A Parasitic Resistance Extraction Tool Leveraged by Image Processing

Dias, Diogo ; Goes, Joao ; Costa, Tiago

DOI
10.1109/ISCAS48785.2022.9937879

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE International Symposium on Circuits and Systems (ISCAS)

A Parasitic Resistance Extraction Tool Leveraged by Image Processing

Diogo Dias
Department of Electrical and Computer Engineering
School of Science and Technology
NOVA University of Lisbon
CTS/UNINOVA
Almada, Portugal
Email: das.dias@campus.fct.unl.pt

João Goes
Department of Electrical and Computer Engineering
School of Science and Technology
NOVA University of Lisbon
CTS/UNINOVA
Almada, Portugal
Email: jg@uninova.pt

Tiago Costa
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics & Computer Science
Delft University of Technology
Delft, Netherlands
Email: T.M.L.daCosta@tudelft.nl

2022 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-6654-8485-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISCAS48785.2022.9937879

***Abstract*—Most academic and commercial tri-dimensional (3D) parasitic resistance extraction EDA/CAD tools rely on finite element methods (FEM) and are mainly suited to digital circuitry. In analog and mixed-signal (AMS) circuits, such as power converters and radio-frequency analog front-ends, the layout structures used for the metal interconnections become much more diversified and complex. This paper proposes an EDA/CAD tool, based on an innovative methodology for 3D parasitic resistance extraction, leveraged by image processing techniques and algorithms. Some practical examples are shown to demonstrate the attractiveness of the proposed tool. Moreover, since our tool efficiently works in the domains of 2D image processing, if an extensive database of layouts is provided and enough training is carried out, advanced deep-learning techniques can be straightforwardly employed, speeding up parasitic resistance extraction in highly complex AMS layouts.**

I. Introduction

Post-layout verification EDA/CAD software tools have growing importance as the semiconductor technology nodes shrink down, allowing for increasingly power efficient and optimized integrated circuit layout designs [1], [2]. Most AC and DC Voltage drop and *electromigration* (*EM*) analysis software tools take advantage of 2D and 2.5D meshing software to perform electric analysis on the several metal structures in each layer. Using finite element methods (FEM), the electric potential difference and extracted parasitic resistance between nodes are calculated [3], [4]. These tools are usually optimized for the verification of power and/or ground nets of digital integrated circuitry [5]–[8]. In analog circuit layouts, more complicated metal structures are present, and the power transferred on dedicated signal lines require IR Voltage drop verification on signal transferring interconnects. As the circuit area increases, the computation time for these EDA/CAD tools also increases, sometimes requiring several hours to complete an extraction.

In this work, it is presented a new methodology leveraged by *Image Processing* techniques in order to extract the central paths for layer interconnections and VIAs on which parasitic resistances are mapped upon. The methodology is integrated in a software tool aiming to provide a layout debugging tool generalized towards analog and/or digital circuits. A comparison between this tool and other referenced existing tools is summarized in Table I.

TABLE I
Feature comparison - referenced tools vs. this tool.

Feature	Tools		
	[4]	[5]–[8]	This Work
Automatic Netlist Extraction	✓	✓/✗	✓
DC EM Simulation (Current Density Mapping)	✓	✓	✗/✓
DC IR-Voltage Drop Simulation	✓	✓	✓
AC Simulation	✗	✓	✗
Easy to Use	✓	✗	✓
Geometry Feature Extraction	A	A	B

Legend: A - Polygon fragmentation methods using mesh structures (FEM); B - Image processing methods for geometry pre-processing.

II. Application Description

The application follows a simple flow of use and operation, making easier the job of the AMS layout engineer and thus accelerating the full inspection of the layout under verification.

