01 Jan 2009

Cavities' Identification Algorithm for Power Integrity Analysis of Complex Boards

S. Cicerone

Antonio Orlandi

Bruce Archambeault
Missouri University of Science and Technology, archamb@mst.edu

Samuel R. Connor

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/ele_comeng_facwork/1478

Recommended Citation

S. Cicerone et al., "Cavities' Identification Algorithm for Power Integrity Analysis of Complex Boards," *Proceedings of the 20th International Zurich Symposium on Electromagnetic Compatibility (2009, Zurich, Switzerland)*, pp. 253-256, Institute of Electrical and Electronics Engineers (IEEE), Jan 2009.
The definitive version is available at https://doi.org/10.1109/EMCZUR.2009.4783438

Cavities' Identification Algorithm for Power Integrity Analysis of Complex Boards

S. Cicerone #1, A. Orlandi *2, B. Archambeault †3, S. Connor †4, J. Fan ‡5, J. L. Drewniak ‡6

Dept. of Electrical & Information Engineering, Univ. of L'Aquila, L'Aquila, Italy
1 cicerone@ing.univaq.it

* *UAq EMC Laboratory, Dept. of Electrical & Information Engineering, Univ. of L'Aquila, L'Aquila, Italy*
2 orlandi@ing.univaq.it

† *IBM RTP, 3039 Cornwallis Rd, NC 27709, USA*
3 barch@us.ibm.com, 4 connor@us.ibm.com

‡ *EMC Laboratory, Missouri Univ. of Science & Technology, Rolla, USA*
5 jfan@mst.edu, 6 drewniak@mst.edu

***Abstract*—The analysis and the design of the Power Delivery Network (PDN) is crucial in the real world of high-speed and high-performance on-board systems. In this context, the Cavity Model (where facing portions of power bus are considered electromagnetic resonant cavities) can be used to study the generation and propagation of noise. Given a real-world board's layout, one of the primary requirements for the application of this technique is the geometrical identification of all the cavities and their connectivity. This paper is focused on the fully automatic generation of this geometrical dataset as part of an integrated tool for the analysis and design of PDN.**

I. Introduction

In today applications, microprocessors and application-specific integrated circuits (ASIC) have thousands of gates switching simultaneously. The impulsive and repetitive current drawn by these active devices from the Power Delivery Network (PDN) is a challenging issue for a correct and reliable PDN design and a severe source of electromagnetic noise generation. In high-speed digital circuit designs, the PDN associated with the PCB plays a vital role in maintaining not only the power integrity (PI), i.e. the high quality of the DC voltage level, but also the signal integrity (SI) as the necessary fidelity of signal and clock wave shapes, and minimizing electromagnetic noise generation. As integrated circuit (IC) technology is scaled downward to yield smaller and faster transistors, the power supply voltage must decrease. As clock rates rise and more functions are integrated into microprocessors and ASICs, the power consumed must increase, meaning that current levels, i.e., the movement of electrical charge, must also increase [1], [2].

The PDN for modern medium-to-high-speed digital PCBs is usually formed from one or more pair of conducting planes used as power (PWR) and power return (very often improperly called "ground" GND). The PDN for digital circuitry has evolved over time, as signal and clock speeds have increased, from discrete power supply wires, to discrete traces, to area fills and ground islands on single/two-layer slow-speed boards, to the planar power bus structure used extensively in today's multi-layer high-speed PCBs. The low inductance associated with charge delivery from the plane to circuit element allows for the storage of relatively easy-to-deliver charge available all over the board.

