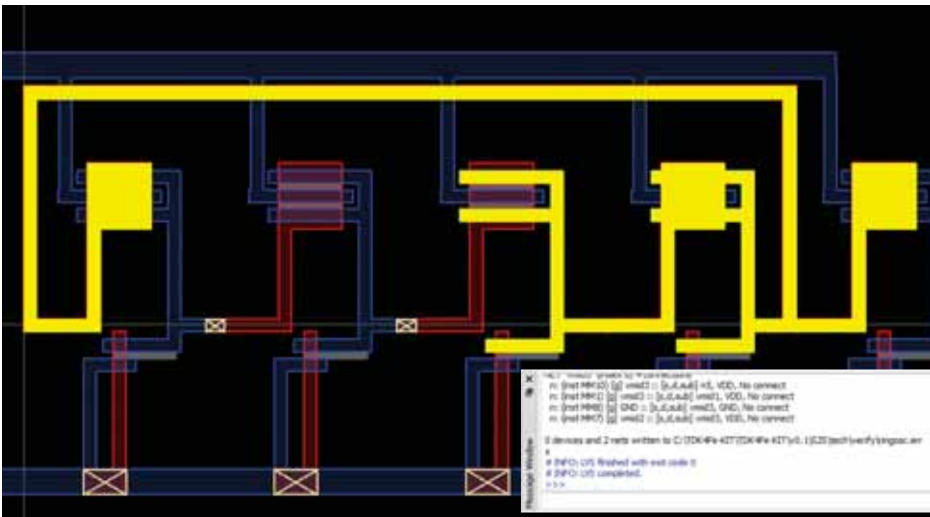
Listing 3.14 | DRC script.

The code first loads all technology layers, and precalculates all derived layers. Following this approach we can reduce DRC checker computation time, because all derived layers used on different rules are calculated only once.

Fig. 3.6a shows the output of the *Glade IC* layout program for a DRC check. In this case all the checker scripts and customizations were performed using the XML database, as explained on section 3.3 below.



(a) | Output of DRC check with one rule violation.



(b) | Output of LVS with all unmatched nets highlighted.

Figure 3.6 | *Glade* customizations.

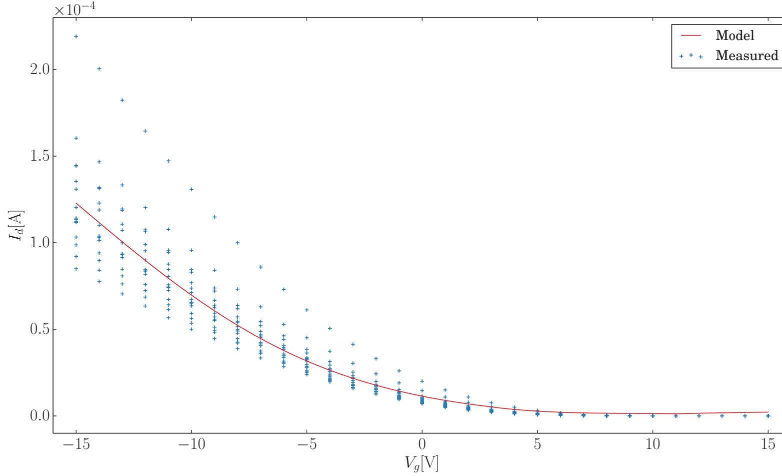


Figure 3.7 Comparison of the integrated model in ngspice and actual device characterization measurements.

Electrical models

The device model case is an example of documentation generation. As stated previously, there are standards in place to represent device electrical models, so it is not necessary to represent it in XML. But by using the model metadata (e.g. name, parameters, default values, etc.), and together with XSL-FO, the PDK generator creates a set of PDF files, with the model information and datasheet, and instructions for using it inside the simulator.

Fig. 3.7 shows an example simulation of the transistor drain current for a gate voltage sweep.

3.3.3 Commercial tool set

In order to integrate the XML database into MaskEngineer, we will extend the PDAFlow API. Therefore, all the tools which connect to this API (*MaskEngineer*, *CleWin*) will have access to the XML design kit. PDAFlow is programmed in C++.

We achieve this integration using *Codesynthesis XML Data Binding compiler for C++* [62]. Provided the XSD schema for the XML database, the compiler generates

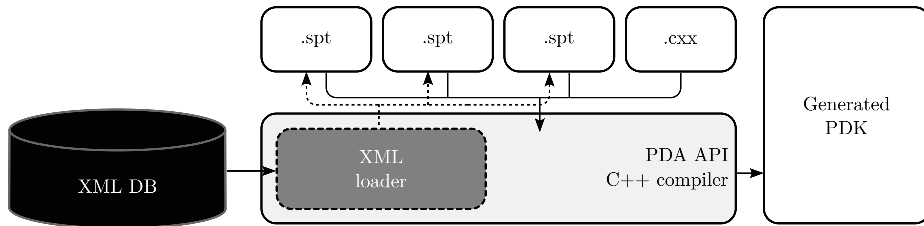


Figure 3.8 PDK generation for PDAFlow API tools. The arrows represent the information flow. Dashed lines represent the added implementation into PDA API code.

C++ classes which gives read and write access to the XML contents from the C++ application.

XML Adaptation

All the tools supporting the PDAFlow API load the design kit from an encrypted library (a Dynamic-link Library (DLL) on windows, and a Shared Object (SO) on linux). This library contains license information, thus allowing the protection of critical technology information, and ensuring that the PDK is usable only by the intended parties. So, in order to obtain a complete kit, XML database has to be integrated at some point in the library creation. On Fig. 3.8 we can see the full PDK generation.

The figure details this integration process. By the creation of a custom XML loader using [62], the database gets transformed into a set of PhoeniX script files. These generated files act as a parent classes, from where the actual design kit files inherit. Therefore, this integration procedure allows a progressive implementation without having to change the design kit scripts. Just changing the XML database will recreate the parent classes and update the design kit. At the end of this step, all files are compiled into a library object, loadable by PDAFlow tools.

Layer related information

MaskEngineer does not work with the concept of layers directly, but introduces the Mask Cross Sections (MCS). The MCS acts as an abstraction to a traditional layer, and defines (for the layout view) how a single shape will be translated into different GDS layers. Fig. 3.9 shows this concept, along with an example which defines a

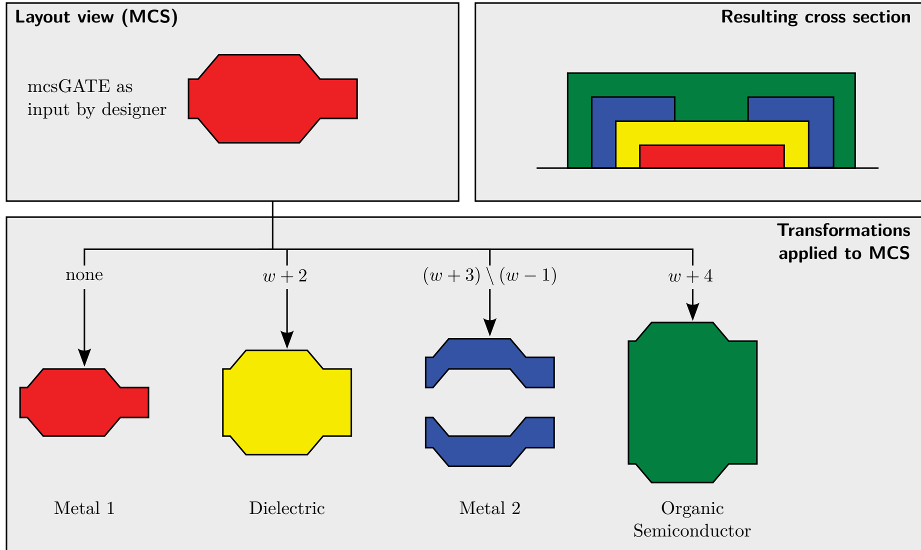


Figure 3.9 *MaskEngineer* cross-section/layer mapping philosophy, w is the shape width. The layer names/colors come from the TDK4PE technology.

full transistor only by drawing the gate area.

Although using this approach looks promising, the technology is still under heavy development making difficult to define clever structures like Fig. 3.9, because they would need heavy changes on each technology iteration step. On the other hand, we can define a 1:1 mapping between MCS and actual technology layers, therefore applying the same strategy as in the free/open source design flow.

Fig. 3.10 shows the result of loading the design kit into *MaskEngineer*. We can see the 1:1 mapping between cross-sections and mask layers.

Design Rules information

The case of the DRC checks is similar to the layers one. In general, all DRC checker tools provide a comparable set of operations, hence, we achieve the integration applying the same strategy as on the free/open source flow. There are some DRC operations which are currently in development, so the XML read utilities translate the DRC checks to *MaskEngineer* script with placeholder function calls. The main

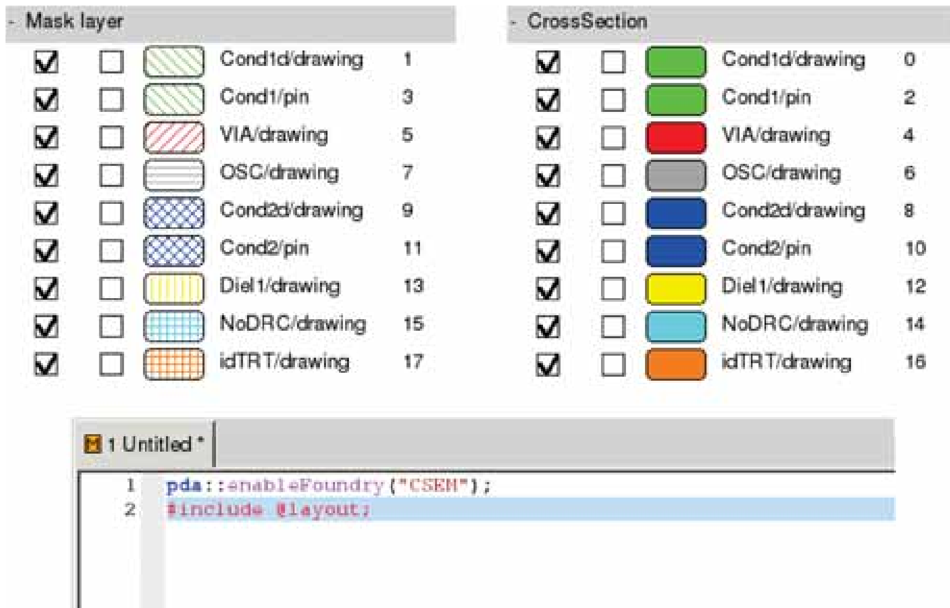


Figure 3.10 Result of loading the design kit into *MaskEngineer*, and the resulting technology layers.

advantage of using this approach is that it decouples the XML and its related functionalities with the core PDAFlow API. This strategy helps on generating a working design kit implementation which will evolve as *MaskEngineer* implements new DRC checks.

Both layer information and DRC rules are generated on the same step. The adapting code generates a set of *MaskEngineer* script parent classes, shown on listing 3.15:

```
#if $(USE_FOUNDRY_Barcelona_02BS)
#define USE_FOUNDRY_Barcelona_02BS 1

#include @mask/tech

class BarcelonaTechnologyBase
  extends TechnologyLayerProcessing() {

  mask::CScreateAddGrid("Metal1d", "filter", "Metal1d", LD(100,10),
                        RGB(255, 0, 0), true, false, 20);
...
  function performDerivations() {
    drcNextLayer();
```

```
    // The numbers are internal representations of the layer
    // AND between Metal1 (12329060) and Metal2 (49315910)
    newLayerAnd("design", 12329060, 49315910, 29999);
    healLayer(29999);
...
  }

  function performChecks() {
    drcLayer_A_EnclosedBy_B(12329060, 49315910,
                            "M1-M2 minimum enclosure", 200);
...
  }

#endif
```

Listing 3.15 | Example of PDAFlow technology base class.

We can see that there is a correspondence between the layer defined in the XML to the MCS (the `mask::CScreateAddGrid` function). All the DRC checks are split between two functions, one which generates the derived layers, and the other one which performs the actual checks. On the design kit, the actual class will inherit from the generated one. An example derived layer definition appears on listing 3.16:

```
#if !$(USE_FOUNDRY_BCA_02MCS)
#define USE_FOUNDRY_BCA_02MCS   1

pcellMaskEngineer_Library(400,0)      pcellME_tdk_bca;
pcellME_tdk_bca.setConnectionReference("box","org");

int techUseBCA;

class foundryBarcelonaTechnology
    dlgname "layout/Technology"
    TexDoc  "Technology description class"
    extends TDK4PE_BarcelonaTechnologyBase() {
...
}

foundryBarcelonaTechnology() techBARCELONA;
Tech=&techBARCELONA;

#endif
```

Listing 3.16 | Example of PDAFlow technology base class.

This extract shows how to create a new layer and save it having the 29999 unique generated identifier. By naming convention, the *MaskEngineer* DRC checker will use all the rule definitions on the base class.

Electrical models and compensation information

The electrical simulator is shared between both design flows. Therefore, as far as database integration we do not need any further modifications. Because *MaskEngineer* does not natively support launching simulations directly from PhoeniX script files, we updated the scripting language. We added support for launching a simulation against the current design.

We performed the integration of *Layout2Bitmap* in a similar way. By extending PhoeniX script language, we added two functions to launch the tape-out process for the current design. This integration is further discussed on chapter 5.

Listing 3.17 shows an example of an inverter design which launches a SPICE simulation:

```
pda::enableFoundry("BARCELONA");
#include @layout;

ml::ground(in0->[0] : ) gnd;
ml::dcsource(out0->gnd@in0 : 30) vcc;
ml::dcsource(out0->gnd@in0 : 30) vin;

var drive = ml::peBCA_PMOS(in0->vin@in0, in1->vcc@in0                  : 50., 200., 15);
var load  = ml::peBCA_PMOS(in0->drive@out0, in1->drive@out0, out0->gnd@in0 : 50., 2000., 15);

var foundry = pda::getFoundry();
var net = pda::newNetlist(foundry, true);

string report = "";
report += ".control\n";
report += "dc |v.vin| 0 30 .1\n";
report += "plot v(|drive.out0|) v(|drive.in0|)\n";
report += ".endc\n";

pda::netSpice(net, "~/invnet.sp", report);
```

Listing 3.17 | Example of inverter design and SPICE simulation.

Listing 3.18 contains the code which creates a simple inverter layout and launches *Layout2Bitmap* from within *MaskEngineer*, therefore generating all bitmap files ready to manufacture:

```
pda::enableFoundry("BARCELONA");
#include @layout;

dsp::clearInfoWin();
```

```
var drive = ml::peBCA_PMOS(in0->[0] : 50., 200., 15);
var load  = ml::peBCA_PMOS(in1->drive@out0 + [0,-900,180] : 50., 2000., 15);

mask::CSselect("Metal2d");
ml::Straight( cin->drive@out0+[0,500,-90] : wfix(800), 2000) conn;

mask::CSselect("Metal1d");
ml::Wire( cin->load@in0 + [400] : wref(conn), {[0,0], [0,2400], [5000,2400]});

// The pdkExport function automatically launches Layout2Bitmap
// and generates the printable bitmap files
pdkExport("/mnt/fb0srv/home/fva/inverter_export/",
          "SimpleInverter");
```

Listing 3.18 | Example of inverter code with *Layout2Bitmap* export.

3.4 Conclusions

The proposed XML database provide an extensible approach to store PE related technology information. Because XML is designed to store structured information making it easily manipulable by computer programs. This allows to abstract technology information from the EDA tool set, and generating specific different customizations for each tool. Therefore, it reduces the complexity of creating new PDK. Because the database and the tools are decoupled, updating the PDK will also update all the customization files used for the different supported EDA tools

The XML representation described on this chapter models the same use cases as the ones covered in OpenPDK [21] (Fig. 1.3, on chapter 1). All non PE specific parts on the proposed schema have a direct correspondence with the OpenPDK specification, hence a conversion to OpenPDK once it is public and widely avaliable and supported across different EDA vendors would not take a significant effort.

In addition, the definition of the DRC rules using basic operations allows the support of complex checks, and maps each rule to a set of commonly defined geometric operatiors. This increases the support of different EDA tools, allowing simpler tools which do not contain complex rule checks perform them using the simple operations, while using the complex functions avaliable on more advanced tools.

All things considered, this design kit abstraction facilities has improved the design kit definition and creation. This approach abstracts each tool details and allows defining all relevant technology information in one unique place and format. The generators then will perform all necessary transformations to have a complete design kit. Demonstrating this, both under TDK4PE project [31], and COLAE project [63],

we managed to create two different design kits for two different technologies. Using the XML database and the generators, we were able to obtain a design kit for both technologies and for both set of tools from one unique representation in a very short period of time: less than two months from initial contact with the foundries with less than three weeks of technical work, including the a few PCells development for each technology. This demonstrates the efficiency of the proposed approach.

To show the usefulness of having this abstract representation of technology information, two different sets of EDA tools have been customized for the use of this PDK, providing a direct path from design to manufacturable designs from both of these tool sets.

Image based technology characterization | 4

After knowing which information is required for a PDK, and which tools cover the design flow, we need procedures to extract critical technology information. Following the presented design flow, the last missing step is the tape-out process which will finally bridge the gap between design and manufacturing. To extract the necessary information, it is essential to develop a new characterization procedure, study ink behavior and extract the FEC rules.

This chapter explains how the experiments to obtain those compensations are generated automatically, and the set of analysis performed on extracted data. Over the rest of this chapter we will present the full analysis framework along with several case studies, ranging from FEC rules extraction to process quality control and characterization.

4.1 Experimental approach

Throughout the TDK4PE and ASPEC-TDK projects we have found that in the process of inkjet printing there is no proper knowledge about the dynamics of fluids, their interaction with the substrates, or the mechanical and thermal effects of these processes. This knowledge is necessary to anticipate suitable compensation strategies which ensure that the morphology of the layer conforms to the expectations of the designed one. Although our own knowledge is far away from those topics, but we can extract from our know-how in microelectronics which are the main shapes and structures where compensations are critical.

Due to this fact we decided to approach the characterization in a proactive way by:

1. Providing a basic set of critical shapes and structures to study compensation needs and effects.
2. Define different compensation strategies (shapes modifications and filling patterns at bitmap level) based on past experimental observations.
3. Produce automatically thousands printed structures with and without different compensation strategies.
4. Characterize such large amount of structures by means of a semiautomatic setup applying electrical tests and image capture.
5. Perform image processing and statistical data analysis on the different results.
6. Extract conclusions on two complementary directions: compensation strategies and rules as well as information about printing process quality to improve its control and accuracy.

Fig 4.1 summarizes this characterization approach used to fix compensation techniques without an in-deep knowledge on the fluidic effects and their related models.

4.1.1 Experiment generation

The *automated layout procedures* section on Fig. 4.1, is the responsible of generating the layout. To perform the generation, it uses the concept of PCells. A PCell is a piece of code which, when executed, will generate a particular layout depending on its parameters.

Together with a scriptable EDA tool, we can develop a script which will automate PCell execution, and therefore, generate a full layout.