A. User Guide

The import, management, visualization and modification of the ".GDS" file is handled in the application by use of the dedicated *Python*™ library *gdspy* (release version 1.6.9) [9]. In order to build the "Input PAD Locations" file, one must first observe the layout of the cell, block or entire system to

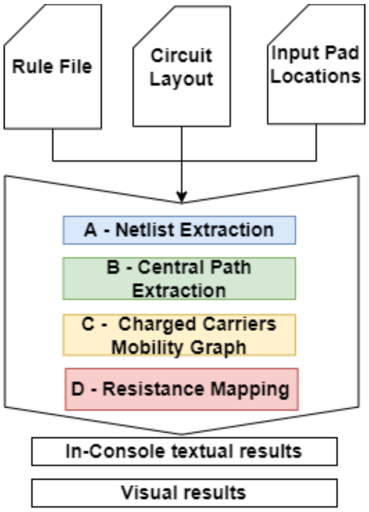


Fig. 1. Application flow of operation.

chose the GDSII coordinates of each input PAD. The software tool's supported operations are:

1) Import the ".GDS" layout file;
2) Visualize the layout file;
3) Import the necessary technology files for the resistance analysis;
4) Run DC resistance distribution analysis;
5) Visualize resistance distribution analysis' resulting coloured heat-map.

B. File Import

For the application to operate there are three necessary files that must be imported:

1) Rule Files: contain technology-specific physical parameters. The considered PDK is the open-source *Skywater* 130 nm fabrication process [10].
2) circuit's "Layout Mask File": Only GDSII geometry files (".GDS") are supported for import.
3) metal "Input PAD Locations File": application-specific text files (.TXT) containing the x, y coordinates of the sites of contact between a voltage supply source and the integrated circuit.

The x and y 2D coordinates are unit-less ('meter-less') - they must be inserted in the GDSII file coordinate system.

III. Methods & Algorithms

The methodologies for point-to-point resistance extraction presented in this section rely on *Image Processing* and *Graph Theory* as tools that allow for the calculation of the electric current paths assumed inside metal layers of any shape, allowing for the extraction of central paths even for metal layers containing multiple slots (metal openings). The flow of operation of the tool is presented in Figure 1.

A. Netlist Extraction

The tool supports automatic netlist extraction from the imported ".GDS" file. Taking advantage of the functionalities of the *Python*™ programming language and the *gdspy* library, the several polygons structures that are on the same layer and intersect each other are joined in a single polygon.

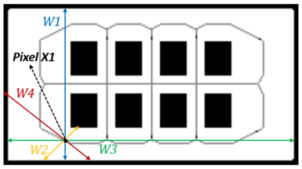


Fig. 2. Image Skeleton (in grey) of a Metal 1 (in white) interconnect with open slots (background in black). 0° segment (blue); 45° segment (yellow); 90° segment (green); 135° segment (red).

On the resulting GDSII library, the several netlists featured in the imported ".GDS" file are extracted by checking which polygons placed in two successive metal layers intercept the same VIA geometric element.

B. Path Extraction

The *gdspy* library supports the translation of the GDSII library geometries to ".SVG" image format, and the *Python™* library *svglib* [11] allows for the translation of ".SVG" format to RGB ".PNG" image formatting. The extraction of the central paths for each metal structure in the GDSII library is separated in two processes involving image processing algorithms:

1) Central Points Extraction: By computing the *Image Skeleton* (represented by the grey pixels in Figure 2) of the image of each metal layer polygon, recurring to the *Guo-Hall Thinning Algorithm* [12], the central pixels are obtained and furthermore converted into 2D GDSII coordinates.

2) Width Extraction: For each pixel belonging to the central path that is extracted using the thinning algorithm, four line segments (presenting angles of 0°, 45°, 90°, and 135°) are built, connecting the nearest borders of the metal structure being processed while intercepting each pixel - as also represented in Figure 2. For each point, the lengths of these four line segments represent the widths of the metal polygon in four different directions, and the assumed metal width is the minimum detected width (1).

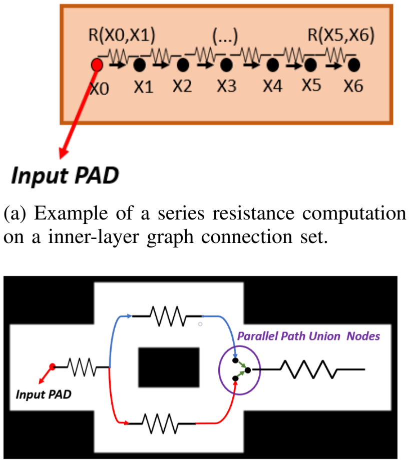
$$W(x_1) = min\{W_1, W_2, W_3, W_4\} \tag{1}$$

where: (see Figure 2) W_1 - length of 0° segment; W_2 - length of 45° segment; W_3 - length of 90° segment; W_1 - length of 135° segment; $W(x_1)$ - width of the metal interconnect at coordinates x_1.