Often the term power bus is used to identify an individual plane pair, whereas the term PDN is used for the entire system of supplying power to circuits placed on the PCB. Noise is generated in the power bus when a digital active device (integrated circuit or transistor) switches between its high and low logical states (switching noise) [3], [4], or it can be coupled to the power bus when a high-speed signal transits through the power bus by signal vias (transition noise) [5]. Noise generated in the power bus can be easily propagated throughout the board. Propagated noise can affect the operation of other active devices (signal integrity) as well as radiate from the PCB (EMI). Among the possible techniques to study the generation and propagation of noise there is the so called Cavity Model [6] in which facing portions of power bus are considered electromagnetic resonant cavities. Given a real-world board's layout, one of the primary requirements for the application of this technique, is the geometrical identification of all the cavities and their connectivity. Then a suitable processing of the geometrical cavities' boundaries is requested for a correct and not over-detailed electromagnetic modeling. After these actions the geometry (containing also the electrical parameters such as electric permittivity of the substrates, electrical conductivity of the planes, etc.) dataset is ready for being input to the cavity model solver.

This paper is focused on the fully automatic generation of the above mentioned geometrical dataset as part of an integrated tool for the analysis and design of PDN.

II. The Multiplane Cavity Model

The extension of the cavity model to a multiplane configuration is straightforward. Each powerbus is segmented in elementary shapes forming cavities; for each cavity, based on [6] the impedance matrix (Z) or the S-parameters are computed among the ports. The cavities are then interconnected by enforcing the electromagnetic boundary conditions such

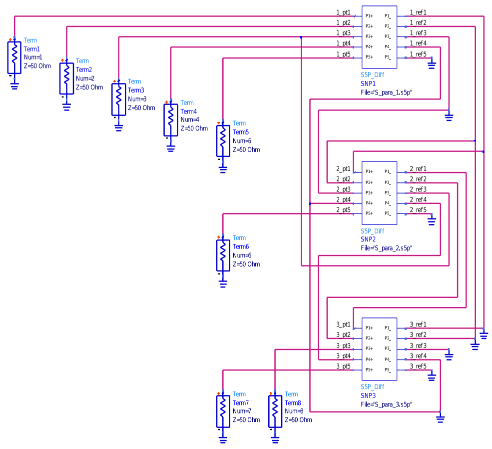

Fig. 1. A three cavities structure assembled in ADS.

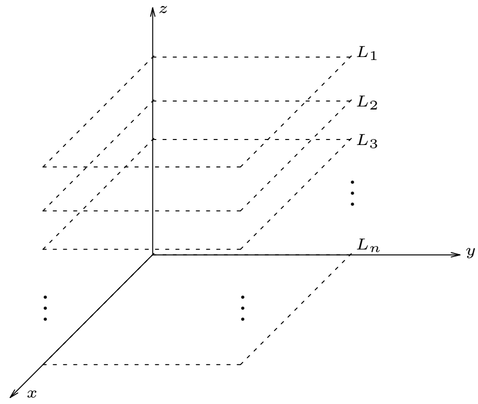

Fig. 2. *An example of multiplane structure consisting of n layers (layers shapes are not depicted).*

as the voltage and currents' continuity. Fig. 1 [4] shows an example of a three cavities structure in which the assembling is manually worked out by using ADS [7].

In the next section it will be considered the algorithms for an automatic and exact identification of the cavities forming a PDN in order to apply the cavity model approach

III. The Cavity Identification Problem

In what follows, a formalization of the cavity identification problem is given. In our context, a **board** can be modeled as a multiplane structure defined by geometric objects and relationships as follows (see Fig. 2):

Definition 3.1:

- a multiplane structure consists of n stacked **layers**; formally, $MS = \{L_1, L_2, \ldots, L_n\}$, where L_1 (L_n, resp.) is the top (bottom, resp.) layer;
- each layer is a rectangle in a 3-dimensional reference system. Each layer lies in a plane parallel to the $z = 0$ plane. All the layers have the same dimensions;
- the distance between two layers is defined by the function $d : MS \times MS \to \mathbb{R}$. Note that, the distance between two layers depends on their z-coordinate only;
- each layer contains **shapes**; shapes of layer L_i can be modeled as a set P_i of **polygons** such that: (i) each polygon has non-zero area, (ii) each polygon may have **holes**, and (iii) polygons are pairwise disjoint;
- polygons in P_i are partitioned into n_i **nets**: $P_i = \{N_i^1, N_i^2, \ldots, N_i^{n_i}\}$, $n_i \geq 1$, $1 \leq i \leq n$;
- a **via** is a vertical line at point (x, y) passing through layers. A via may or may not "be connected" to a given layer. Given $p \in P_i$, the relationship $connects(via(x, y), p)$ holds iff $via(x, y)$ is connected to p.