To obtain the final design, this script will read a configuration file specifying all the experiment constraints. This includes which type of samples will be used, which PCell parameters will vary, and their range values (for example the sample sizes and orientation), the desired number of repetitions, and several physical parameters such as the foil and sample dimensions.

With this information the script will generate all combinations and repetitions of the samples and perform a random permutation of these generated combinations, arranging the samples across the foil, and saving the used random seed. This last step is very important for experiment repeatability. At the end of the generation

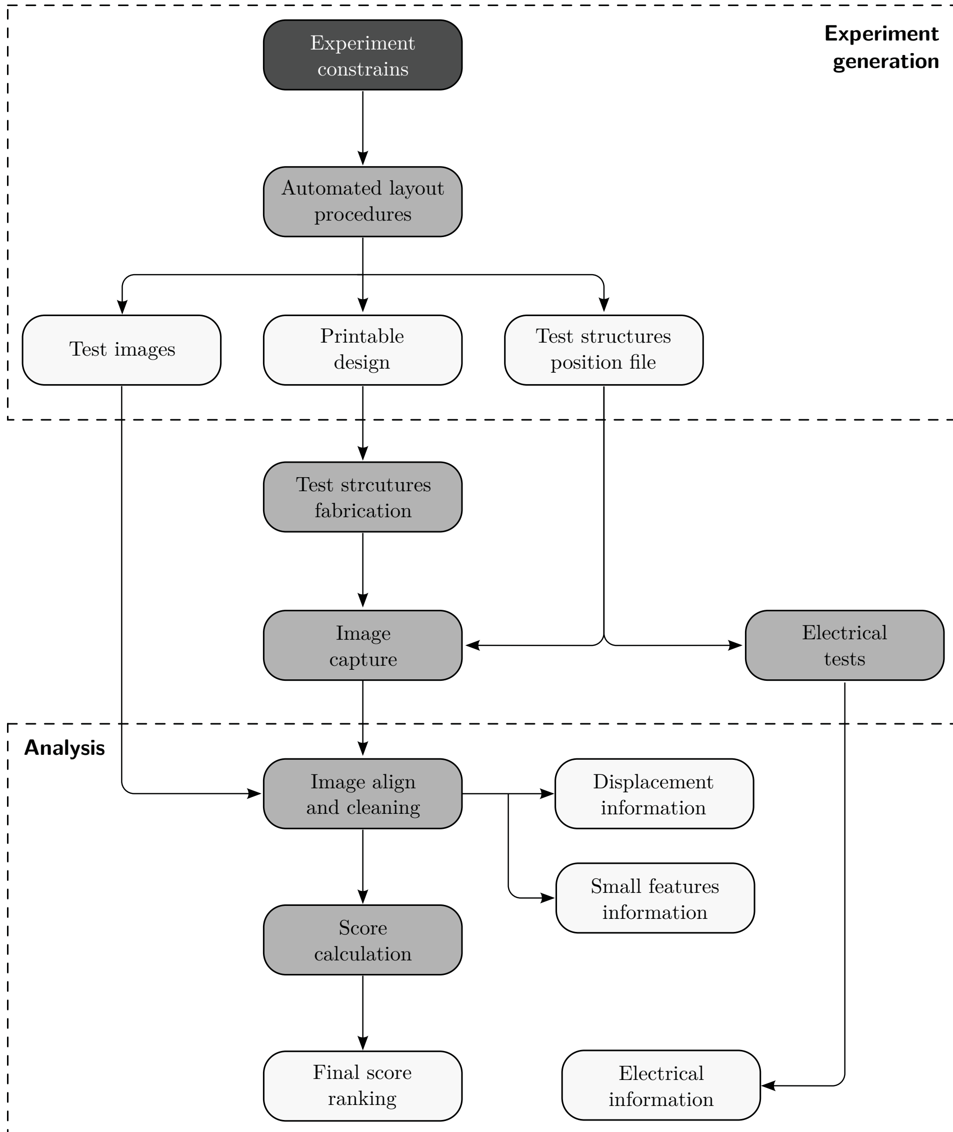


Figure 4.1 General flow diagram of characterization procedure. This diagram contemplates both optical and electrical characterization.

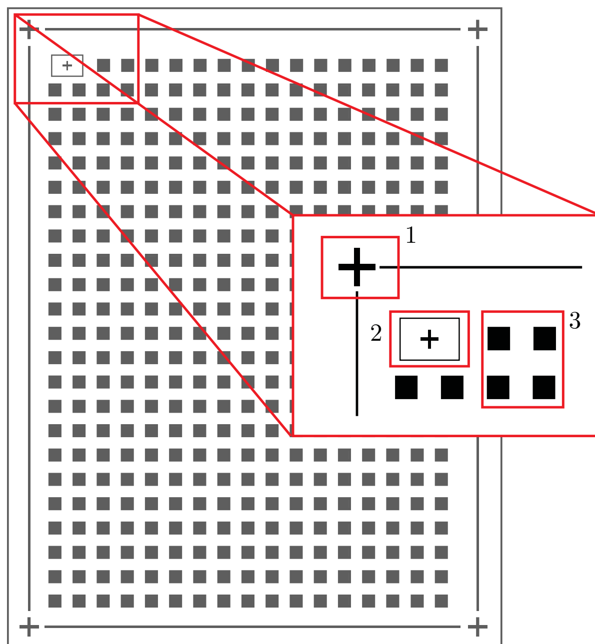


Figure 4.2 Generated experiment layout. The resulting design comprises 1) the substrate alignment marks, 2) the camera alignment structure, and 3) the experiment samples.

step we will obtain a design more tolerant across localized printing errors. As Fig. 4.2 shows, the generated foil contains two extra structures: the manufacturing alignment marks situated on the four foil corners, which are used on fabrication to maintain substrate alignment across each printing step, and the camera alignment structure, used on the analysis phase to correctly align the substrate in the probe station.

After generating the design, it is converted to a set of printable bitmap files using the *Layout2Bitmap* framework. In addition, the generation step also creates a set of reference or template files for comparison purposes. The reference files are a set of bitmaps which represent the expected result after fabrication, and they have the same resolution as the probe station microscope and video capture device. This allows a direct comparison of the obtained versus expected results.

The last piece of generated information is the TVD. The characterization control software will use this CSV file to drive the probe station position [64], therefore

stepping on top of each generated sample. The positioning error of the probe station stepper motor is ±1 µm, well below the printhead error or the camera resolution.

Finally the script generates a TVD file containing the position of each sample across the foil (relative to the camera alignment structure), and each sample parameters. This file will be used to drive the probe station to perform electrical tests and/or to capture an image for each of the approximately 30 000 printed samples [64]. Along with each image, it will save all the parameter values. On the designed experiment, the whole capture step takes approximately 10h (equivalent to approximately 1.2 s per sample) for each foil. But because the probe station is driven by a custom instrumentation control application, it is a batch process that can be left unattended along the night.

Automatically generating the whole test design reduces significantly the needed time, complexity, and possibility of errors. A whole DIN A5 foil design, with the TVD file, bitmap files for manufacturing, and all the template bitmap files used for comparison is generated in less than 30 s using a workstation with an i7 processor. In addition to this reduction in design time, this approach has other benefits, such as allowing an arbitrary position of each sample (the position is stored on the TVD file). This leads to the possibility of area optimization, avoiding large non-printed areas and maximizing the number of test samples per foil.

An extract of the whole generation script can be seen on listing 4.1.1:

```
...

# Different variations on parameters
cells = ["lcomp", "ncomp", "xcomp"]
comp = ["", "p1c1", "p2c1", "c1", "c2", "ce", "h1", "h2"]
sizes = [80, 160, 200, 220]
notch_sizes = [80, 160, 200, 220]
orientations = [R0, R90, R180, R270]

# Configuration parameters depending on foil size
# and sample size
sample_size_x = 900
sample_size_y = 900
frame_size_x = 900
frame_size_y = 900

n_samples_x = 145
n_repetitions = 44

structure_extension = 280

...
```

```
# Generate all combinations for the L and X structures
combinations = [[x, y] for x in comp for y in sizes]

...

# Supposing all different structure combinations are stored
# on all_structures array

random.shuffle(all_structures)

# Place all PCell instances
for i in xrange(len(all_structures) - len(coordinates)):
    y = i / n_samples_x
    x = i - y * n_samples_x
    label = createLabel(x, y + 1)
    xpos = x_start + sample_size_x * x - 140
    ypos = y_start - sample_size_y * (y + 1) - 220
    coordinates.append((label, xpos, ypos))

    origin = Point(xpos * dbu, ypos * dbu)
    cell = cv.dbCreatePCellInst(lib.libName(), structure,
                                'layout', origin, orientation)

    cell.dbReplaceProp('param1', p1)
    cell.dbReplaceProp('param2', p2)
    cell.dbReplaceProp('strategy', comp)
...
```

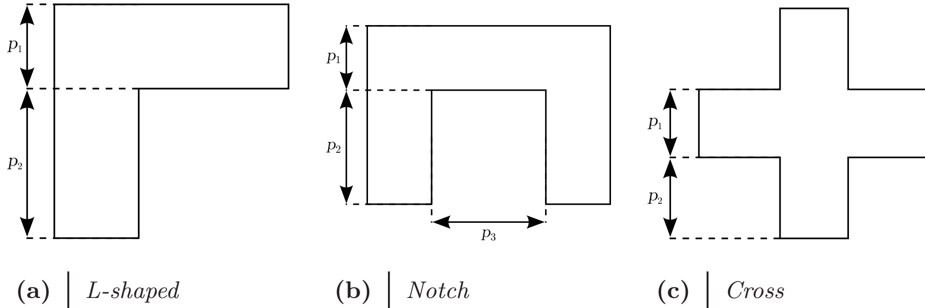
Listing 4.1 | Generation script extract.

After a first part dedicated to the user's customization of experiment constraints, the script generates all different possible parameter values and saves them in an array. This array is randomly shuffled, and finally all the instances are placed on the design.

Compensation structures and PCells

Given our experimental approach, and in order to fulfill our first three experimental requirements, we need to fix which shapes are most critical and need compensations. After a study of the most recurrent shapes appearing on typical PE devices and circuits, and the previous work of Elkin et al. [35], we ended implementing the structures shown on Fig. 4.3.

The test structure set is composed of three different shapes with different parameters. Typically p_1 acts as the structure width and p_2 the length. The *notch* structure has an extra parameter defining the separation (p_3). Therefore by varying all these parameters and rotating the resulting shapes to contemplate all orientations, we



(a) | *L-shaped* (b) | *Notch* (c) | *Cross*

Figure 4.3 | Set of different test structures with their parameters.

can generate a full experiment foil automatically. These PCells have the general structure outlined on listing 4.1.1:

```
# A part from lcomp, it can be ncomp for notch, and xcomp for cross
def lcomp(cv, param1=80, param2=100, ds=20, layer="Metal1", strategy="c2"):

    # In addition to the parameters, we can specify the
    # drop spacing and the target layer, so the PCell
    # can be ported easily to other layers/technologies

    lib = cv.lib()
    dbu = lib.dbuPerUU()
    tech = lib.tech()
    ly = tech.getLayerNum(layer, "drawing")

    # Here should go the drawing code, different
    # for each structure
```

Listing 4.2 | General PCell structure.

The generation script will instantiate a PCell for each sample, with the appropriate parameters and orientation. Therefore, changing PCell parameters will vary the printed structure.

Fig. 4.4 shows the different compensation operations applied to a *L-shaped* test structures. Compensations *c1* and *c2*, are compensation variants applied to the corners, *h1*, and *h2* are "hammerhead" compensations, *ce* makes a one drop trench around the perimeter, and *p1c1* and *p2c1* refer to the application of a stipple pattern on interior junctions. These last compensations are performed in conjunction of one corner compensation, leading to *p1c1*, and *p2c2*. Lastly, Fig. *none* contains the structure without any compensation, acting as a control group.

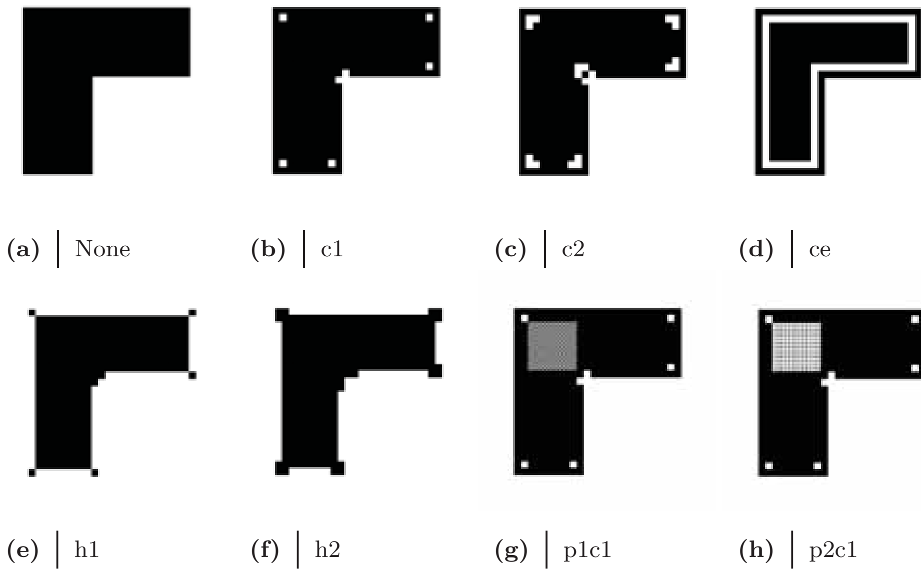


Figure 4.4 Set of compensations applied to a *L* test structure. The *None* corresponds to the control group.

4.2 Analysis

To evaluate the printed shapes, and to extract printing parameters, the characterization setup analyzes each image using a set of image processing algorithms [65]. The overall procedure is to condition captures images by thresholding using Otsu's method [66], clean the obtained image using morphological operations [67], and align it with the expected results using Fourier cross-correlation [68]. Once the images are correctly prepared, we can compare them and extract similarity results. All these procedures will be detailed along this section.

Preliminary analysis on image data led us to consider studying ink spreading, test structure misalignment, and number of detected satellite droplets in addition to the sample score. This data gives more information about process stability and possible substrate curing deformations. Concerning the main points of the experimental approach presented on section 4.1, this section covers the last three points.

4.2.1 Image processing operations

The first step to process the captured image is to convert it to a binary image (B/W). This is performed by the Otsu algorithm [66]. This algorithm calculates a threshold value which will best separate the image into two different regions or classes. This method assumes that the image only has two different regions, and returns the threshold which minimizes intra-class variance, thus maximizing inter-class variance. Therefore, the obtained threshold is the optimal value which separates those two classes. In this case the two classes on the image are the regions with and without ink.

In order to further condition the captured images they need to pass through a cleaning process. This procedure is based on the set of morphological operations in binary images [67, 69]: erosion, dilation, opening, and closing, and on connected component labelling [70, pp. 63-103].

Morphological operations on the captured images are done with an structuring element, which is a shape (in our case a square centered on the origin) with a certain dimensions. Fig. 4.5 has the graphical representation of this set of operations.

The mathematical definition of these operations is as follows. Let A be the captured image, and B the structuring element. Then, erosion is defined as

$$A \ominus B = \bigcap_{b \in B} A_{-b} = \{z \in E | B_z \subseteq A\}, \tag{4.1}$$

where B_z is the translation of B by the vector z, i.e. $B_z = \{b + z | b \in B\}, \forall z \in E$. Therefore, as Fig. 4.5a shows, the final result is the set of all points that the center of B reaches, when moves inside A. The final result is the dark gray shaded part.

Dilation is defined as

$$A \oplus B = \bigcup_{a \in A} B = \{z \in E | (B^s)_z \cap A \neq \emptyset\}, \tag{4.2}$$

where B^s is the symmetric of B, i.e. $B^s = \{x \in E | -x \in B\}$. The final result is the set of points which reaches B, when its center moves inside A. Dark gray shaded part of Fig. 4.5b shows the resulting shape.

Then, combining equations (4.1), and (4.2), the opening is defined as

$$A \circ B = (A \ominus B) \oplus B \tag{4.3}$$

and the closing as

$$A \bullet B = (A \oplus B) \ominus B. \tag{4.4}$$

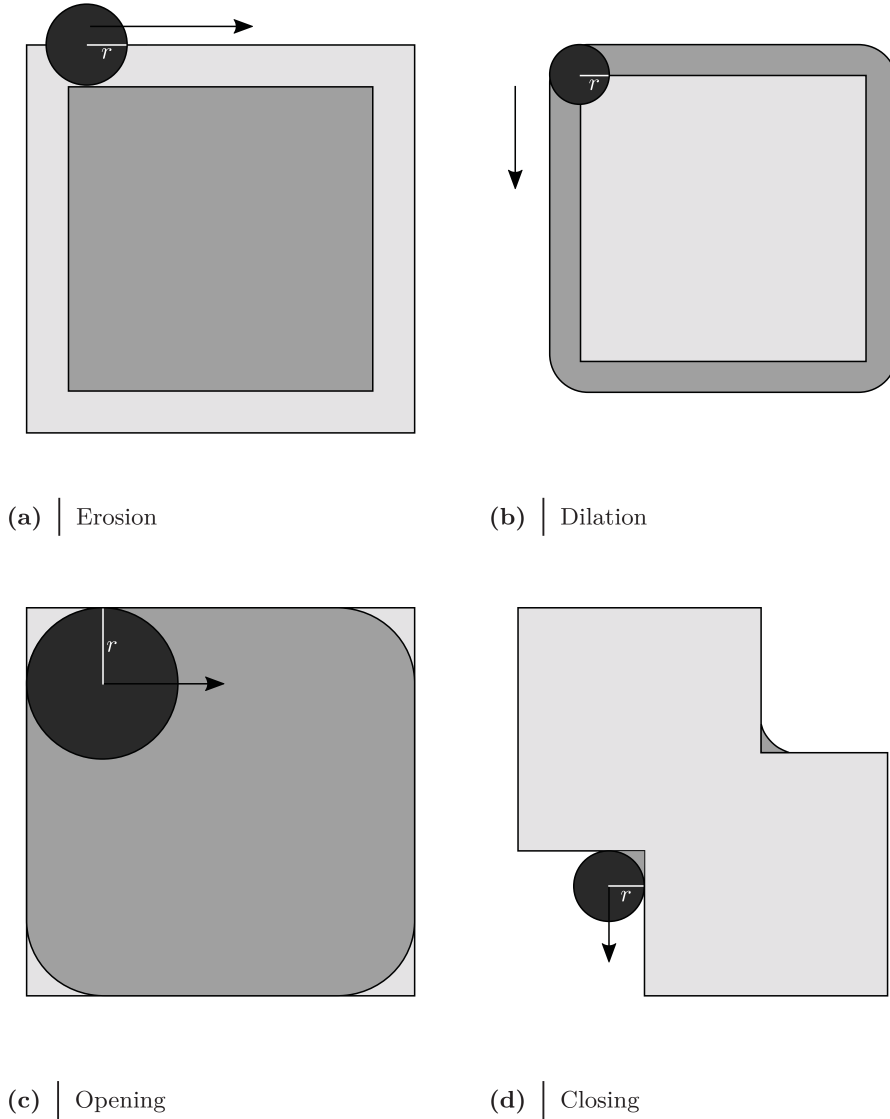
Fig. 4.5c, and 4.5d contains an example of the application of these operations.