The conversion of coordinates and width from pixel units to GDSII library units is done by performing a linear domain transformation by using the GDSII layer and its respective image *bounding boxes* (rectangular boundaries).

C. Current Drifting Graph

After importing the input PAD locations file, an inner-layer connectivity graph through out the pixels belonging to the central path of each metal layer is built, interpreting each central



(a) Example of a series resistance computation on a inner-layer graph connection set.

(b) Example of a inner-layer graph connection set featuring a parallel resistance path with 3 parallel path union nodes as well.

Fig. 3. Inner-layer graph connection set examples.

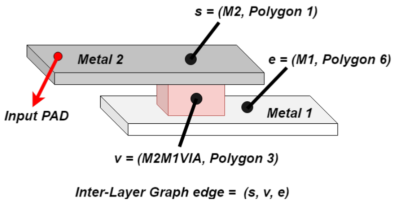


Fig. 4. Inter-layer graph connection example.

path associated pixel as a graph node. Parallel processing is employed in this task [13] for each node that branches out of a path, to address all parallel nodes "simultaneously". The graph, exemplified in Figure 3, is constituted by two types of ending nodes: parallel path union nodes (ending nodes featuring neighbour ending nodes) and final nodes (ending nodes featuring no neighbour nodes).

An inter-layer connectivity graph, exemplified in Figure 4 is also built for connections established through the VIAs layers, in which the central point of each VIA polygon is interpreted as an input PAD location for the layer to which the connection is done. For each input PAD specified by the user in the input PAD locations file a set of inner and inter-layer graphs will be created.

$$R(x_1, x_2) = \frac{\rho}{M_{TH}} \frac{||(x_1, x_2)||}{W(x_2)} \tag{2}$$

where : ρ - metal layer resistivity $[\Omega m]$; M_{TH} - metal layer thickness $[m]$; x_1 - previous node; $R(.)$ - resistance between successive nodes; $[\Omega]$; $W(.)$ - nodal metal width $[m]$; $||x_1, x_2||$ - euclidean distance between successive node coordinates x_1 and x_2.

D. Resistance Mapping

In this section, the methods used in point-to-point parasitic resistance extraction are presented. The resistance value between each point of a connection is always attributed to the final node of each connection, and a color code is attributed to it. Lower resistance values will be coloured with "colder" colours closer to the minimum resistance value (coloured in dark blue), while higher resistance values will be coloured in "hotter" colours converging towards the maximum resistance value of the layout (coloured in dark red), as observed in Figures 5, 6 and 7. The series resistance between two successive nodes of the connectivity graph is given by (2). Depending on whether there are parallel current paths or not in the interconnect, two different approaches are used:

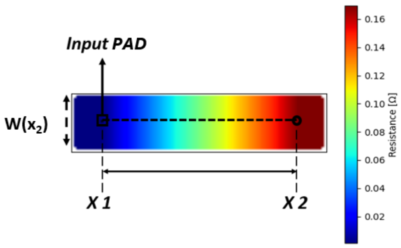


Fig. 5. Distributed resistance mapping example of a straight strip belonging to metal 1

1) Series Resistance Mapping: Using *Dijkstra's Shortest Path Algorithm* [14] the single shortest path between the input PAD associated node and the ending-nodes of the connectivity graph of the metal structure is obtained. This guarantees that, in the case of multiple possible paths extracted between the referred nodes, only the lowest resistance one is considered for each end-node, neglecting possible parallel resistance paths. The series parasitic resistance between the input point coordinates (x_i) and each point of the central path (x_n) is obtained through (3). This method is represented in Figures 3a and 5.

$$R(x_i, x_n) = \sum_{i=1}^{size\{P_N\}} R(x_{P_N\{i\}}, x_{P_N\{i+1\}}) \tag{3}$$