Given a polygon p, by $\bar{p}$ we denote the translation of p at the $z = 0$ plane.

Definition 3.2: Let $p_1 \in P_i$ and $p_2 \in P_j$, $i < j$, be two polygons fulfilling the following conditions:

- $\bar{p_1} \cap \bar{p_2} = p$, $p \not\equiv \emptyset$;
- $p \cap \bar{p}' = \emptyset$, for each $p' \in P_k$, and for each $k \in [i+1, j-1]$.

In this case, the polyhedron having the polygon p as base area and $d(L_i, L_j)$ as height is called **physical cavity**. Such a cavity is denoted as $cav(p_1, p_2)$; we say that $cav(p_1, p_2)$ is *induced* by p_1 (by *projecting* onto p_2).

Notice that, according to the cavity definition and to the fact that polygons of a layer are pairwise disjoint, if two cavities share some points, then such points must belong to the lateral surface of the cavities. This properties leads to the following definitions.

Definition 3.3: Let $c_1 = cav(p_1, p_2)$ and $c_2 = cav(p_1, p_3)$ be two cavities induced by the same polygon p_1 by projecting onto different polygons p_2 and p_3. c_1 and c_2 are **horizontal adjacent** if they share some points.

Definition 3.4: Let $c_1 = cav(p_1, p_2)$, $c_2 = cav(p_2, p_3)$, and c_3 be three distinct cavities. If c_3 shares points in the lateral surfaces with both c_1 and c_2, then c_1 and c_2 are **vertical adjacent**.

Definition 3.5: Let $c_1 = cav(p_1, p_2)$ and $c_2 = cav(p_3, p_4)$ be two cavities. If there exists a via $via(x, y)$ such that both $connects(via(x, y), p_1)$ and $connects(via(x, y), p_3)$ hold, then c_1 and c_2 are **via adjacent**.

Two cavities are adjacent if they are horizontal, vertical, or via adjacent.

Definition 3.6: Given a sequence of cavities $c_1, c_2, \ldots, c_q$, $q \geq 2$, such that c_i and c_{i+1} are adjacent, $1 \leq i < q$, then we say that c_1 and c_q are **connected**.

Problem 1 (Physical Cavities)*:* Given a multiplane structure MS, computes all the physical cavities.

The cavity model solvers currently available require that cavities must have *primitive polygons* as base. At the moment, the primitive polygons consist of triangles and rectangles with the following additional constraints:

- each triangle has a fixed shape: the internal angles must have values 90, 45, and 45 degrees;
- the triangle hypotenuse must lie on the boundary of p
- the aspect ratio of each rectangle is at most 20:1

Let cav be a physical cavity. Constraints above imply that cav has to be further processed before to be considered as a “valid” input for the cavity model solver. In particular, if p is the base polygon of cav, then the boundary of p has to be “approximated” to provide a new polygon p' such that it can be partitioned into primitive polygons. Such a subdivision produces **logical cavities**, that is, all the polyhedra induced by triangles and rectangles. So, the following additional two combinatorial problems naturally arise: POLYGON APPROXIMATION, and POLYGON DECOMPOSITION.

*Problem 2 (*POLYGON APPROXIMATION*):* Let p be a polygon with holes. Approximate p into a polygon p' such that the boundary of p' is formed by horizontal, vertical, or oblique (45 degrees wrt exes) segments. The difference between the segments of p and p' is less than some error according to a given criterion $\mathcal{C}$.