Connected component labelling assigns an unique label to each different region of the image, depending on a given heuristic. Fig. 4.6 illustrates the possible heuristics. We consider that the darkest shaded pixels on the figure are *4-connected* to the black central pixel. Adding the light shaded pixels to the *4-connected* ones gives the *8-connected* pixels to the black pixel.

By labelling the captured image and identifying uniquely different connected regions, we can query each part for different values such as the perimeter or the area, thus facilitating the detection of satellite drops.

Printing quality

In the case of our captured images, the opening and closing operations, on eqs. (4.3), and (4.4) respectively, are very important, because they allow the removal of small dark and light features respectively. By performing an opening and a closing, we can eliminate features which are smaller than the structuring element size, hence detecting satellite drops. Then number of eliminated features therefore is a good indicator of printing quality.



(a) | Erosion

(b) | Dilation

(c) | Opening

(d) | Closing

Figure 4.5 | Example of morphological operations using a disk of radius r as a structuring element. The light gray shaded region is the original shape, the black is the structuring element, and the medium gray is the resulting shape.

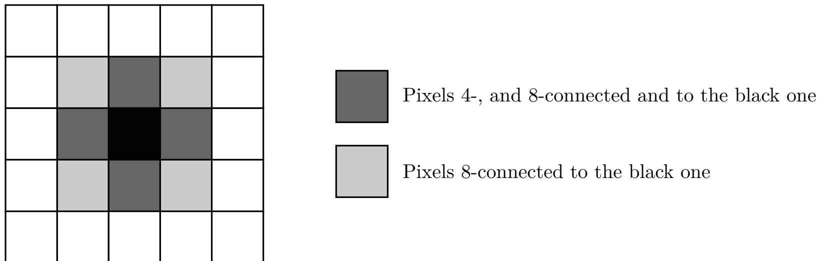


Figure 4.6 | Different heuristics for connected components labelling.

In order to avoid too much distortion on the image, after labelling the image with an *8-connected* heuristic, we discard regions depending on an area threshold. This threshold is dependent on the experiment, but given the dimensions of the experiment samples, it is safe to assume features smaller than two droplets can be discarded.

Fig. 4.7 shows the cleaning procedure step by step, while applied on a single sample. Then, by comparing the initial image with the cleaned result, we obtain the number of features eliminated (i.e. number of satellite drops).

As the result of this process, we can plot the number of removed features in a heatmap representing the whole foil, as in Fig. 4.8a. This representation gives an idea of any possible correlation between sample position and printing direction with the number of removed features.

We can observe some periodic horizontal stripes on Fig. 4.8a. Taking into account the printing direction for this specific foil (from right to left and bottom to top), we can observe that the stripes follow the printhead, and are approximately equally spaced.

To confirm if there is a periodic effect on the number of removed features we can calculate the mean across the x axis. Fig. 4.8b contains the obtained profile. Although it is difficult to see some periodicity, we can apply eq. (4.6) with the profile with itself, therefore applying autocorrelation [71]. By autocorrelating a signal we can extract existing periodic patterns possibly obscured by noise. Fig. 4.8c shows the results of the autocorrelation. We can observe two distinct peaks at $x = 34$, and $x = 68$. Hence we can confirm that it exists a periodic component, but due to how the foils were fabricated we still cannot extract which is the real cause behind this effect.

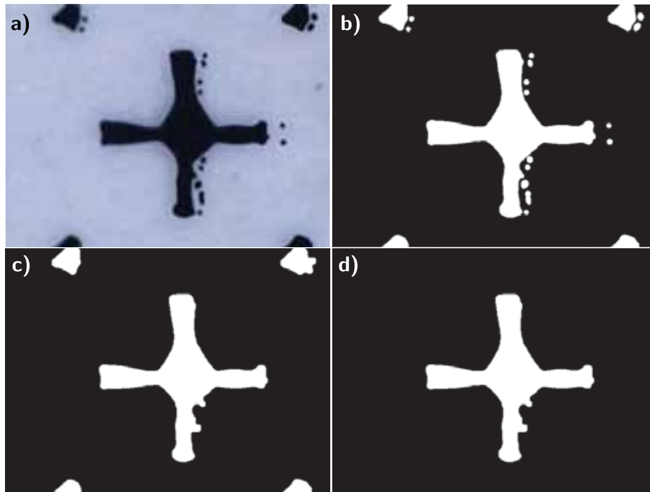


Figure 4.7 Results of the cleaning process step by performed operations: a) captured image, b) after thresholding, and opening/closing, and d) after small features removal.

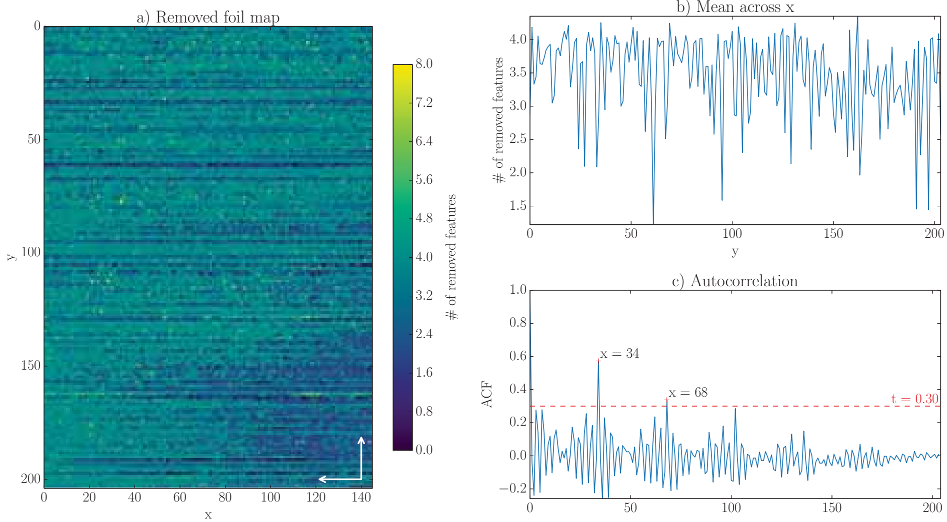


Figure 4.8 a) Removed map of a whole DIN A5 foil. The printing direction is indicated by the white arrows (right to left and top to bottom), b) Mean values across the x axis, and c) Autocorrelation plot.

4.2.2 Misalignment

The analysis procedure uses Fourier-based alignment methods to align captured images and test template images. Fourier-based alignment relies on the property that the Fourier transform of a shifted signal has the same magnitude but linearly varying phase [72, p. 59]. In addition, by another Fourier property, the convolution of two signals in spatial domain correspond to the multiplication on Fourier domain [72, p. 60], and due to the time reversal property [72, p. 60] we have that $f(-x) \overset{\mathcal{F}}{\leftrightarrow} \mathcal{F}^*(f(x))$.

The cross-correlation operation between two signals measure the similarity, and is calculated as

$$(f \star g)[n] \overset{\text{def}}{=} \sum_{m}^{\infty} f^*[m] \cdot g[m+n] \tag{4.5}$$

being f and g the two functions, and n is the displacement.

Considering that $(f(x) \star g(x))[n] \leftrightarrow (f(x) * g(-x))[n]$, being $(f * g)$ the convolution, the previously defined Fourier properties, we define the cross-correlation of the two images in the Fourier domain as

$$(i(x,y) \star t(x,y)) = \mathcal{F}^{-1} \left\{ \mathcal{F}[i(x,y)] \cdot \mathcal{F}^*[t(x,y)] \right\}, \tag{4.6}$$

where $\mathcal{F}$ is the fourier transform, $\mathcal{F}^*$ is the complex conjugate, and $i(x,y)$ and $t(x,y)$ are the captured and template images respectively.

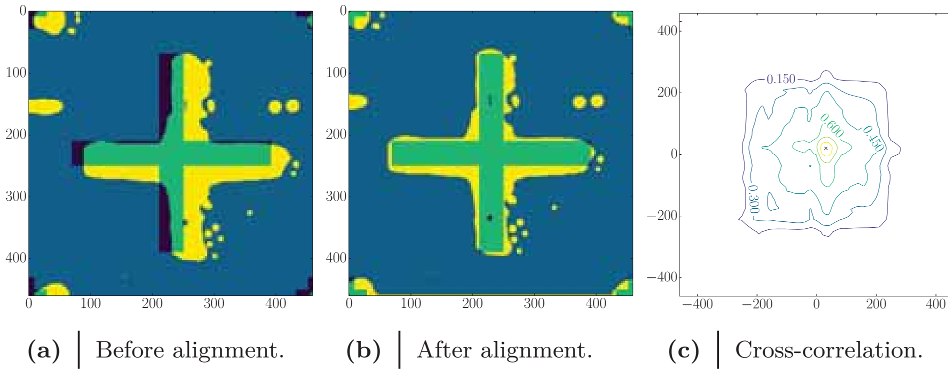
Then, the actual displacement between the captured and test image is the location where the maximum of the $\text{xcorr}(x,y)$ function (4.6). Therefore the displacement is

$$(x_0, y_0) = \underset{(x,y)}{\arg\max} \, (i(x,y) \star t(x,y)) \tag{4.7}$$

where $i(x,y)$ is the captured image, $t(x,y)$ the template test image, and x_0, y_0 is the displacement in the x and y directions respectively.

Substrate deformation

Although the sample position was automatically generated from the design, initial analysis procedures shows some sample structure misalignment. Due to the



(a) Before alignment. **(b)** After alignment. **(c)** Cross-correlation.

Figure 4.9 Example of image misalignment. On the cross-correlation contour plot, the maximum is marked with a black × (central dot on (c)).

printhead and sample dimensions and the sample positions we cannot attribute this misalignment to the printer stepper. The thermal curing processes used after printing each layer cause some slight substrate deformation, which can accumulate error between layers. Fig. 4.9 shows an example comparison before and after alignment.

The automatic misalignment calculation allows to quantify how much misalignment there is along both X and Y axes of the foil. Hence, modelling this behavior can lead to corrections capable of improving the whole process yield, as usually fabricated devices depend on more than one process step, thus being sensitive to alignment issues.

Taking all misalignment calculated for each sample using eq. (4.6) and representing it as a vector field, results in the plot on Fig. 4.10. The calculated vector for each sample is $\vec{v} = (x_0, y_0)$. Hence, the amount of substrate deformation is calculated by taking the magnitude of $\vec{v}$.

Looking at the stream plot on the figure we can conclude that it exists some radial deformation centered around the $(0, 0)$ position, confirming that the substrate expands on the printing process. The deformation is centered on the top-left corner because it is the origin position of the probe station, therefore all misalignment is relative to the start. It has to be noted that the represented stream lines are an approximation, showing the general direction of the underlying vector field.

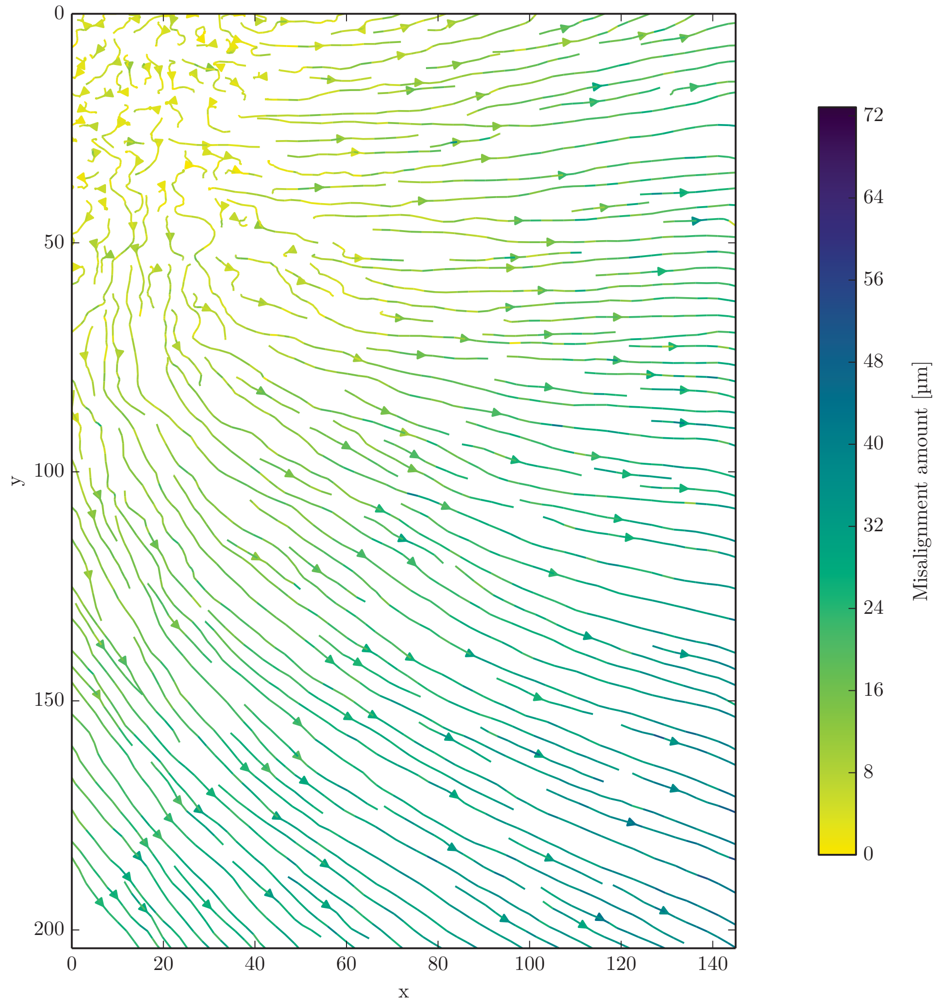


Figure 4.10 Misalignment across a whole DIN A5 foil. The vector streamlines indicate the substrate dilation directions, and the color the amount.

$i'(x,y)$	Region
0	Part without ink that should be filled
1	Part without ink that is not filled
2	Part with ink that is filled
3	Part with ink that should not be filled

Table 4.1 | Different image region identifiers.

4.2.3 Score calculation

In order to evaluate the fidelity between the captured image and the expected results it is necessary a similarity measure. At this processing stage, both captured and template images have the same dimensions, and are aligned to maximize overlap.

The metric has the following requirements: a) it needs to adapt to different sample dimensions and shapes (without too much penalty), and b) it needs weighting depending on the distance. It is preferable to have small ink spreading along the whole sample rather than a bigger quantity in a localized position.

Therefore, the proposed metric is as follows. Let i be the captured image, and t the test template image. The compared image i' is defined as

$$i' = 2 \cdot i - t + 1, \tag{4.8}$$

where table 4.1 contain the different possible values of i'.

Therefore, with the compared image we can define two regions of interest as

$$R_1 = \{(x,y) \in i' : i'(x,y) = 0\} \tag{4.9a}$$

$$R_2 = \{(x,y) \in i' : i'(x,y) = 3\} \tag{4.9b}$$

To calculate the final score, we apply a Distance Transform (DT) [73, 74] to each of the regions calculated on equations (4.9), adding the pixel values and normalizing by the shape perimeter P:

$$S_p = \frac{\sum \mathrm{DT}(R_1) + \sum \mathrm{DT}(R_2)}{P} \tag{4.10}$$

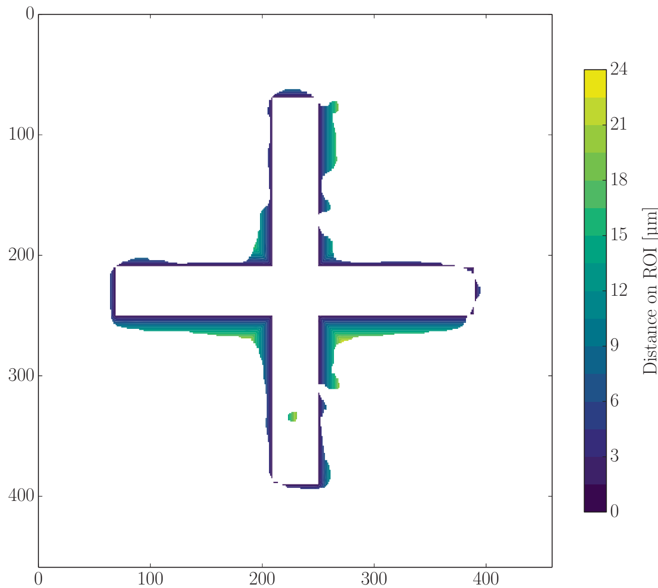


Figure 4.11 | Resulting distance map transform on the regions of interest.

An example result showing the resulting DT appears on Fig. 4.11. Then the captured image similarity to the expected results increases as $S_p \to 0$, giving a comparable measure for the samples. In addition, changing shape, and shape dimensions alters the perimeter, normalizing the final score, and making different structures directly comparable.

Ink spreading

Using this similarity metric between obtained printed results and the template image, and combining it with the application of the image processing morphological operators defined on section 4.2.1 it is possible to grow and shrink the template image (using dilation and erosion).

By growing and shrinking the template image we compensate possible ink spreading or reduction. Therefore, if we consider the score calculation as in eq. (4.10) as a function of the amount the test image is dilated (or eroded), minimizing the results lead to the amount of ink spreading.

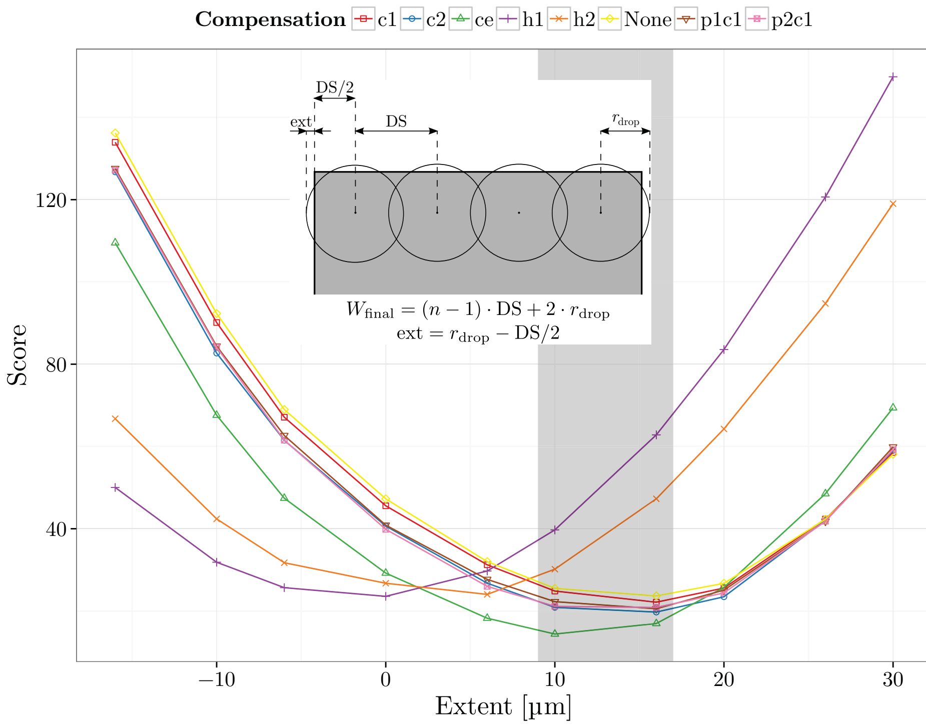


Figure 4.12 Score median depending on test template extension. The diagram shows the drop placement with the appropriate measurements.