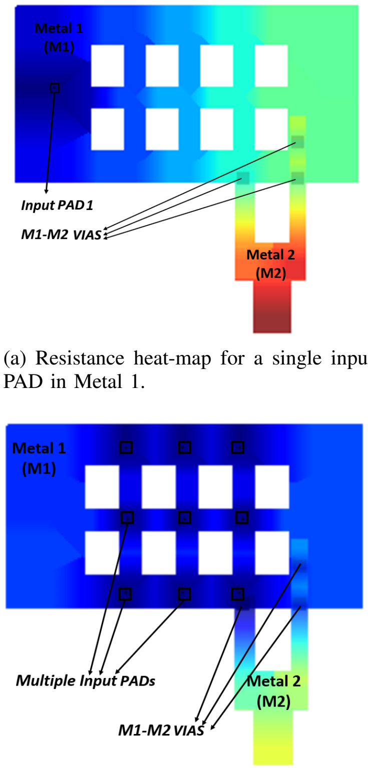
where : P_N is the set of nodes that constitute a path between the input node x_i and the ending node x_n.

2) Parallel Resistance Mapping: As a first step, for each one of the parallel path union nodes (exemplified in Figure 3b), the series parasitic resistance is calculated between the input node and the node itself $(R(x_i, x_N))$ and its neighbouring end-nodes $(R(x_i, x_{S_N\{k\}}))$. The equivalent parallel resistance for each of the referred nodes is afterwards computed using the inverse of the sum of the corresponding conductances - as represented in equation (4).

$$R_p(x_i, x_n) = \frac{1}{\frac{1}{R(x_i, x_n)} + \sum_{k=1}^{size\{S_N\}} \frac{1}{R(x_i, x_{S_N\{k\}})}} \tag{4}$$

where: S_N is the set of neighbour end-nodes of the x_n node.

In a second step, the resistance map is constructed from the input node towards the parallel path interception nodes (with the resistance at ending-nodes being the maximum possible resistance values in between them). In a third and final step,



(a) Resistance heat-map for a single input PAD in Metal 1.

(b) Resistance heat-map for multiple (9 - nine) input PADs in Metal 1.

Fig. 6. Resistance heat-map of a testing layout mask. Warmer colours encode a higher resistance; Cooler colours encode a lower resistance.

the series resistance is mapped between each parallel path interception node towards each final end-node.

The resistance of each VIA is of course also taken into account in the starting resistance value in inter-layer interconnections.

IV. Results

The proposed EDA/CAD tool has been tested on simple 2D and 3D layouts. The obtained results for simpler netlist layouts and more complex layouts featuring multiple open slots on the metal structure, as shown in Figure 6 and Figure 7, present promising and accurate resistance distribution results for a tool relying on image processing techniques rather than FEM. The proposed tool allows for single or multiple input PADs to be considered in the same netlist, as it is possible to observe from Figures 6a and 6b. Figure 7 shows that the proposed tool can help to detect layout problems such as the lack of enough VIAs by featuring faster approaches to the saturation resistance value on metal strips with the same shape.

The obtained average computing times for the resistance map of the layout presented in Figure 7, presenting an area of 152.1828 μm^2 (12.8100 μm × 11.8800 μm), are shown in Table II - obtained using an Intel(R) Core(TM) i7-6700HQ CPU with 4 Cores @ 2.60 GHz. Overall, the results obtained in Table II are quite promising for the light complexity of the layout, when compared to other referenced tools.

TABLE II
Computing time measurements for the layout example shown in Figure 7.

Operation	Average Computing Time [s]	%
Netlist Extraction	0.0432	0.93
Central Paths Extraction	4.0339	86.67
Connectivity Graph Development	0.5591	12.01
Resistance Mapping	0.0182	0.39
Total Computing Time	4.6544	100

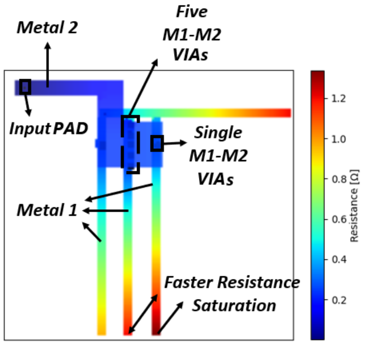


Fig. 7. Observation of a faster resistance saturation when establishing a connection between successive metal layers using less contact VIAs.