In what follows, the approximated polygon is called *quasi-orthogonal* polygons (in contrast to orthogonal polygons, for which the boundary is formed by horizontal and vertical segments only). The follows problem refer to the segmentation.

*Problem 3 (*POLYGON DECOMPOSITION*):* Let p a quasi-orthogonal polygon with holes. Compute a partition of p into primitive components with the minimum number of elements.

Even if the notion of cavity connection has been given for the physical case, it can be easily extended to logical cavities. So, the following concludes the formulation of the cavities identification problem.

*Problem 4 (*CAVITY CONNECTIONS*):* Let LC be a set of logical cavities. Determine the cavity connection relationship in LC.

IV. THE CAVITY IDENTIFICATION ALGORITHM

In this section we discuss the cavity identification algorithm. It is based on algorithms that solve the four combinatorial problems defined in the previous section. All the problems fall in the computational geometry area [8].

A. Algorithms’ Description

Due to space limitations, in this section we give an high level description of the algorithms (neither formal descriptions nor proofs for the properties underlying the algorithms are provided).

The problem PHYSICAL CAVITY can be solved by an iterative algorithm which uses operations like *intersection* and *difference* between polygons with holes, while the problem POLYGON APPROXIMATION can be solved by adapting the MiniMax method [9] for polygon approximation.

The problem POLYGON DECOMPOSITION can be solved as follows. In principle, to decompose a quasi-orthogonal polygon, it is possible to use four cutting directions: vertical (|), horizontal (—), left-oblique (╲), and right-oblique (╱). It can be shown that at most two out of four are allowable (in [10], it is shown that it is NP-hard to decompose a polygon if at least three cutting directions can be used). Moreover, it can be shown that the problem can be solved first by computing *all* the possible triangles and then by decomposing the remaining part. Notice that, the remaining part is an orthogonal polygon with holes that can be decomposed into rectangles only.

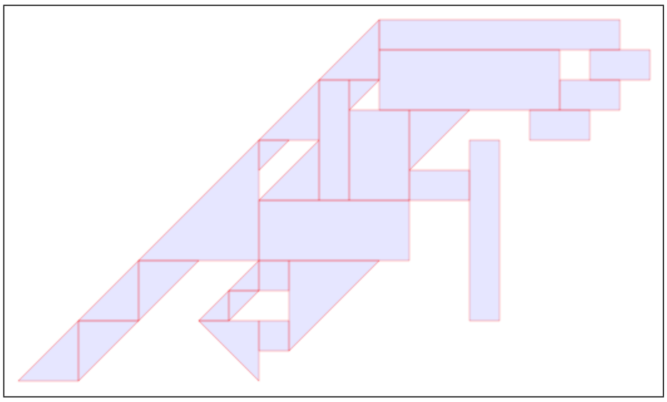

Fig. 3. Example of quasi-orthogonal polygon decomposition

In [11], Imai and Asano devised an optimum algorithm to decompose an orthogonal polygon into rectangles.

According to properties above and by using the algorithm in [11], we solved the problem of decomposing a quasi-orthogonal polygon p by the following steps:

1) compute all the triangles of p;
2) compute p' by removing from p all the triangles computed at Step 1;
3) complete the decomposition by applying the ImaiAsano algorithm to p';
4) check whether there are “long” rectangles; in case, cut them to fulfill the constraint on the max aspect ratio.

Fig. 3 shows an application of this decomposition algorithm. The last problem, CAVITY CONNECTIONS, can be solved by applying basic polygon operations.