Fig. 4.12 shows the score median value depending on the amount the template image is grown. The shaded region belongs to a minima of all compensation scores and belongs to the ink amount bulging from each shape, except for *h1* and *h2*, which makes sense as those compensations remove a drop from the border of the sample (and as a side effect, shrinking the shape). This amount coincides approximately with the drop radius, as seen on the diagram on Fig. 4.12.

We can observe that the estimated ink spread on all printed structures corresponds approximately to the drop radius. Therefore, confirming Ramon et al. results [75], the estimated final feature width will be:

$$W_{\text{final}} = (n-1) \cdot \text{DS} + 2 \cdot r_{\text{drop}}, \tag{4.11}$$

where r_{drop} is the drop radius, DS is the drop spacing, and n is the number of drops deposited. These measurements apply to silver nanoparticle ink on top of PEN substrate.

4.2.4 Statistical procedures

The last step in result processing is performing statistical tests to the obtained data. Having this procedure established, we are able to state conclusions about the manufacturing process and compensations with confidence.

Starting with compensation information, we can model the expected score with a linear regression [76]:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_7 x_7 + \epsilon \tag{4.12}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (i.e. it is a random variable following a normal distribution with 0 mean and σ^2 variance). In the compensations case, β_0, and $\beta_1 - \beta_7$ correspond to the test case and the compensation effects respectively.

Because score is always positive we can expect a highly skewed distribution, hence a non-normal ϵ on eq. (4.12).

To reduce skewness and transform the data to more normal-like distribution, we use the Box-Cox transform [77, 78]. This transform belongs to a family of functions which perform monotonic transformations on data using power functions. The general equation is

$$Y_i^{(\lambda)} = \begin{cases} \frac{Y_i^{\lambda}-1}{\lambda}, & \text{when } \lambda \neq 0 \\ \ln(Y_i), & \text{when } \lambda = 0 \end{cases}, \tag{4.13}$$

where λ is extracted from the data. To obtain λ for the score data, we apply Maximum Likelihood Estimator (MLE) methods.

As shown on [79, chap. 14], the log-likelihood of the compensations model can be written as

$$\log \mathcal{L} = -\frac{n}{2} \log \left[\sum_{i=1}^{n} \left(\frac{Y_i^{(\lambda)} - (\beta_0 + \beta_1 x_i)}{\dot{Y}^{\lambda}} \right)^2 \right] \tag{4.14}$$

for all our variables of interest. On eq. (4.14), $\dot{Y}$ is defined as

$$\dot{Y} = \exp \left[\frac{1}{n} \sum_{i=1}^{n} \log Y_i \right] \tag{4.15}$$

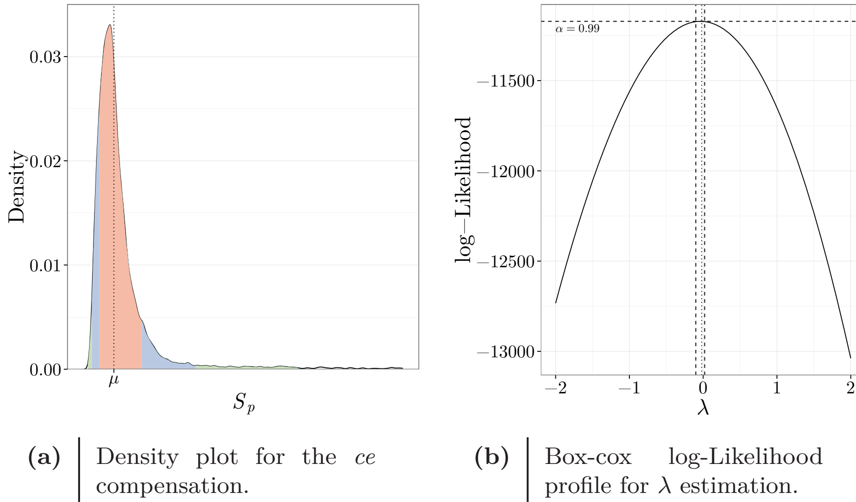
Having the likelihood function $\lambda \mapsto \log \mathcal{L}$, we can find the appropriate λ by finding the maximum. If desired, an estimated confidence interval for λ can be obtained by intersection with a line at a distance of χ^2_α from the maximum.

Fig. 4.13 shows these methods applied to a whole foil. Looking at Fig. 4.13a we can see that the results are highly positively skewed, therefore we need to apply eq. (4.13) to the data.

The result of applying MLE techniques as in eq. 4.14, and finding the maximum value is shown on Fig. 4.13b. Looking at the plot we can assume that $\lambda = 0$ for our data, therefore the results can be approximated as a log-normal distribution. Note that λ is not exactly 0, but in order to facilitate future calculations we can use a λ value from a set of common values from the set $\lambda \in \{-2, -1, -1/2, 0, 1/3, 1/2, 1, 2\}$

Table 4.2 contains the measurements of skewness and kurtosis of the transformed data, and the calculated values approximate to the normal distribution ones.

Once data is transformed and has a normal like distribution we can proceed to estimate means and compare variances. We can observe on Fig. 4.14a that the box whiskers are approximately the same size, and that the mean and median lie approximately in the same value. Looking at Fig. 4.14b, which plots the logarithm of the score data quantiles against a theoretical normal distribution, shows a straight line, therefore confirming that the score data is approximately normal.



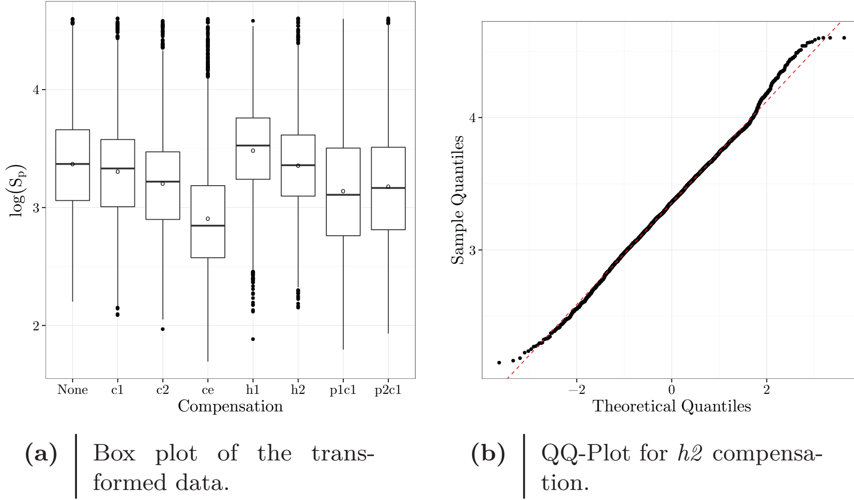
(a) Density plot for the *ce* compensation.

(b) Box-cox log-Likelihood profile for λ estimation.

Figure 4.13 Initial data distribution and profile log-Likelihood for Box-Cox transform. Note: The shaded regions correspond to σ, 2σ, and 3σ intervals.

Compensation	Skewness (γ_1)	Kurtosis (γ_2)
c1	0.12532329	3.050832
c2	0.25280056	3.229631
ce	0.69212017	3.688154
h1	-0.38408613	3.107953
h2	0.04551156	3.184986
None	0.13634569	2.708675
p1c1	0.21738517	2.572452
p2c1	0.23738604	2.729070

Table 4.2 Measures of skewness and kurtosis of $log(S_p)$ for different compensations. It approximates the normal distribution values ($\gamma_1 = 0, \gamma_2 = 3$)



(a) Box plot of the transformed data.

(b) QQ-Plot for *h2* compensation.

Figure 4.14 General distribution of transformed score results.

Once we can state the normality of the score results, we can proceed to study which compensation behaves better (if any), and analyze the score differences between them. The first step will be the calculation of the mean confidence and the prediction intervals on the data.

The mean confidence interval gives an approximation of the overall score values, so we have an initial indicator on which compensation behaves better. The interval is defined as:

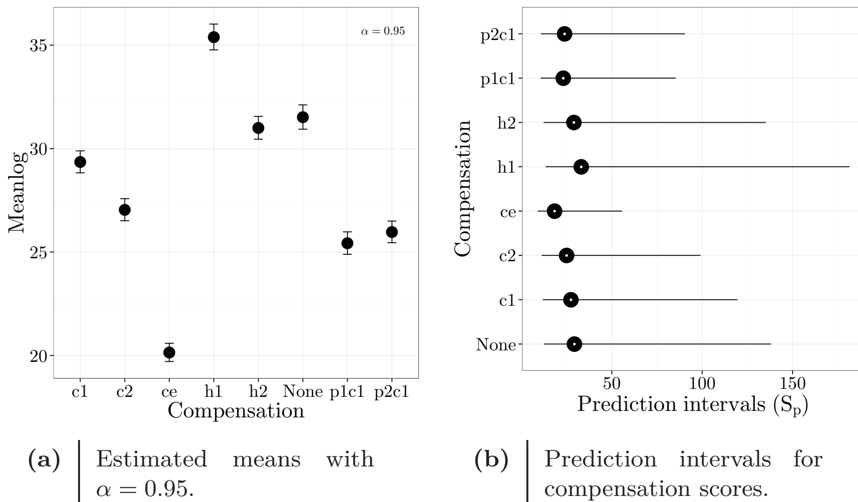
$$\mu \pm \bar{x} - t_{\alpha/2}\frac{s}{\sqrt{n}} \tag{4.16}$$

This estimation of the mean of the whole population has an α degree of confidence, given there are n samples in the experiment. The $\bar{x}$ and s variables refer to the sample mean and standard deviation respectively.

In addition, using eq. (4.17) we obtain the prediction intervals. These intervals estimate the score values for future observations with a given probability, therefore allowing outlier detection (i.e. filtering non-valid samples). It is defined as

$$\bar{x} \pm z_{\alpha/2}\sigma\sqrt{1 + 1/n}, \tag{4.17}$$

where $z_{\alpha/2}$ is the critical value for the normal distribution with α degree of confidence,



(a) Estimated means with $\alpha = 0.95$.

(b) Prediction intervals for compensation scores.

Figure 4.15 Initial estimated means and prediction intervals for outlier filtering

σ is the population standard deviation, and n is the number of samples.

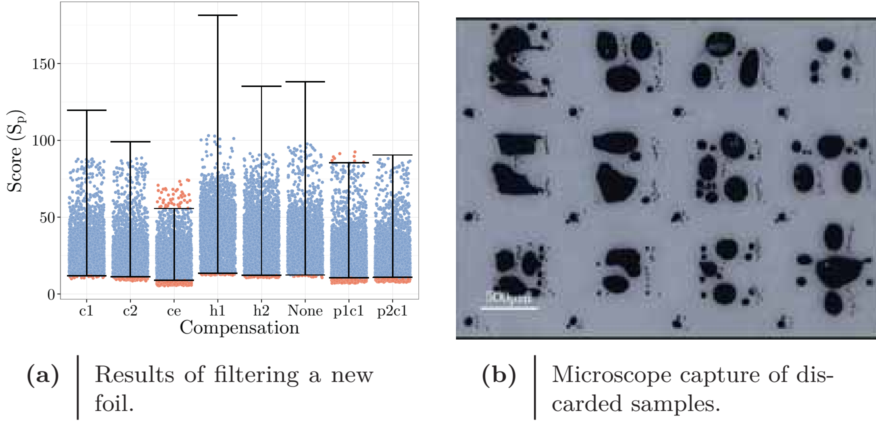
Having the prediction intervals allows filtering new foils, repeating analysis to compare sample score means without the outliers effect.

Fig. 4.15 show the outcome of these two calculations. The preliminary mean estimates, on Fig. 4.15a, shows that the *ce* compensation behaves the best for all structures (S_p is the lowest), but still we cannot be sure of the significance of the differences.

To asses the significance we filter a different new foil using the prediction intervals, shown on Fig. 4.15b to discard non-valid samples. Fig. 4.16a and Fig. 4.16b illustrate the discarded samples on a new foil and a capture of some of them. Clearly, the discarded samples show evident printing and dewetting effects not caused by the compensations themselves.

By applying this filter we obtain a first indicator of that foil printing quality. If the number of discarded samples is too high means that there are some possible manufacturing issues.

To assess the statistical significance on compensation effects comparison we conduct a one-way ANOVA. ANOVA is a generalization of Student's t test [76] which reduces the chance to make Type I errors when performing multiple comparisons (this kind



(a) Results of filtering a new foil.

(b) Microscope capture of discarded samples.

Figure 4.16 Prediction intervals and filtered new foil data.

of errors are also called *false positives*). In addition a Welch correction [80] is included, thus compensating for possible heteroscedasticity of the data (i.e. the variances among all compensations are not equal).

The test shows that the compensation has a significant effect on final score value for all tested compensations ($p < 0.05$, $\eta^2_{\text{partial}} = 0.298$), thus confirming that applying compensations has an effect on final sample score. With these results is convenient to perform a post-hoc analysis to obtain which compensation behaves better. We use the Tukey HSD analysis [81], which obtain the actual differences on the mean score with an $\alpha = 0.95$ confidence. The results are shown on Fig. 4.17. Because a lower score value means more similarity with the expected results, values more negative imply a better improvement. From the figure, it is evident that the *ce* compensation is the best option, confirming previous results, and reducing significantly the amount of ink deposited outside intended regions.

Fig. 4.18 shows actual microscope image captures of the effect of the *ce* compensation on notch structures. The captured images come from two different samples with the same layout pattern and parameters (i.e. width, length, separation, and orientation). We can observe that applying the *ce* compensation leads to a better matching structure, reducing the amount of ink bulging into the inner notch feature. The sample also shows sharper angles on the corners, thus in general, the final shape represents more faithfully the original.

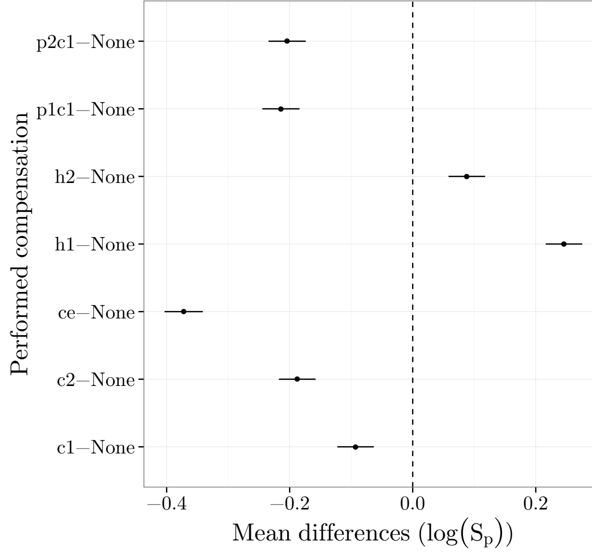


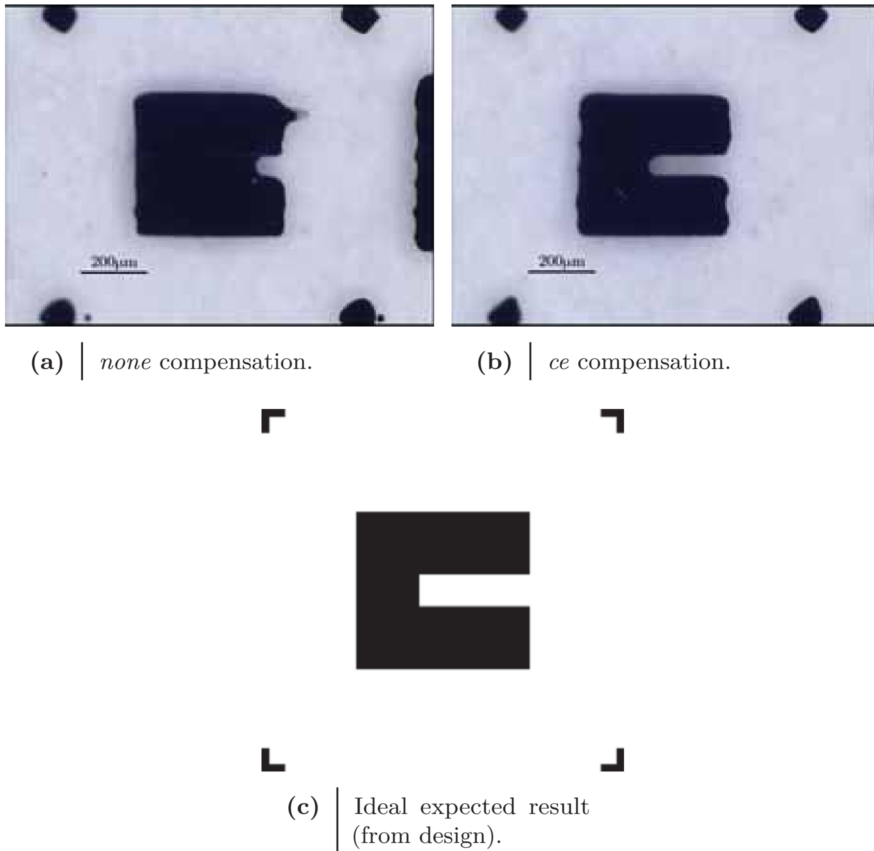
Figure 4.17 | Tukey HSD post-hoc test results.

4.2.5 Electrical tests

So far the tested structures used for compensation comparison are not directly common used devices on PE designs, although commonly appearing as parts of them. Therefore, to validate the best obtained compensation against an interesting device, we chose to test Organic Thin Film Transistors (OTFTs), whose general structure and cross-section appear on Fig. 4.19.

One of the most critical parts is the shape of the fingers, which correspond to the source and drain of the transistor. Looking at the most used compact model for OTFTs, the UMEM model by Estrada et al. [82], the transistor current above threshold is defined as

$$
\begin{aligned}
I_{\mathrm{DS}} = & \frac{W}{L} \cdot C_{\mathrm{diel}} \frac{\mu_{\mathrm{FET}} \cdot (V_{\mathrm{GS}} - V_{\mathrm{T}})}{\left(1 + R\frac{W}{L} \cdot C_{diel}\mu_{\mathrm{FET}} \cdot (V_{\mathrm{GS}} - V_{\mathrm{T}})\right)} \\
& \times \frac{V_{\mathrm{DS}}(1 + \lambda \cdot V_{\mathrm{DS}})}{\left[1 + \left[\frac{V_{\mathrm{DS}}}{V_{\mathrm{DSat}}}\right]^{m}\right]^{1/m}} + I_0
\end{aligned}
\tag{4.18}
$$



(a) *none* compensation.

(b) *ce* compensation.

(c) Ideal expected result (from design).

Figure 4.18 Comparison of the same structure with different compensations.