Transposing the proposed tool from *Python*™ programming language into a compiled, low-level programming language (such as *C++*™), the timings presented in Table II can be further reduced. Furthermore, it is expected that migrating the image processing tasks to a GPU will also lead to great improvements in its running-time. The tool must be tested on more complex VLSI layouts to obtain a more comprehensive measurement on the advantages and disadvantages of the methodologies used for resistance mapping or voltage drop analysis presented in this work over other approaches.

V. Conclusion

This paper proposed an EDA/CAD tool, based on an innovative methodology for 3D parasitic resistance extraction, leveraged by image processing techniques and algorithms. Some practical examples of small layouts have been shown to demonstrate the attractiveness of the proposed tool. Moreover, since our tool efficiently works in the domains of 2D image processing, if an extensive database of layouts is provided and enough training is carried out, advanced deep-learning techniques can be straightforwardly employed, speeding up parasitic resistance extraction in highly complex AMS layouts.

VI. Acknowledgements

This research work has been supported by XILINX Ireland and also by FEDER funds through National Funds provided by FCT - Fundação para a Ciência e a Tecnologia, I.P., under the scope of the strategic project of CTS/UNINOVA (UIDB/00066/2020).

REFERENCES

[1] Marković, D., et al., ”The Role of Post-Layout Verification in Microprocessor Design”, 2004, Proceedings of MIPRO 2004 27th International Convention, MEET & HGS.

[2] Jain, Bhawana & Khare, Kavita, ”A Comparative Study of Methodologies to Optimize Post- layout Challenges”, 2015, International Journal of Computer Applications. 124. 27-30. doi: 10.5120/ijca2015905552.

[3] E. Liu, et al., ”Effective resistance approach for DC analysis of power grid on through-silicon interposer (TSI),” 2012, IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS), 2012, pp. 1-4, doi: 10.1109/EDAPS.2012.6469410.

[4] M. Ershov, et al., ”P2P and Rmap - new software tool for quick and easy verification of power nets,” 2015 37th Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD), 2015, pp. 1-8, doi: 10.1109/EOSESD.2015.7314754.

[5] N. Chang, et al., ”Efficient multi-domain ESD analysis and verification for large SoC designs,” EOS/ESD Symposium Proceedings, 2011, pp. 1-7.

[6] Y. Shao, Z. Peng and J. Lee, ”Analysis of IR-drop in 3-D IC packaging using a non-conformal domain decomposition method,” 2012 Asia-Pacific Symposium on Electromagnetic Compatibility, 2012, pp. 405-408, doi: 10.1109/APEMC.2012.6237966.

[7] M. Ershov, et al., ”EDA software for verification of metal interconnects in ESD protection networks at chip, block, and cell level,” 2013 35th Electrical Overstress/Electrostatic Discharge Symposium, 2013, pp. 1-7.

[8] L. Xue and D. Jiao, ”Rapid Modeling and Simulation of Integrated Circuit Layout in Both Frequency and Time Domains From the Perspective of Inverse,” in IEEE Transactions on Microwave Theory and Techniques, vol. 68, no. 4, pp. 1270-1283, April 2020, doi: 10.1109/TMTT.2020.2966699.

[9] Gabrielli, L. H. (n.d.) - gdspy 1.6.9 documentation. Retrieved from https://gdspy.readthedocs.io/en/stable/gettingstarted.html.

[10] Welcome to SkyWater SKY130 PDK’s documentation! - SkyWater SKY130 PDK 0.0.0-308-gdb2e067 documentation. (n.d.). Retrieved from https://skywater-pdk.readthedocs.io/en/latest/index.html#.

[11] Svglib. PyPI. (n.d.). Retrieved from https://pypi.org/project/svglib/.

[12] Gramblička, Matúš & Vasky, Jozef. (2016). Comparison of thinning algorithms for vectorization of engineering drawings. 94. 265-275.

[13] Threading - thread-based parallelism. threading - Thread-based parallelism - Python 3.10.0 documentation. (n.d.). Retrieved from https://docs.Python.org/3/library/threading.html.

[14] Dijsktra’s algorithm. GeeksforGeeks. (2021, September 10). Retrieved from https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/.