B. Implementation Issues

All the algorithm briefly discussed in the previous section have been implemented by using CGAL [12]. CGAL is an open source project whose aim is to provide an easy access to efficient and reliable geometric algorithms in the form of a `C++` library. For instance, concerning data structures, we used the basic classes `Polygons_with_holes_2` and `Polygons_set_2` to model shapes and nets respectively; concerning computation, we exploited the `intersection()` and `difference()` methods of such classes. For solving the POLYGON DECOMPOSITION, instead of `Polygons_set_2`, we used `Arrangments_2` to model polygons with holes. Given a set C of planar curves, the arrangement $\mathcal{A}(C)$ is the subdivision of the plane into zero-dimensional, one-dimensional and two-dimensional cells, called vertices, edges and faces, respectively induced by the curves in C [13]. The use of arrangments allowed us to simplify the insertion of a chord into a polygon as a basic mechanism to perform a polygon decomposition.

C. The Final Software Architecture

It has been realized a software application, named Cavity Identification Tool (`CIT`), that implements the algorithms

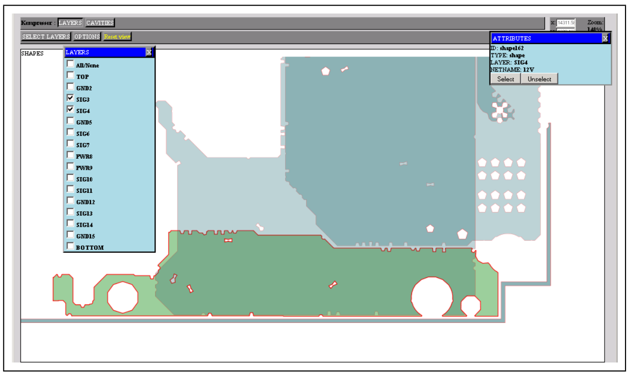

Fig. 4. The shapes' visualizer. A zooming on shapes at layers SIG3 and SIG4 only. The opacity property of SVG graphic elements allows to see the overlap between shapes.

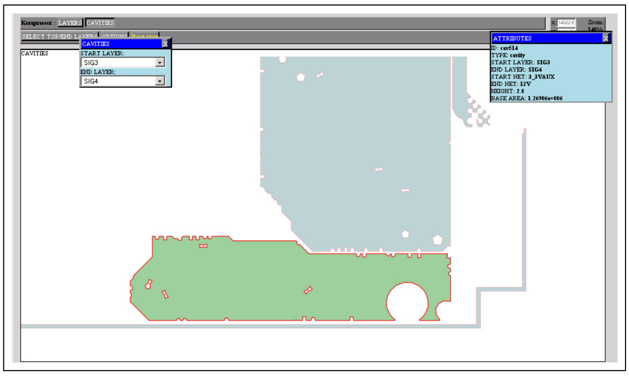

Fig. 5. The physical cavities' 2D visualizer. A zooming on cavities from layer SIG3 to layer SIG4. Focus on the green shape forming the top of a cavity.

described before and provides additional functionalities, such as shapes and cavities visualization.

`CIT` has been designed as a client-server application. The client is written in `Java` and is responsible for realizing a GUI that allow the user to provide the main input (all the requested board data) as well as additional parameters (for instance, the max aspect ratio for rectangles). The server is responsible for implementing all the `C++` algorithms based on CGAL.

Each time the server computes geometrical information (e.g., physical cavities), it sends to the client such information encoded in SVG [14] format. The client implements a geometric info *visualizer* that allows the user to have a promptly feedback about the performed computation. The client implements the SVG visualization and manipulation by exploiting the Batik [15] graphic toolkit. Figs. 4 and 5 give an idea about the visualization functionality of the tool.

The client-server communication has been realized by means of the XML-RPC [16] protocol.

V. Algorithm's Validation and Results

We performed an experiment by running the tool over a PCB consisting of: 16 layers, 221 nets, 498 shapes, and 16287 vias.

The import phase required 460.23 seconds (data imported from txt files extracted from a Cadence® Allegro® PCB editor project file). The execution of the module responsible for computing *all* the physical cavities produced the following output: 14685 physical cavities computed in 288.53 seconds. By running the same module with a threshold on the size about the area of the polygon at the top of a physical cavity, we get the result shown in Fig. 6.