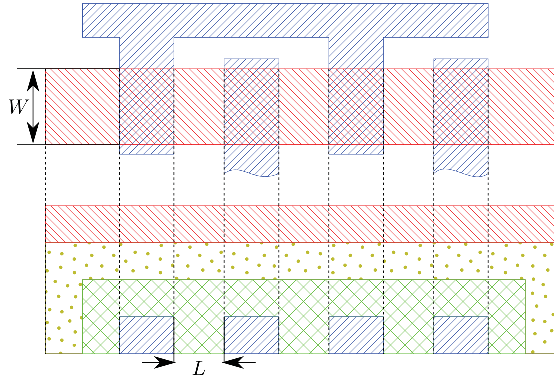


Figure 4.19 OTFT structure as designed and resulting cross-section. Blue parts are the source and drain electrodes, red is the gate. The channel forms between fingers.

where W and L are channel width and length respectively, C_{diel} is the gate capacitance, R is the summed source and drain electrodes resistance, and λ and m are fitting parameters.

Therefore if we are able to reduce channel length (L) while maintaining the width (W), according to eq. (4.18), the transistor current will increase, thus the transistor performance will be improved [83].

To validate the whole characterization process and check that the selected compensation performs correctly, we generated and characterized different fingered structures. This particular structure corresponds directly to the source and drain of OTFTs, and can be tested with an electrical conductivity test, hence reducing substantially the analysis time. This test consists in placing a probe to each side of the structure, and then checking for conductivity. In the case of electrical conductivity, means that the source and drain electrodes are touching, therefore making them unsuitable for a working OTFT device. Fig. 4.20 illustrates the electrical conductivity test results.

The plot summarizes the proportion of good finger structures (i.e. not short-circuited) for the *ce* compensation and the control group. The results suggest that looking at 40 µm design separation, and smallest finger width, we are able to print approximately half of the finger structures without short-circuit (52%), and by increasing the finger width to 100 µm we achieve a 89% of correct structures. On the other hand, we have to increase the separation to 60 µm to start having working finger structures without compensation, and we only obtain a good working

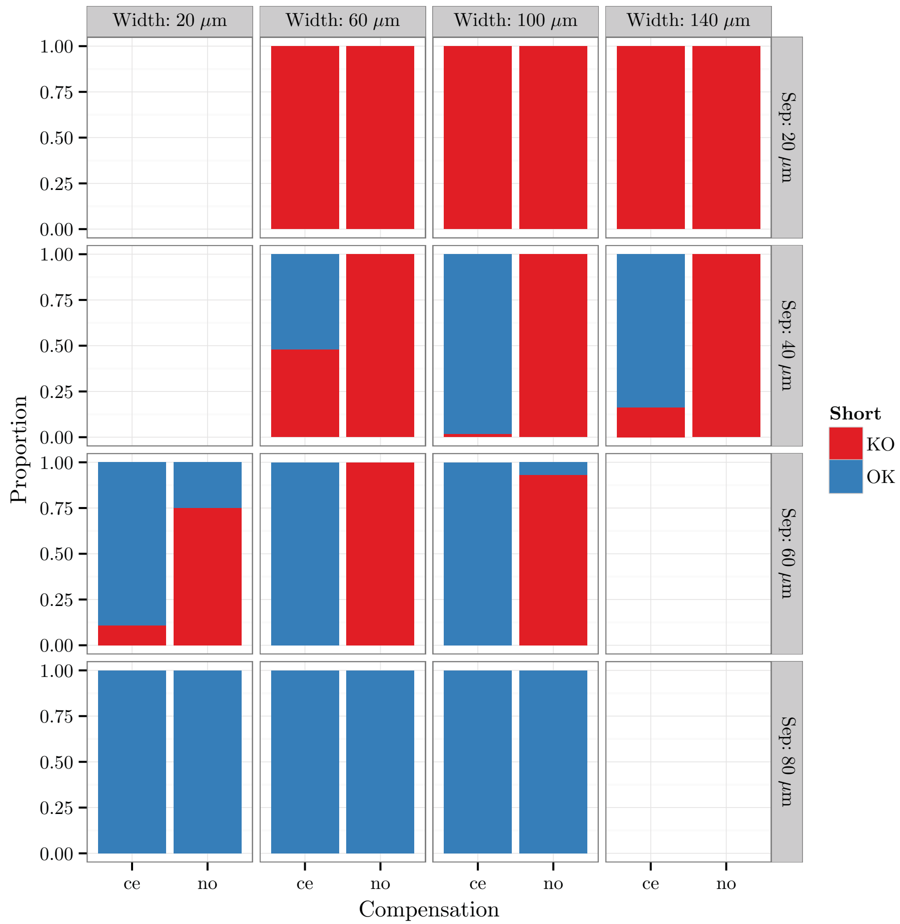


Figure 4.20 Electrical test results of the finger structures. The separation is not the actual designed separation, but the effective separation accounting to ink spreading.

		40 µm sep.	60 µm sep.
20 µm width	ce	—†	0.893 ± 0.061
	no	—†	0.250 ± 0.146
60 µm width	ce	0.519 ± 0.130	1.000 ± 0.017
	no	0.000 ± 0.079	0.000 ± 0.077
100 µm width	ce	0.981 ± 0.025	1.000 ± 0.016
	no	0.000 ± 0.084	0.070 ± 0.123

†This samples were not generated because the finger width is too low, so the compensation cannot be applied.

Table 4.3 Approximated $\alpha = 0.95$ confidence intervals for proportion of working fingers.

proportion if we double the minimal separation, resulting to 80 µm.

The electrical results summarized on 4.20 can be modelled with a binomial distribution because the outcomes are independent success/failure experiments (i.e. there is a short-circuit or not). Therefore, approximating the distribution error as a normal distribution [84], we can calculate the confidence intervals as:

$$\hat{p} \pm z_{\alpha/2}\sqrt{\hat{p}(1-\hat{p})/n}, \tag{4.19}$$

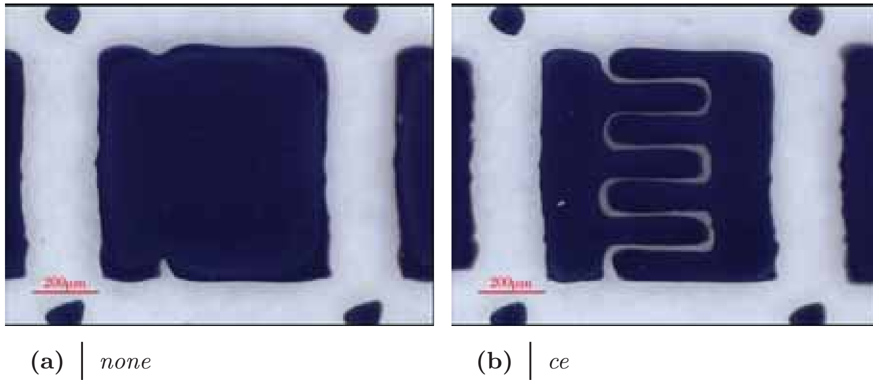
where $\hat{p}$ is the proportion of successes, n is the number of samples, and $z_{\alpha/2}$ is the normal percentile for an α confidence. This interval approximation does not behave correctly when $\hat{p}$ is close to 0 or 1. To compensate this effect, Agresti et al. [85], and Bonett et al. [86] propose some adjustments. Then, the interval is defined as:

$$\tilde{p} \pm \sqrt{\frac{1}{\tilde{n}}\tilde{p}(1-\tilde{p})} \tag{4.20}$$

where $\tilde{n} = n + z^2_{\alpha/2}$, $\tilde{p} = \frac{1}{\tilde{n}}\left(X + \frac{1}{2}z^2_{\alpha/2}\right)$, and X is the number of successes.

Table 4.3 has the calculated $\alpha = 0.95$ confidence intervals for some interesting finger widths and separations. We can observe that even with a 20 µm separation the yield improves around 50%, and by making the fingers 120 µm wide, the number of working structures raise to 98%.

As an example of the improvement achieved with the compensation, Fig. 4.21 shows



(a) | *none* **(b)** | *ce*

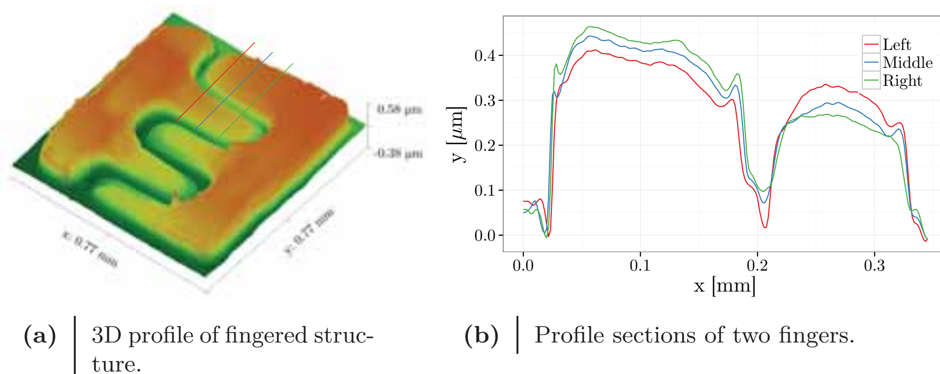
Figure 4.21 | Resulting finger structures without and with compensation.

microscope captures of two finger structures, with and without compensation. The images show the compensation effects on samples printed with the minimum width and separation, and appearing in a close location on the foil, therefore reducing the risk of some isolated manufacturing issues to cause the contact of the fingers. It is evident that only by applying the compensation we are able to obtain non short-circuited fingers.

Lastly, Fig. 4.22 shows an interferometer 3D profile of the whole structure, and the resulting profile on three different finger sections. Fig. 4.22a shows some difference in ink accumulation on the different fingers, having the most ink volume in the finger contacts. This is caused mainly because bigger area regions attract ink due to fluidic effects and ink coalescence [87, 88]. The profile shown on 4.22b, taken on different points on the finger, shows a difference in height, although not exceeding 100 nm and being small compared to the total finger height.

4.3 Conclusions

Starting from the necessity of having an understanding of the printing procedures this chapter proposes a full characterization methodology to extract key process and technology parameters. Therefore, knowing those parameters allows the creation of a set of FEC techniques, which ensure that the final fabricated design resembles the intended one as much as possible. These techniques are the inkjet PE equivalent for the traditional RET, such as OPC and ILT in traditional microelectronics. Hence, the proposed characterization procedure starts with the initial approach presented on [35] and adds automated result extraction, allowing to perform statistical analysis



(a) 3D profile of fingered structure.

(b) Profile sections of two fingers.

Figure 4.22 3D interferometer profile and finger section cut for a compensated fingered structure. The three cuts correspond to different positions inside the structure.

on the obtained data.

During the optical image conditioning processes to extract we obtained extra information about the printing process: the sample was not aligned properly with the template image, and there appeared a large amount of satellite droplets not caused by the compensation under study, but to the printhead. Consequently, when correcting the captured image, we extracted information about misalignment and number of satellite drops.

The misalignment information is calculated on a whole sample, therefore it indicates a possible substrate deformation during the manufacturing process. We saw that it exists some substrate dilation when printing Sunchemical EMD5603 silver ink on top of flexible PEN substrate. Modelling this behavior provides information useful on future process control, and possible corrections to compensate this deformation depending on the foil position.

Regarding the number of removed satellite droplets, this information is very useful as a global indicator of the overall printing process quality. We demonstrated that with this analysis we can detect if there are some printing anomalies, but due to the information gathered during manufacturing, we still cannot conclude which is the real cause of these printing imperfections. The origin cause remains elusive, but due to previous experimental observations we can hypothesize that a malfunctioning nozzle (being temporally or permanently clogged, or nozzles with non-matching ejection velocities) could cause some of these imperfections.

The characterization procedure based on a semiautomated probe station [65, 64]

allowed the mass-characterization of more than half a million printed samples without requiring high manual intervention, therefore obtaining statistical significant results. Specifically for the used silver nanoparticles ink on top of flexible PEN substrate using industrial D-class DPN 10 pL printheads, the results have been consistent along different foils printed on different runs spanning several days. Furthermore, the best compensation obtained have been successfully verified using transistor finger structures performing electrical continuity tests. The results show an outstanding improvement on shape morphologies, obtaining detailed fingered structures without electrical short circuits. This is of relevant importance in the fabrication of devices such as capacitors or transistors where interdigitated structures are used, because the printed shape is critical for device performance, and the whole process stability. The proposed compensation improves shape morphology allowing scaling down, thus improving resolution and process yield.

Automatic Compensations: The *Layout2Bitmap* tape-out framework | 5

After characterizing ink behaviour and extracting which are the appropriate compensation strategies, rules, and patterns for the specific process, we need to move them into the actual automatic tape-out processes to obtain directly printable designs.

Therefore, for our technology setup, based in a fully inkjet printing process using a Dimatix printer, we need some framework which converts typical CAD formats to a set of bitmap files, which will define each drop final position.

5.1 *Layout2Bitmap*

The *Layout2Bitmap*, from now on *L2B*, is the responsible of reading the design and generating a set of bitmaps for each printing layer. For each layer, or layer region, it will apply the needed compensations according to the technology description.

5.1.1 Tool description

L2B is the tool which performs the conversions from final layout designs to directly printable bitmap files. Initially the conversion was just devoted to the discretization

of the design to a bitmap having the printer resolution, but following this thesis evolution we implemented several different options and optimizations. Currently *L2B* incorporates most of the compensation results in terms of patterns and rules obtained from the previous chapter results.

Therefore, being able to use the *L2B* tool since the start of the project and evolving it according to characterization results, allowed its use in the characterization procedures. This helped both in bug solving, and in speeding the tape-out for experiment generation.

The procedures implemented by *L2B* to perform these automatic compensations and conversions are succinctly: Loading the design, reading the appropriate compensations, performing all the optimizations, and generating the final set of printable files. All of these steps are described thoroughly along the following sections.

5.1.2 Design loading

In order to export the design, it is necessary a format which is widely supported across microelectronics CAD tools, while allowing some customization or extensibility, thus enabling adding custom technology information.

After the evaluation of several alternatives, we chose to use the GDS stream format [89]. This format originally developed by Calma in 1970, was used for controlling integrated circuit photomask plotting. From then, it became a *de facto* industry standard for data exchange at layout level. It is used often as the last step in the design physical cycle, acting as a foundry interface format.

There is a newer alternative, the successor of the GDS format Open Artwork System Interchange Standard (OASIS), which is maintained by SEMI international standards association. Although it presents some improvements over GDS, such as smaller file size and new primitive types, the format specification is private and there are more tools supporting GDS rather than OASIS.

GDS File format

The GDS stream format is a binary format which describes a library (in our case a design), which consists in several structures (i.e. cells). Each structure contains several primitive elements, or references to other cells, allowing hierarchical designs. The file also contains some metadata, such as creation and modification times, for the library and the structures.

The available element types supported by this specification are:

Boundary contains a list of point coordinates. Each point is the vertex of the defined polygon. Fig. 5.1a shows the polygon for points $P_1, P_2, \cdots, P_{10}$.

Path contains a list of points that indicate a path. Its required properties are the offset width and the path type. Fig. 5.1b shows an example path with points P_1, P_2, P_3, P_4.

Text contains a string of text. Although its parameters specify position and font type, this element does not have any graphical representation.

Box defines a box using three coordinates. Fig. 5.1c shows a box represented with coordinates P_1, P_2, P_3.

SREF contains a structure reference. This construct allows a hierarchical library definition, allowing nested instances of structures.

AREF contains an array reference. Though similar to the SREF type, the AREF is used to place the instances in a regular matrix form.

In addition to the unique attributes each element has, they also have the layer and datatype specification. These two identifiers, defined on the technology information, act as a reference to the actual layer this element belongs to.

At present moment, the *L2B* tool only accepts non-hierarchical (i.e. flattened) design files. Therefore, both SREF and AREF element types are ignored. For this reason, *L2B* only accepts GDS files with one structure defined. In addition, text elements are ignored, because are only for annotation purposes and have no graphical meaning. These limitations do not affect the *L2B* tool usefulness, as all current EDA tools support exporting flat GDS files.

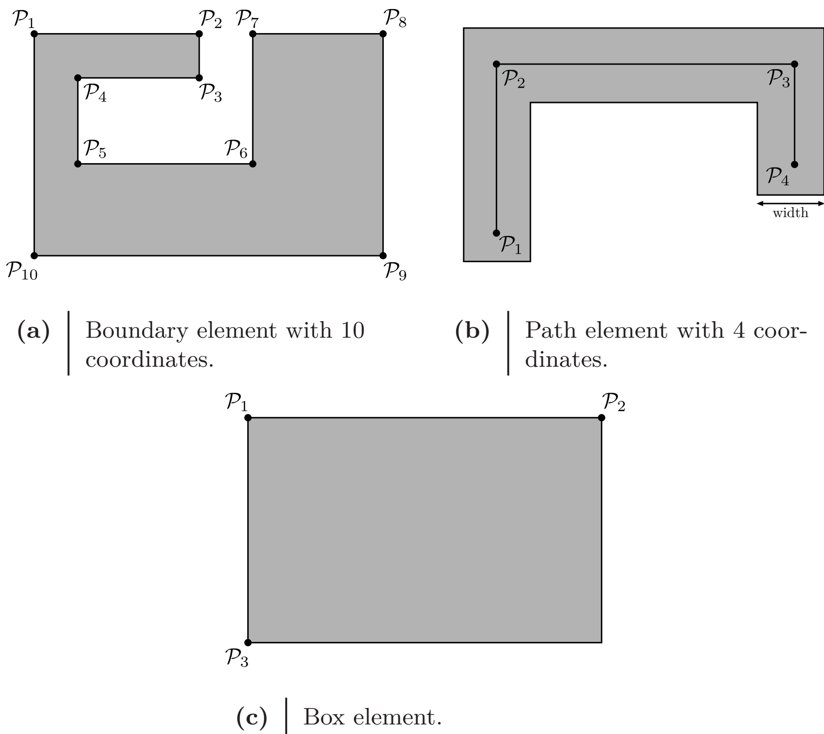
5.1.3 Design compensation

In order to apply the needed compensations, the *L2B* tool works directly with the polygons defined on the GDS file. Therefore we need some conventions about how to represent the shapes.

All the primitives from the GDS file are directly translated to a list of points, specifying the boundary of the polygon. The inner face of the contour always lies to the left side of the edges. Therefore, on Fig. 5.2 we can observe the two different representations for clockwise and counter-clockwise vertices order.

As for polygons, there are three types:

Simple: Polygons whose edges do not intersect nor overlap (seen on Fig. 5.3a).

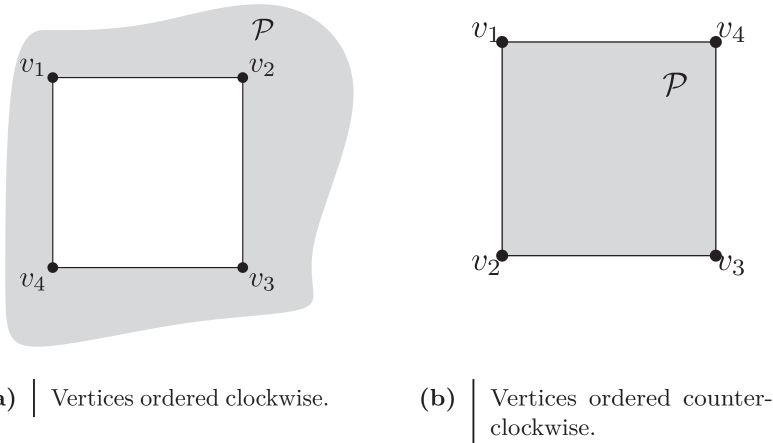


(a) Boundary element with 10 coordinates.

(b) Path element with 4 coordinates.

(c) Box element.

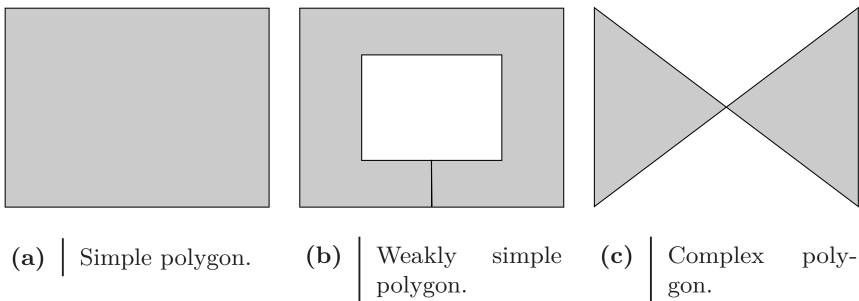
Figure 5.1 Available primitive element types defined in GDS specification.



(a) Vertices ordered clockwise.

(b) Vertices ordered counter-clockwise.

Figure 5.2 Representation of the two available polygon boundaries.



(a) Simple polygon.

(b) Weakly simple polygon.

(c) Complex polygon.

Figure 5.3 Representation of the two available polygon boundaries.

Weakly simple: Polygons whose edges do not intersect but overlap (seen on Fig. 5.3b).

Complex: Polygons whose edges intersect (seen on Fig. 5.3c).

Most CAD tools output is formed by simple polygons, thus simplifying the methods and algorithms needed.

Lastly, by having these assumptions, we can generalize the polygon representation creating polygon with holes. A polygon with holes is represented combining simple

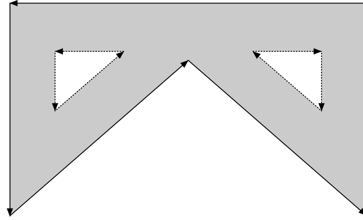


Figure 5.4 | Polygon with holes example.

boundaries with different orientations:

Outer boundary is the polygon that defines the outer limit of the polygon with holes. It is oriented counter-clockwise, as shown on 5.2b.

Holes are several polygons indicating holes. The orientation is the inverse from the outer boundary, as shown on Fig. 5.2a.

Fig. 5.4 contains an example of a polygon with two holes, together with the edge directions.

Geometrical boolean operations

Considering that polygons represent a division of the space, we can define a regularized set of operations on them. Therefore, seeing the polygons as mathematical sets, we can define the set boolean operations on them. On Fig. 5.5 we have the graphical representation of the defined operations between polygons $\mathcal{P}_1$ and $\mathcal{P}_2$.

Intersection: The resulting polygon shown on figure 5.5a, $\mathcal{P}_1 \cap \mathcal{P}_2$ contains the region that belongs to $\mathcal{P}_1$ AND $\mathcal{P}_2$.

Union: The resulting polygon shown on figure 5.5b, $\mathcal{P}_1 \cup \mathcal{P}_2$ contains the region that belongs to $\mathcal{P}_1$ OR $\mathcal{P}_2$.

Difference: The resulting polygon shown on figure 5.5a, $\mathcal{P}_1 \backslash \mathcal{P}_2$ contains the region that belongs to $\mathcal{P}_1$ but not to $\mathcal{P}_2$.

Symmetric difference: The resulting polygon shown on figure 5.5d, $(\mathcal{P}_1 \backslash \mathcal{P}_2) \cap (\mathcal{P}_2 \backslash \mathcal{P}_1) = \mathcal{P}_1 \oplus \mathcal{P}_2$ contains the region that belong to $\mathcal{P}_1$ or $\mathcal{P}_2$ but not to both.

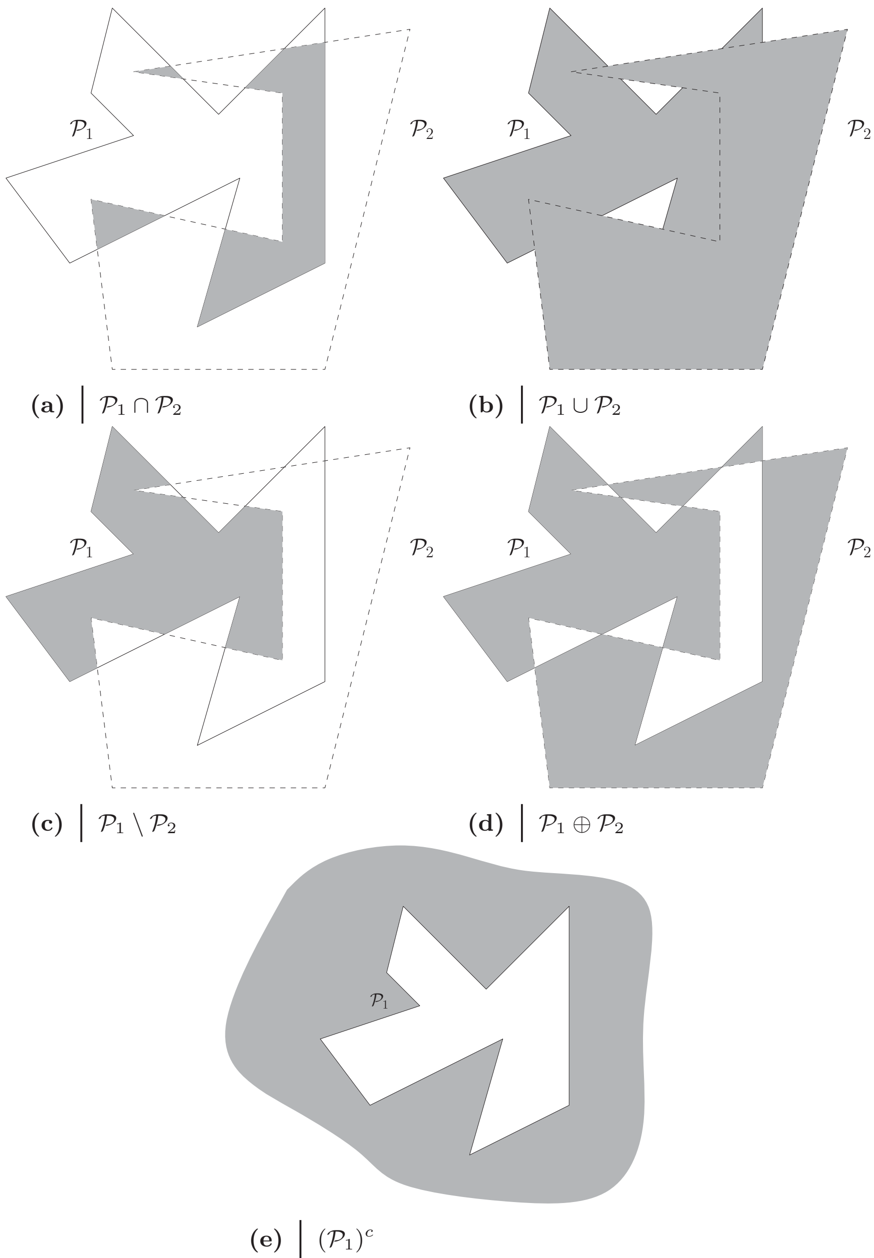


Figure 5.5 | Regularized set of boolean operations on polygons.

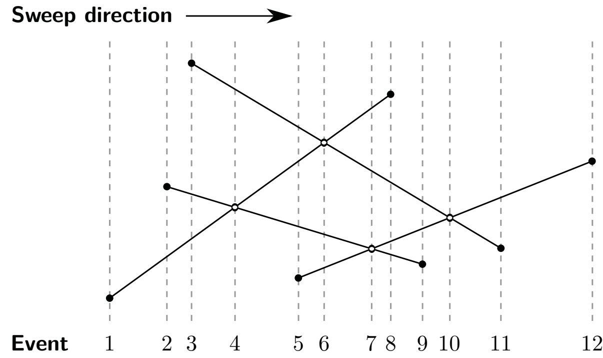


Figure 5.6 | Sweep line approach to segment intersections.

Complement: Also known as the inverse, the resulting polygon shown on figure 5.5e, $(\mathcal{P}_1)^c$ contains the region that does not belong to $\mathcal{P}_1$. This operation can lead to unbounded polygons.

By applying and combining all these operations between layers defined in a whole layout, we can extract the different regions of interest. In the context of EDA tools, the boolean operations are widely used, such as device and parasitics extraction or performing DRC checks. Therefore, L2B uses these operations to extract the areas of interest which will be compensated.

One of the most common algorithms to perform boolean operations rely on the sweep line approach [90]. This common approach in computational geometry exploits the locality of the problem, and is based on a conceptual line (vertical or horizontal) which is swept across the plane stopping at specific points or *events*. Then, the only objects being processed are the ones intersecting this imaginary line, or the ones in the close vicinity, thus reducing considerably the algorithm complexity. The algorithm completes once the line has passed over all objects. Fig. 5.6 shows an example sweep line to calculate segment intersections. Considering that a planar subdivision of the space can represent a polygon, we can obtain geometrical boolean operations calculating an overlay between two polygons, and computing the intersections between their edges [90–94]. These algorithms are all based on the intersection finding algorithm by Shamos and Hoey in [95] using the sweep line approach.

The overlay between two subdivisions $\mathcal{S}_1$, and $\mathcal{S}_2$ is the subdivision $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$. There is a face in $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$ if and only if there are faces f_1 in $\mathcal{S}_1$, and f_2 in $\mathcal{S}_2$ [90, 96]. This means that $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$ is the subdivision of the plane induced by the edges of

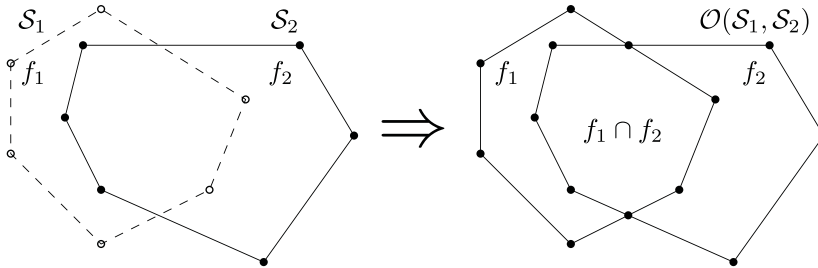


Figure 5.7 | Overlay of two planar subdivisions.

$\mathcal{S}_1$ and $\mathcal{S}_2$, as illustrates Fig. 5.7.

Every planar subdivision, in addition to edges and vertices, has its bounded faces labeled. Hence, the resulting boolean operation corresponds to the boolean operation on the face labels. For example, to calculate $\mathcal{S}_1 \cap \mathcal{S}_2$ the faces extracted will correspond to the labels f_1 and f_2.

The complement is a special case, because there is no need of applying these algorithms. The vertices ordering represent the bounded face, so we can obtain the complement by reversing the vertex list.

The current implementation of the polygon boolean operations set inside *L2B* uses the Vatti algorithm [97, 98]. This implementation provides a numerically robust algorithm towards possible degeneracies [99], and supports the calculation of intersections with integer coordinates, but has a complex implementation. In addition, it is an efficient algorithm, capable of working with large number of polygons, therefore is able to process large layouts.

Polygon offsetting

The other main geometrical operation needed for compensations is polygon offsetting. With this operation *L2B* is able to grow and shrink the shapes while maintaining the topology of the polygon, as illustrates Fig. 5.8. The figure shows the original shape, outlined with a solid line, and an interior and exterior offsets, outlined with dashed lines.

This operation is also widely used on CAD tools , mainly on Computer Aided Manufacturing (CAM) programs, for example on creating tool-paths for pocket machining . In the compensations case, this type of operations are useful to remove

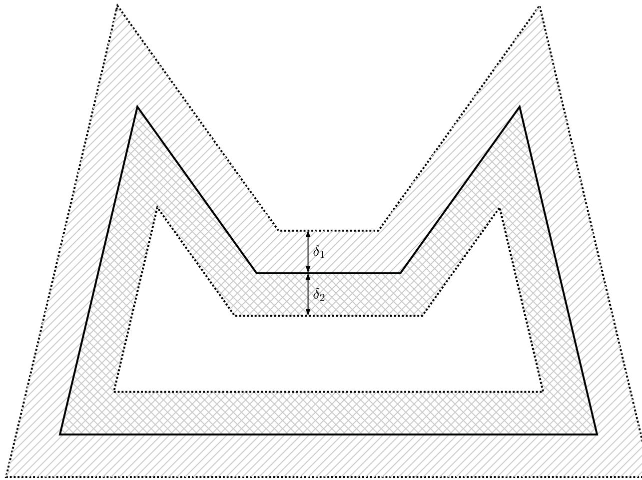


Figure 5.8 Polygon offsetting examples. The shaded region corresponds to the amount offseted (δ_1 for exterior offset, δ_2 for interior offset).

regions with an specified area or correct ink spreading effects.

There are several alternative algorithms to perform polygon offsetting. Because the *L2B* tool works on the vector level and not on the image or raster level, the complexity of offsetting increases, although gives more flexibility for further processing.

One of the alternatives, presented on [100] consists on the calculation of the offset using a Voronoi diagram, but the offset contains rounded corners. In [101, 102], there is an implementation of the straight skeleton. This novel geometric structure is a topological skeleton which contains only straight lines. It is defined by the continuous shrinking of the edges at the same speed. The main drawbacks to this algorithm is the high complexity of processing the edge events (i.e. edge collapsing and polygon split).

Considering that *L2B* tool contains a boolean processing library aware of winding numbers, we can use them to calculate offset curves [103]. The winding number is defined as follows. Let $\mathcal{P}$ be a polygon, q any point not belonging to $\mathcal{P}$, and R any ray from infinity to q not crossing any vertex of $\mathcal{P}$. Then, the winding number $\omega(R, \mathcal{P})$ is:

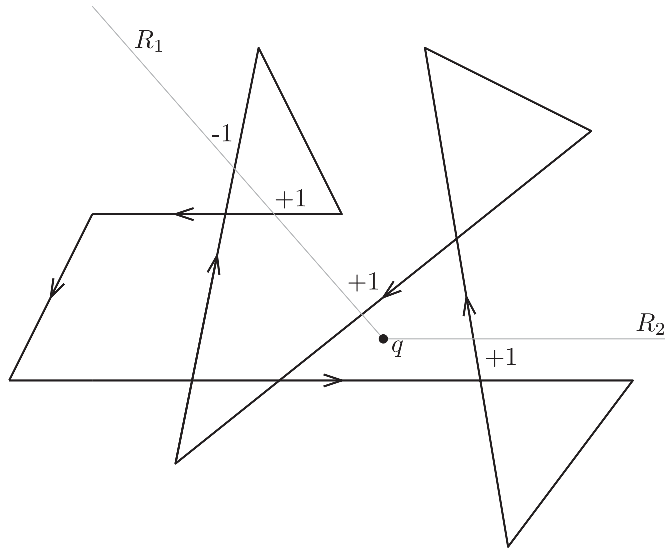


Figure 5.9 Winding number calculation for region containing point q. The resulting number is the same regarding which ray is shot.

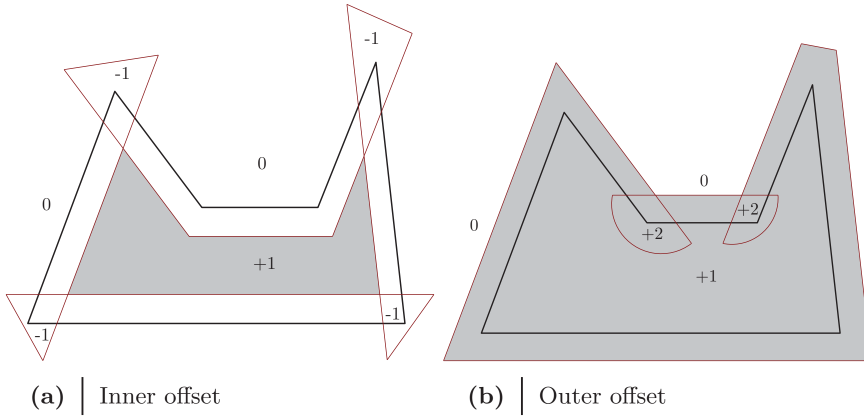
$$\omega(R, \mathcal{P}) = \sum_{e_i \in \mathcal{P}} \Psi(R, e_i) \tag{5.1}$$

where for each edge e_i, $\Psi(R, e_i)$ is:

$$\Psi(R, e_i) = \begin{cases} 0 & \text{if } R \text{ does not intersect} e_i \\ -1 & \text{if } e_i \text{ crosses } R \text{ in CCW direction from } q \\ 1 & \text{if } e_i \text{ crosses } R \text{ in CW direction from } q \end{cases} \tag{5.2}$$

Therefore, the winding number for a region is the number of counterclockwise turns that an object tracing the contour does around the point q. Fig. 5.9 contains a graphical representation of the winding number calculation for a point q with two different rays R_1 and R_2. The resulting number is independent of the ray.

First the algorithm constructs the raw offset curve. This curve is calculated from the original polygon, offsetting each edge (opposite edge normal for inner offsets and along edge normal for outer offsets) and connecting disjoint segments. After the raw offset curve is defined, *L2B* boolean functions process the curve and discard



(a) | Inner offset

(b) | Outer offset

Figure 5.10 | Offset curves calculation. The shaded region corresponds to the final calculated shape.

all the regions where $\omega(R, \mathcal{P}) > 0$, that is a positive winding number fill rule. The resulting region will be the offset curve.

Bitmap discretization

Both printers used to deposit materials in this thesis, the Dimatix DMP-2831 and the DMP-3000 Materials Printer support several input formats as a design input. The provided utilities contain converters for GDS, OASIS, and Drawing Interchange Format (DXF) formats to 1bit depth bitmap files. Using those conversion programs though limit the possibilities for manipulating the design (i.e. we cannot control ink density depending on region). So, in order to have as much control as possible of the printing process, *L2B* tool needs to generate bitmap files.

The bitmap file format [104] is a raster graphics format used to store digital images. It is a very simple format, which can store two-dimensional image data. The bitmap file format the printer uses has to be monochrome, having 1bit per pixel, and for the printer, each pixel corresponds to ejecting a drop on that particular location.

Because the deposited drop diameter is much higher than the printer drop spacing, it is possible to skip the ejection of some drops without producing holes on the printed shape. Therefore we can control the amount of ink deposited on each region by printing a pixel pattern instead of filling completely the polygon. Hence, depending on the region, we can reduce the ink quantity and reduce bulging or obtain a smoother surface.