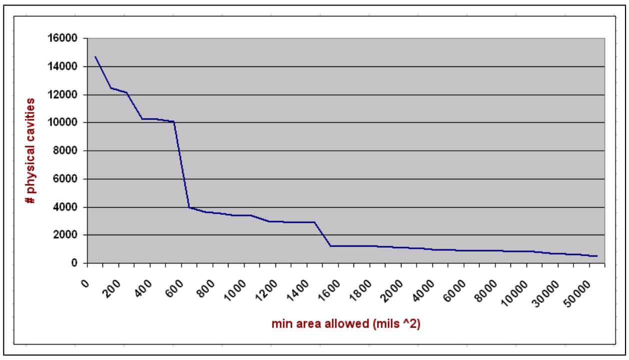

Fig. 6. Number of physical cavities by varying the min area allowed.

The experiment has been performed on a pc with the following main features: Intel Dual Core E2140 @ 2800 MHz, RAM 2GB DDR2 800 MHz, GNU/Linux Kernel 2.6 x86_64, GCC 4.2.3.

References

[1] J. Knighten, B. Archambeault, J. Fan, G. Selli, L. Xue, S. Connor, and J. Drewniak, "PDN Design Strategies: I. Ceramic SMT Decoupling Capacitors. Does Location Matter?" *IEEE EMC Society Newsletter*, vol. 208, pp. 56–67, 2006.
[2] M. Swaminathan, K. Joungho, I. Novak, and J. Libous, "Power distribution networks for system-on-package: status and challenges," *IEEE Trans. on Advanced Packaging*, vol. 27, no. 2, pp. 286–300, 2004.
[3] T. Hubing and J. Drewniak and T. Van Doren and D. Hockanson, "Power bus decoupling on multilayer printed circuit boards," *IEEE Trans. on EMC*, vol. 37, no. 2, pp. 155–162, 1995.
[4] R.M. Rizzi and G. Selli. G. Antonini. A. Orlandi and J.L. Drewniak, "A systematic approach for transient analysis of power distribution networks with active device," in *EMC Europe Workshop*, 2007.
[5] N. Na, J. Choi, S. Chun, M. Swaminatham, and J. Srinivasan, "Modeling and transient Simulation of planes in electronic packages," *IEEE Trans. on Advanced Packaging*, vol. 23, no. 3, pp. 340–352, 2000.
[6] C. T. Lei, R. W. Techentin, and B. K. Gilbert, "High-frequency characterization of power/ground plane structures," *IEEE Trans. Microwave Theory Tech.*, vol. 47, pp. 562–569, 1999.
[7] "Agilent technologies, advanced design sistem 2008 user manual," `http://eesof.tm.agilent.com`.
[8] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. Spinger-Verlag, 1985.
[9] Y. Kurozumi and W. A. Davis, "Polygonal approximation by the minimax method," *Computer Graphics and Image Processing*, vol. 19, pp. 248–264, 1982.
[10] A. Lingas and V. Soltan, "Minimum convex partition of a polygon with holes by cuts in given directions," *Theory Comput. Systems*, vol. 31, pp. 507–538, 1998.
[11] H. Imai and T. Asano, "Efficient algorithms for geometric graph search problems," *SIAM J. on Computing*, vol. 15, no. 2, pp. 478–494, 1986.
[12] "CGAL, Computational Geometry Algorithms Library," `http://www.cgal.org`.
[13] P. K. Agarwal and M. Sharir, *Handbook of Computational Geometry*. Elsevier, 2000, ch. Arrangements and their applications, pp. 49–119.
[14] "SVG, Scalable Vector Graphics Library," `http://www.w3.org/Graphics/SVG/`.
[15] "Batik SVG toolkit," `http://xmlgraphics.apache.org/batik/`.
[16] "XML-RPC," `http://www.xmlrpc.com/`.