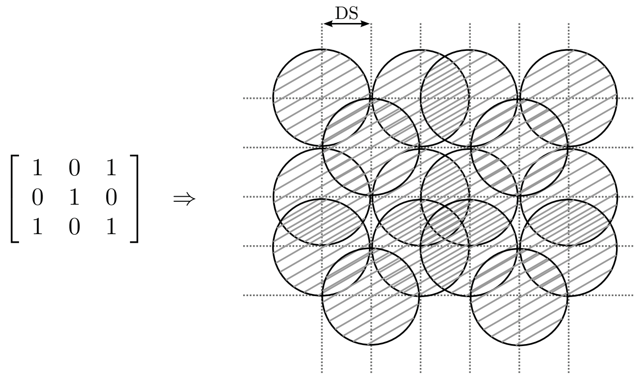


Figure 5.11 | Pattern matrix definition and resulting drop positions on the grid.

A pattern is defined as a binary matrix, indicating whether a drop should be ejected or not. This pattern is repeated along the x and y axis filling the whole polygon. Fig. 5.11 contains an example pattern and the resulting filling.

In order to efficiently fill all the layout with the appropriate pattern, *L2B* uses a sweep line filling algorithm [95, 105] as discussed earlier. To correctly fill the polygons, a scan line is swept from top to bottom, and the events are regularly generated at each grid location. Then, all intersecting points with the edges are added to the queue, ordered in lexicographic xy-ordering and the regions are filled using an odd-even fill rule. Fig. 5.12 shows the process applied to an arbitrary shape.The odd-even rule states that a point is inside a polygon when the winding number $\omega(R, \mathcal{P})$ is odd, and therefore has to be filled. This holds for all simple polygons, which is the case after all the processing.

Compensations application

Wrapping all the different operations described on this chapter, the *L2B* tool operation following the flow chart on Fig. 5.13:

1. Read technology information database, extracting layer information, graphical properties, and compensation operations definition.

2. Read the design from the GDS file.

3. For each compensation on the database:

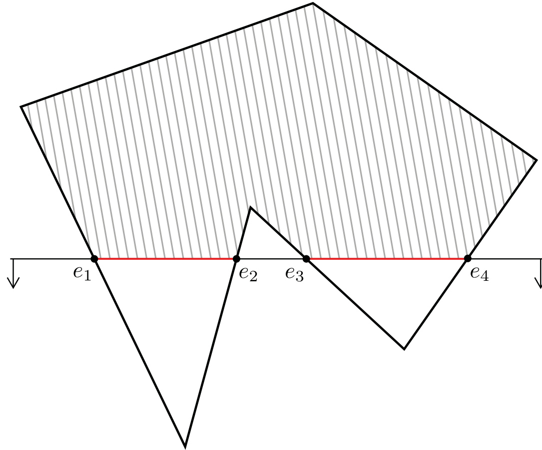


Figure 5.12 Sweep line fill procedure. The shaded area is already filled, and the red highlighted scanline is the active region.

(a) Execute the specified boolean operation

(b) If the defined drop diameter is not 0, shrink to compensate bulging effects.

(c) Scanline-fill the resulting polygons.

4. After applying all the compensation operations, generate one BMP file for each printing step.

5.2 *Layout2Bitmap* into design flows

The overall architecture for *L2B*, on Fig. 5.14, is built in a modular way. All the central operations belong to the *L2B* library, which wraps all the different components (i.e. the geometry engine, the scanfill library and the bitmap I/O functions), and the plugin engine.

Having this modular architecture allows the integration of this EDA component inside different design flows, just by wrapping the *L2B* library into a specific tool. This section contains the description of this possible integration with the two available design flows, the one implemented with the set of Free/OSS tools, and the implemented using the commercial EDA package *MaskEngineer*.

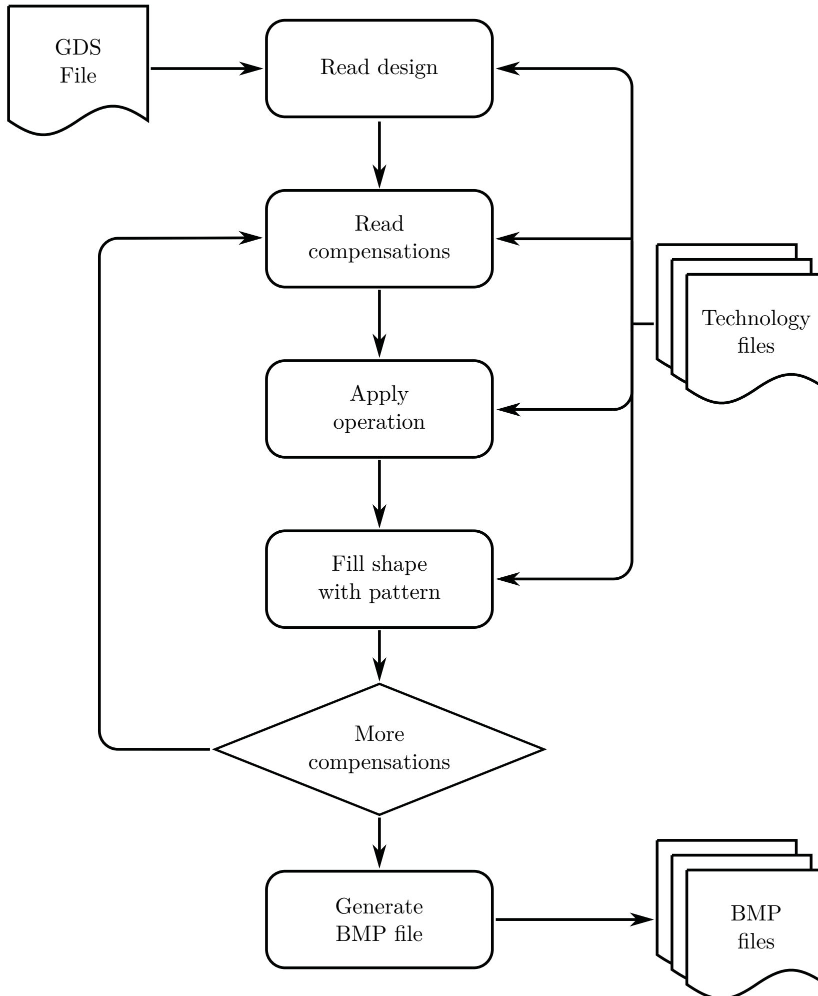
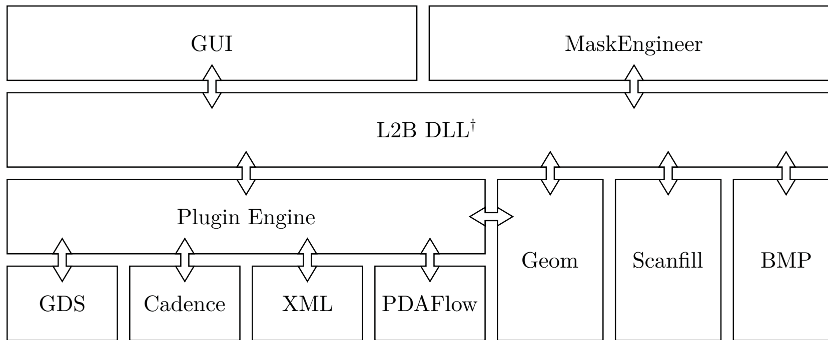


Figure 5.13 | Flow diagram of the general *L2B* operation.



†Contains the compensation algorithm implementation.

Figure 5.14 | General structure of the *L2B* framework.

5.2.1 Free/OSS flow integration: Glade

The design conversion to printable files takes place on the last part of the design process. Therefore it makes sense to integrate the layout compensation on the layout design tool, so when the design is complete it is possible to generate the files for fabrication.

Because Glade is not open source, we do not have the possibility to modify it to link with the *L2B* libraries, hence we will use the Glade script interface to call the *L2B* Graphical User Interface (GUI) program. Glade configuration files will create a new custom menu entry which will perform all the tape-out processes.

Because of the assumptions done when loading the designs, we need to pre-process the final design by flattening all the cells and exporting a GDS with the top level structure. After performing this pre-processing, the script launches the *L2B* GUI passing the needed parameters (input file and technology location), and the compensated BMPs are created.

5.2.2 Commercial flow integration: *MaskEngineer*

In the case of the integration of the *L2B* framework into the commercial flow, we achieved a seamless integration, due to the partnership inside the TDK4PE project.

We achieve the integration by compiling in the *L2B* library into the *MaskEngineer* executable, and just by creating a specific plug-in to obtain the final design. This

integration did not require any modification of the core functionality except for the plug-in. *MaskEngineer* supports reading the technology information from the same XML database as *L2B*, thus not requiring any change.

Because compensations are process dependent, the actual calls to the *L2B* functions appear on the design kit creation. According to PDAFlow manual [106, 59], the foundry definition file contains the function `export(string file)` which creates the final mask designs for the design. In our case, instead of masks, this function creates the set of BMP files. Listing 5.1 contains the full commented code.

```
function export(string file) {
  // Call the process() function for each layer
  string all_output = process_design2output("all_output");

  // Call the export and compensation functions of L2B
  mask::exportToBMP(all_output, file);
  return 1;
}

// And then, on the final top-level design
pdk::pdkExport("directory/to/export/to",
               "Processing");
```

Listing 5.1 Full listing of the export function inside the design kit definition for PDAFlow design kits, and its usage.

5.3 Results and conclusions

L2B provides an extensible framework for implementing automated compensations. Being able to characterize and extract the needed compensation operations for a given technology process (see chapter 4), provides the final element needed to close the gap between designers and technology developers.

The framework architecture allows an easy integration into different design flows covered by different tools. As a demonstration, *L2B* has been integrated into two different design flows, one implemented using Free/OSS tools, and the other using a commercial package.

In addition *L2B* can perform different compensations, due to the full set of computational geometry algorithms it implements. Therefore, once we can extract the necessary operations for a specific technology process, they can be easily implemented.

As the result of its plug-in engine, the *L2B* framework can be extended to other input file formats without modifying the core functionality. This has been demonstrated by writing a plug-in capable of reading the information directly from the *MaskEngineer* mask format.

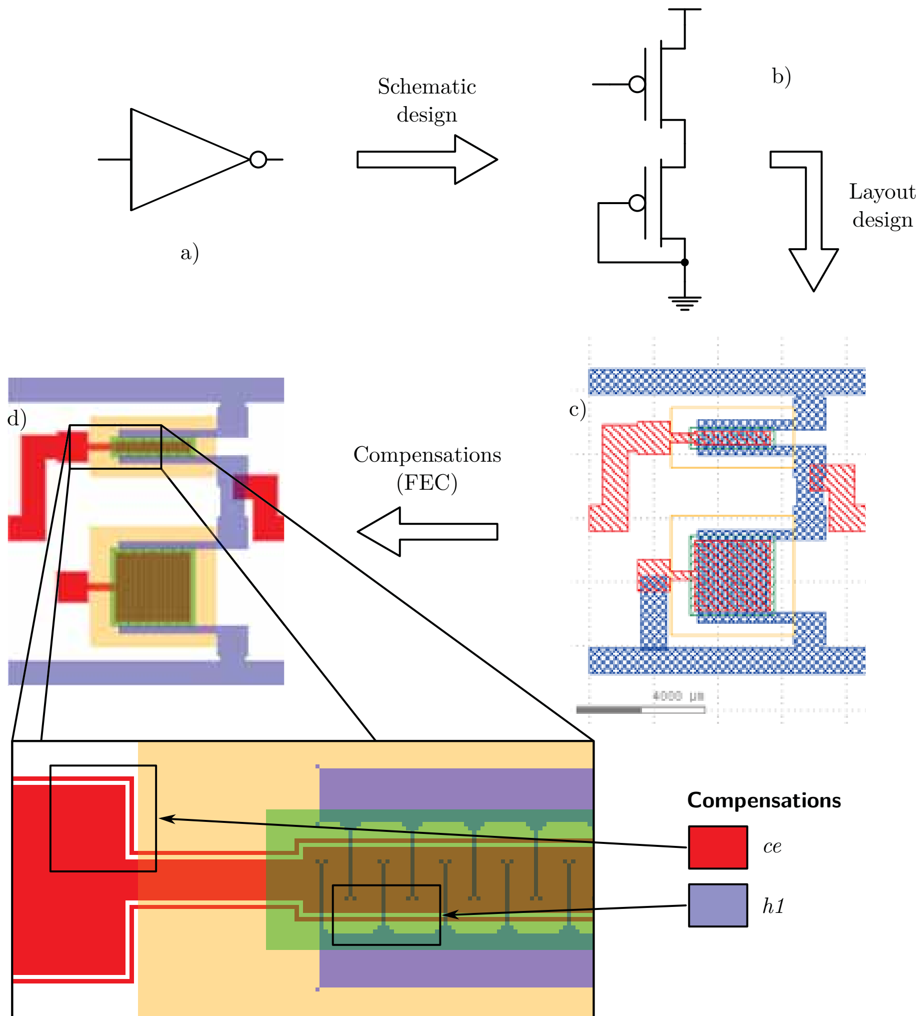


Figure 5.15 Schematic to generated bitmaps process: a) symbol, b) schematic, c) layout, d) set of generated bitmaps, and e) close view on performed compensations. *Layout2Bitmap* processes the layout on c) to generate a set of masks on d).

Global conclusions and further activities | 6

The work presented on this thesis intended to supply a design methodology and a complete full-custom design kit adapted to inkjet PE technologies. In addition, to fill the gap between circuit designers and technology developers, this thesis sought to provide a framework responsible of performing the tape-out procedure along with all necessary corrections, and convert the design to a set of files understandable by the printer equipment.

The proposed XML database and its access and manipulation API, described on chapters 2, and 3, achieves the decoupling of the technology information from the used EDA tools, therefore removing the dependence traditionally appearing between a PDK and a set of EDA tools: any changes or updates of the stored information on the XML database are automatically reflected on all supported EDA tools, and once a different EDA tool is supported it will be able to access the basic infomation of all PDKs. To further demonstrate the feasibility of this approach, we managed to create two design kits for two different technologies (one gravure based, the other using lithography processes) supported by two different EDA tool sets from this unique representation, and with less than two weeks of technical work, mostly spent in developing specific PCells for each different foundry.

Chapter 4 describes in detail the compensation characterization methodologies, and provides important insights on the analysis of the deposited ink behavior. Together with a semi-automated setup, we have been able to characterize and analyze more than half a milion samples. This setup has provided the extraction of key technology information:

- The calculation and quantification of the substrate deformation amount across a whole printed foil.

- An image analysis procedure that indicates the overall printing and process quality by analyzing the number of satellite drops.
- Establishment of a metric useful to compare the fidelity between the intended design and the final printed result. This score has been sucessfully used to evaluate different compensation operations which improve the printed results, and extract which correction outperforms the rest. After an initial analysis, several more designs have been used to evaluate the best compensation in a fingered structure, commonly appearing on the construction of OTFT devices, achieving a significant improvement and allowing scaling down the device, and therefore improving the performance.

All the extracted process and compensation results have been tested across several different foils, and printed across different batches, obtaining a consistent result between them. This confirms the robustness of both the characterization procedure, and the extracted results.

Lastly, chapter 5 describes the *L2B* framework, which performs the tape-out procedure applying all necessary corrections. By using this tool we finally close the gap between design and fabrication, abstracting this process. This extensible framework contains an implementation of the extracted compensation operations from chapter 4, therefore a designer can use it to generate direclty printable files without an in-depth knowledge of the underlying technology. This framework, used along the whole TDK4PE project to help with technology development and improvement, has been integrated into two different EDA tool sets, one of them commercial. In addition to the usage in the TDK4PE project, *L2B* has been downloaded by a group in University of Virginia.

In summary, the main objectives proposed in this thesis have been achieved. By building this complete PDK together with the characterization methodology, and the post-layout manipulation tool, we have obtained a direct path from design to a set of manufacturable files, automating and reducing the knowledge required to create files understandable by the printer for a fixed technology. Nevertheless, there are still open issues which need to be dealt with.

The brief conclusions appearing on each chapter end do not contain any reference to future work derived from the topics presented on this thesis. Taking the test structures developed on chapter 4, and using the semi-automated characterization setup and analysis procedures, we could create some process control structures. By including them in every manufactured foil, we would control the fabrication process quality.

To continue the work in compensation operations, both the analysis presented on chapter 4, and the framework on chapter 5 could be enhanced using simple

numerical methods, as the work of Soltman et al. [24, 30]. In addition, because of the sucess in printing very detailed fingered structures using industrial printing printheads, the next step should be build the full OTFT stack and perform electrical measurements. This work focused on the analysis and compensation of a specific material deposited on a specific substrate, therefore a next logical step would be to apply the same methodology to other materials over different sufraces, thus obtaining the compensation operations for the full layer stack of a technology.

There is a lot of room for improvement of the framework presented on chapter 5. Extending the current compensation procedures taking into account the substrate deformation could lead to significant improvements in foil fabrications. By modelling the deformation induced in each manufacturing step, the final misalignment between layers could be reduced.

All in all, this work is a significant foundation for the current status of PDKs and specific tools for PE technologies, but it needs still more development. Although a the results presented on this thesis provide a complete, direct, and automated path from design to manufacturing for PE processes, there is still room for improvement, characterizing new materials and integrating new compensation methods. Although by having a set of EDA, PDKs, and semi-automated characterization equipment, the efforts needed to push the technology have been greatly reduced.

References

[1] S. R. Forrest. *The path to ubiquitous and low-cost organic electronic appliances on plastic.* Nature, 428(6986):911–918, 2004.

[2] S. L. Swisher, M. C. Lin, A. Liao, E. J. Leeflang, Y. Khan, F. J. Pavinatto, K. Mann, A. Naujokas, D. Young, S. Roy et al. *Impedance sensing device enables early detection of pressure ulcers in vivo.* Nature communications, 6, 2015.

[3] R.-Z. Li, A. Hu, T. Zhang and K. D. Oakes. *Direct writing on paper of foldable capacitive touch pads with silver nanowire inks.* ACS applied materials & interfaces, 6(23):21721–21729, 2014.

[4] Uniqarta. *Ubiquitous, unobtrusive electronics.* Technical report, August 2015. URL `http://www.uniqarta.com/uploads/3/3/4/5/3345725/uniqarta_white_paper_20150826.pdf`.

[5] C. Sekine, Y. Tsubata, T. Yamada, M. Kitano and S. Doi. *Recent progress of high performance polymer oled and opv materials for organic printed electronics.* Science and Technology of Advanced Materials, 15(3):034203, 2014.

[6] L. Grande, V. T. Chundi, D. Wei, C. Bower, P. Andrew and T. Ryhänen. *Graphene for energy harvesting/storage devices and printed electronics.* Particuology, 10(1):1 – 8, 2012. ISSN 1674-2001. doi:http://dx.doi.org/10.1016/j.partic.2011.12.001. URL `http://www.sciencedirect.com/science/article/pii/S1674200111002094`.

[7] R. Coehoorn, V. van Elsbergen and C. Verschuren. *High efficiency oleds for lighting applications*. In E. Cantatore, editor, *Applications of Organic and Printed Electronics*, Integrated Circuits and Systems, pages 83–100. Springer US, 2013. ISBN 978-1-4614-3159-6. doi:10.1007/978-1-4614-3160-2_4. URL `http://dx.doi.org/10.1007/978-1-4614-3160-2_4`.

[8] J. R. Castrejon-Pita, W. Baxter, J. Morgan, S. Temple, G. Martin and I. Hutchings. *Future, opportunities and challenges of inkjet technologies*. Atomization and Sprays, 23(6), 2013.

[9] I. M. Hutchings and G. D. Martin. *Inkjet technology for digital fabrication*. John Wiley & Sons, 2012.

[10] T. Shield, D. Bogy and F. E. Talke. *Drop formation by dod ink-jet nozzles: A comparison of experiment and numerical simulation*. IBM Journal of Research and Development, 31(1):96–110, 1987.

[11] J. Zaumseil and H. Sirringhaus. *Electron and ambipolar transport in organic field-effect transistors*. Chemical reviews, 107(4):1296–1323, 2007.

[12] H. Klauk. *Organic circuits on flexible substrates*. In *IEEE InternationalElectron Devices Meeting, 2005. IEDM Technical Digest*. 2005.

[13] G. Schmidt, M. Bellmann, B. Meier, M. Hambsch, K. Reuter, H. Kempa and A. Hübler. *Modified mass printing technique for the realization of source/drain electrodes with high resolution*. Organic Electronics, 2010.

[14] T.-C. Huang, K. Fukuda, C.-M. Lo, Y.-H. Yeh, T. Sekitani, T. Someya and K.-T. Cheng. *Pseudo-cmos: A design style for low-cost and robust flexible electronics*. Electron Devices, IEEE Transactions on, 58(1):141–150, 2011.

[15] E. C. Smits, S. G. Mathijssen, P. A. Van Hal, S. Setayesh, T. C. Geuns, K. A. Mutsaers, E. Cantatore, H. J. Wondergem, O. Werzer, R. Resel et al. *Bottom-up organic integrated circuits*. Nature, 455(7215):956–959, 2008.

[16] C. Mead and L. Conway. *Introduction to VLSI systems*, volume 802. Addison-Wesley Reading, MA, 1980.

[17] C. A. Mead. *Vlsi and technological innovation*. 1979.

[18] U. of Minnesota VLSI Group. *Organic Process Design Kit (OPDK)*. URL `opdk.umn.edu`, consulted on 2015-07-06.

[19] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh et al. *Freepdk: An open-source variation-aware design kit*. In *Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on*, pages 173–174. IEEE, 2007.

[20] Si2 Foundation. *Si2: Project Info - OPDKC Specifications*. URL https://www.si2.org/openeda.si2.org/projects/opdkcspec/, consulted on 2015-07-06.

[21] Si2 Foundation. *OpenPDK Coalition*. URL https://www.si2.org/opdkc_index.php, consulted on 2015-07-06.

[22] A. de la Fuente Vornbrock, D. Sung, H. Kang, R. Kitsomboonloha and V. Subramanian. *Fully gravure and ink-jet printed high speed pBTTT organic thin film transistors*. Organic Electronics, 11(12):2037–2044, December 2010. ISSN 15661199. doi:10.1016/j.orgel.2010.09.003. URL http://linkinghub.elsevier.com/retrieve/pii/S1566119910002971.

[23] T. Sekitani, Y. Noguchi, U. Zschieschang, H. Klauk and T. Someya. *Organic transistors manufactured using inkjet technology with subfemtoliter accuracy*. Proceedings of the National Academy of Sciences, 105(13):4976, 2008.

[24] D. B. Soltman and V. Subramanian. *Inkjet-printed line morphologies and temperature control of the coffee ring effect*. Langmuir : the ACS journal of surfaces and colloids, 24(5):2224–31, March 2008. ISSN 0743-7463. doi:10.1021/la7026847. URL http://www.ncbi.nlm.nih.gov/pubmed/18197714.

[25] H.-Y. Tseng and V. Subramanian. *All inkjet-printed, fully self-aligned transistors for low-cost circuit applications*. Organic Electronics, 12(2):249–256, February 2011. ISSN 15661199. doi:10.1016/j.orgel.2010.11.013. URL http://linkinghub.elsevier.com/retrieve/pii/S156611991000371X.

[26] H. Kang, D. B. Soltman and V. Subramanian. *Hydrostatic optimization of inkjet-printed films*. Langmuir : the ACS journal of surfaces and colloids, 26(13):11568–73, July 2010. ISSN 1520-5827. doi:10.1021/la100822s. URL http://www.ncbi.nlm.nih.gov/pubmed/20408524.

[27] Z. N. Wang and Z. N. Tang. *Numerical simulation of droplet formation of piezoelectric ink-jet printing*. In *Advanced Materials Research*, volume 174, pages 191–194. Trans Tech Publ, 2011.

[28] H. Tan, E. Torniainen, D. P. Markel and R. N. K. Browning. *Numerical simulation of droplet ejection of thermal inkjet printheads*. International Journal for Numerical Methods in Fluids, 77(9):544–570, 2015. ISSN 1097-0363. doi:10.1002/fld.3997. URL http://dx.doi.org/10.1002/fld.3997.

[29] D. Siregar, J. Kuerten and C. van der Geld. *Numerical simulation of the drying of inkjet-printed droplets*. Journal of Colloid and Interface Science, 392:388 – 395, 2013. ISSN 0021-9797. doi:http://dx.doi.org/10.1016/j.jcis.2012.09.063. URL http://www.sciencedirect.com/science/article/pii/S0021979712011095.

[30] D. B. Soltman. *Understanding Inkjet Printed Pattern Generation.* Ph.D. thesis, University of California, Berkeley, 2011.

[31] FP7 European Commission funded project, grant nr. 287682. *Technology & Design Kits for Printed Electronics (TDK4PE)*, 2011. URL `http://www.tdk4pe.eu`.

[32] Spanish government funded project (MINECO), reference PN-TECTEC2011-29800-C03-01. *Application Specific Printed Electronics Circuits - Technology and Designk Kits (ASPEC-TDK)*, 2012.

[33] *Phoenix software.* `http://www.phoenixbv.com`. Accessed: 2015-09-08.

[34] J. Mujal, E. Ramon and J. Carrabina. *Methodology and tools for inkjet process abstraction for the design of flexible and organic electronics.* International Journal of High Speed Electronics and Systems, 20(04):829–842, 2011.

[35] E. Diaz, E. Ramon and J. Carrabina. *Inkjet Patterning of Multiline Intersections for Wirings in Printed Electronics.* Langmuir, 29(40):12608–12614, 2013. doi:10.1021/la402101d.

[36] A. Sangiovanni-Vincentelli. *The tides of eda.* Design Test of Computers, IEEE, 20(6):59–75, Nov 2003. ISSN 0740-7475. doi:10.1109/MDT.2003.1246165.

[37] L.-T. Wang, Y.-W. Chang and K.-T. T. Cheng. *Electronic design automation: synthesis, verification, and test.* Morgan Kaufmann, 2009.

[38] L. W. Nagel and D. O. Pederson. *SPICE: Simulation program with integrated circuit emphasis.* Electronics Research Laboratory, College of Engineering, University of California, 1973.

[39] L. W. Nagel. *Spice2: A computer program to simulate semiconductor circuits.* ERL Memo ERL-M520, 1975.

[40] L. Lavagno, L. Scheffer and G. Martin. *EDA for IC implementation, circuit design, and process technology.* CRC press, 2006.

[41] L. Pang, Y. Liu and D. Abrams. *Inverse lithography technology (ilt): What is the impact to the photomask industry?* In *Proc. SPIE*, volume 6283, page 62830X. 2006.

[42] Svilen Krustev. *Toped - open source layout editor.* URL `http://www.toped.org.uk/`, Consulted on 2015-06-12.

[43] Keith Sabine. *Glade - A fast IC Layout Editor/Viewer.* URL `http://www.peardrop.co.uk`, Consulted on 2015-06-12.

[44] C. Ebeling, N. McKenzie and L. McMurchie. *The gemini users guide*. 1994.

[45] Steven M. Rabin. *Electric VLSI*. URL `http://www.staticfreesoft.com`, Consulted on 2015-06-12.

[46] Tim Edwards. *Magic VLSI*. URL `http://www.opencircuitdesign.com/magic`, Consulted on 2015-06-12.

[47] Tim Edwards. *XCircuit*. URL `http://www.opencircuitdesign.com/xcircuit`, Consulted on 2015-06-12.

[48] Jean-Pierre Charras and Dick Hollenbeck and Wayne Stambaugh. *KiCad EDA Software Suite - KiCad EDA*. URL `http://www.kicad-pcb.org`, Consulted on 2015-06-12.

[49] gEDA Project. *gEDA Project*. URL `http://www.geda-project.org`, Consulted on 2015-06-12.

[50] O. V. International. *Verilog-A Language Reference Manual Analog Extensions to Verilog HDL*. 1996.

[51] L. Lemaitre, G. Coram, C. M. Andrew and K. Kundert. *Extensions to verilog-a to support compact device modeling*. In *Behavioral Modeling and Simulation, 2003. BMAS 2003. Proceedings of the 2003 International Workshop on*, pages 134–138. IEEE, 2003.

[52] Ngspice team. *Ngspice circuit simulator*. URL `http://www.ngspice.org`, Consulted on 2015-06-12.

[53] P. Nenzi and H. Vogt. *Ngspice Users Manual Version 25*. 2013.

[54] F. Cox, W. Kuhn, H. Li, J. Murray, S. Tynor and M. Willis. *Xspice software user's manual*. Georgia Tech Research Corp, 1992.

[55] L. Lemaitre, C. McAndrew and S. Hamm. *Adms-automatic device model synthesizer*. In *Custom Integrated Circuits Conference, 2002. Proceedings of the IEEE 2002*, pages 27–30. IEEE, 2002.

[56] Albert Davis. *Gnucap - The Gnu Circuit Analysis Package*. URL `http://www.gnucap.org`, Consulted on 2015-06-12.

[57] A. Davis. *Gnucap The Gnu Circuit Analysis Package Users manual*. 2006. URL `gnucap.org/gnucap-man.pdf`.

[58] A. Davis. *The gnucap model compiler*. In *Behavioral Modeling and Simulation, 2002. BMAS 2002. Proceedings of the 2002 IEEE International Workshop on*, pages 112–117. IEEE, 2002.

[59] The PDAFlow Foundation. *PDAFlow Foundation*. URL `http://www.pdaflow.org`, Consulted on 2015-06-12.

[60] A. Cerdeira and M. Estrada and R. García and A. Ortiz-Conde and F.J. García Sánchez. *New procedure for the extraction of basic a-Si:H TFT model parameters in the linear and saturation regions*. Solid-State Electronics, 45(7):1077 – 1080, 2001. ISSN 0038-1101. doi:http://dx.doi.org/10.1016/S0038-1101(01)00143-5.

[61] M. Estrada and I. Mejía and A. Cerdeira and J. Pallares and L.F. Marsal and B. Iñiguez. *Mobility model for compact device modeling of OTFTs made with different materials*. Solid-State Electronics, 52(5):787 – 794, 2008. ISSN 0038-1101. doi:http://dx.doi.org/10.1016/j.sse.2007.11.007.

[62] B. Kolpackov. *C++/Tree Mapping User Manual*. 2014.

[63] FP7 European Commission funded project, grant nr. 288881. *Comercialization of Large Area Electronics (COLAE)*, 2011. URL `http://www.colae.eu`.

[64] Adrià Conde and Jofre Pallarès and Lluís Terés and Carme Martínez-Domingo and Eloi Ramon and Jordi Carrabina. *PCell based devices & structures for Printed Electronics and related semiautomatic characterization loop*. In *DCIS'2013: XXVIII Conference of Circuits and Integrated Systems*. 2013.

[65] F. Vila, J. Pallares, A. Conde and L. Teres. *A fully-automated methodology and system for printed electronics foil characterization*. In *Microelectronic Test Structures (ICMTS), 2015 International Conference on*, pages 188–192. March 2015. ISSN 1071-9032. doi:10.1109/ICMTS.2015.7106138.

[66] N. Otsu. *A threshold selection method from gray-level histograms*. Automatica, 11(285-296):23–27, 1975.

[67] J. Serra and L. Vincent. *An overview of morphological filtering*. Circuits, Systems and Signal Processing, 11(1):47–108, 1992.

[68] R. Szeliski. *Image alignment and stitching: A tutorial*. Foundations and Trends® in Computer Graphics and Vision, 2(1):1–104, 2006.

[69] P. Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[70] L. Shapiro and G. C. Stockman. *Computer vision. 2001*. ed: Prentice Hall, 2001.

[71] P. F. Dunn. *Measurement and data analysis for engineering and science*. CRC press, 2014.

[72] A. V. Oppenheim, R. W. Schafer, J. R. Buck et al. *Discrete-time signal processing*, volume 2. Prentice-hall Englewood Cliffs, 1989.

[73] P. Felzenszwalb and D. Huttenlocher. *Distance transforms of sampled functions.* Technical report, Cornell University, 2004.

[74] P. F. Felzenszwalb and D. P. Huttenlocher. *Distance transforms of sampled functions.* Theory of computing, 8(1):415–428, 2012.

[75] E. Ramon, C. Martínez-Domingo and J. Carrabina. *Geometric design and compensation rules generation and characterization for all-inkjet-printed organic thin film transistors.* Journal of Imaging Science and Technology, 57(4):40402–1, 2013.

[76] W. C. Navidi. *Statistics for engineers and scientists.* McGraw-Hill Higher Education, 2008.

[77] G. E. Box and D. R. Cox. *An analysis of transformations.* Journal of the Royal Statistical Society. Series B (Methodological), pages 211–252, 1964.

[78] R. Sakia. *The box-cox transformation technique: a review.* The statistician, pages 169–178, 1992.

[79] R. Davidson and J. G. MacKinnon. *Estimation and Inference in Econometrics.* Number 9780195060119 in OUP Catalogue. Oxford University Press, March 1993.

[80] B. L. Welch. *The Generalization of 'Student's' Problem when Several Different Population Variances are Involved.* Biometrika, pages 28–35, 1947.

[81] J. W. Tukey. *Comparing individual means in the analysis of variance.* Biometrics, 5(2):pp. 99–114, 1949. ISSN 0006341X. URL `http://www.jstor.org/stable/3001913`.

[82] M. Estrada, A. Cerdeira, J. Puigdollers, L. Resendiz, J. Pallares, L. Marsal, C. Voz and B. Iñiguez. *Accurate modeling and parameter extraction method for organic tfts.* Solid-state electronics, 49(6):1009–1016, 2005.

[83] P. Hart, T. Van 'T Hof and F. Klaassen. *Device down scaling and expected circuit performance.* Electron Devices, IEEE Transactions on, 26(4):421–429, Apr 1979. ISSN 0018-9383. doi:10.1109/T-ED.1979.19444.

[84] L. D. Brown, T. T. Cai and A. DasGupta. *Interval estimation for a binomial proportion.* Statist. Sci., 16(2):101–133, 05 2001. doi:10.1214/ss/1009213286. URL `http://dx.doi.org/10.1214/ss/1009213286`.

[85] A. Agresti and B. A. Coull. *Approximate is better than "exact" for interval estimation of binomial proportions.* The American Statistician, 52(2):119–126, 1998.

[86] D. G. Bonett and R. M. Price. *Adjusted wald confidence interval for a difference of binomial proportions based on paired data.* Journal of Educational and Behavioral Statistics, 37(4):479–488, 2012.

[87] S. Chang and P. Paul. *Effect of ink drop coalescence on print quality in solid ink printing.* In *NIP & Digital Fabrication Conference*, volume 2004, pages 429–434. Society for Imaging Science and Technology, 2004.

[88] B. Derby. *Inkjet printing of functional and structural materials: fluid property requirements, feature stability, and resolution.* Annual Review of Materials Research, 40:395–414, 2010.

[89] C. D. Systems. *Design Data Translator's Reference.* Cadence, 2008.

[90] M. D. Berg, O. Cheong, M. V. Kreveld and M. Overmars. *Computational geometry: algorithms and applications.* Springer, 2008. doi: 10.1007/978-3-540-77974-2.

[91] H. Kriegel, T. Brinkhoff and R. Schneider. *An efficient map overlay algorithm based on Spatial Access Methods and Computational Geometry.* In *Proc. Int. Workshop on Database Management Systems for Geographical Applications, Capri, Italy*, pages 194–211. 1991.

[92] F. Martínez, A. J. Rueda and F. R. Feito. *A new algorithm for computing Boolean operations on polygons.* Computers & Geosciences, 35(6):1177–1185, June 2009. ISSN 00983004. doi:10.1016/j.cageo.2008.08.009.

[93] V. Milenkovic. *Practical methods for set operations on polygons using exact arithmetic.* In *Proc. 7th Canad. Conf. Comput. Geom*, pages 55–60. Citeseer, 1995.

[94] E. Fogel, R. Wein, B. Zukerman and D. Halperin. *2D regularized Boolean set-operations.* In *CGAL User and Reference Manual.* CGAL Editorial Board, 3.6 edition, 2010.

[95] M. I. Shamos and D. Hoey. *Geometric intersection problems.* 17th Annual Symposium on, pages 208–215, 1976. doi:10.1109/SFCS.1976.16.

[96] U. Finke and K. H. Hinrichs. *Overlaying simply connected planar subdivisions in linear time.* In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 119–126. ACM, New York, NY, USA, 1995. ISBN 0-89791-724-3. doi:http://doi.acm.org/10.1145/220279.220292.

[97] B. R. Vatti. *A generic solution to polygon clipping.* Communications of the ACM, 35(7):56–63, 1992.

[98] M. K. Agoston. *Computer Graphics and Geometric Modeling: Implementation and Algorithms.* Springer, 2005.

[99] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra and C. Yap. *Classroom examples of robustness problems in geometric computations.* Computational Geometry, 40(1):61–78, 2008.

[100] M. Held. *On the computational geometry of pocket machining*, volume 500. Springer Science & Business Media, 1991.

[101] O. Aichholzer, F. Aurenhammer, D. Alberts and B. Gärtner. *A novel type of skeleton for polygons.* Journal of Universal Computer Science, 1(12):752–761, 1995. doi:10.3217/jucs-001-12-0752.

[102] P. Felkel and S. Obdrzalek. *Straight skeleton implementation.* In *Proceedings of spring conference on computer graphics.* Citeseer, 1998.

[103] X. Chen and S. McMains. *Polygon offsetting by computing winding numbers.* In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 565–575. American Society of Mechanical Engineers, 2005.

[104] W. Wouters. *Bmp format–windows bitmap file format specifications, v1. 1.* Clean Coding Company, 1997.

[105] J. D. Foley, A. Van Dam, S. K. Feiner, J. F. Hughes and R. L. Phillips. *Introduction to computer graphics*, volume 55. Addison-Wesley Reading, 1994.

[106] A. Bakker, N. Olij, D. Marín and L. Terés. *D5.1. Specification of the API for PE design flow implementation.* Confidential deliverable, TDK4PE Project (FP7/2011-2014 contract agreement no 287682), 2012.