WEB-Based Automatic Layout Generation Tool
with Visualization Features

João D. Togni* 
togni@inf.ufrgs.br
André I. Reis 
andreis@inf.ufrgs.br
R. P. Ribas 
rpribas@inf.ufrgs.br
Instituto de Informática - UFRGS 
Av. Bento Gonçalves, 9500 - CxP.15064 
CEP91501-970 Porto Alegre - RS, Brazil

Abstract

This paper presents a WEB-based manager tool for layout generation. Layout generators for MEMS, digital and analog circuits are provided. This manager allows the user to choose a particular device and set respective specific parameters for the automatic layout construction. This paper presents also two simple examples of generators created for demonstration of the functionality of the manager. Besides the top view layout illustration, such tool offers visualization features such as two-dimensional (2D) cross-section view, according to a cut line defined by the user, and three-dimensional (3D) representation, in VRML format. The tool is easily portable since it has been completely developed in Java platform.

Resumo

Este artigo apresenta uma ferramenta de gerência para geração de leiaute baseada em WEB. Geradores de layout para circuitos digitais e analógicos, bem como para micro-sistemas ou MEMS podem ser incluídos neste ambiente. Este artigo também apresenta dois geradores de células para exemplificar e demonstrar a funcionalidade do gerenciador sugerido. Além da representação em duas dimensões comumente usada para ilustrar layouts de circuitos integrados, esta ferramenta permite vistas em corte (2D) do layout a partir de uma linha de corte definida pelo usuário. Também existe a possibilidade de extração do layout em formato VRML para ser visualizado em três dimensões (3D) através de browser apropriados. Por fim, a ferramenta é facilmente portável uma vez que ela está sendo totalmente desenvolvida em linguagem Java.

*Author supported by PROPESQ-UFRGS Brazilian scholarships.
1. Introduction

Most physical synthesis tools are cell-based, in the sense that such tools generate a final layout for a circuit or module by placing and routing leaf cells [1]. Leaf cells normally came from a pre-designed cell library or from specialized cell generators. In the case of cell generators, a set of parameters are required for the specific layout geometries and defined by the final user. This paper presents a WEB-based manager tool for leaf cell generation developed in Java platform.

Even when considering simple leaf cells, like a NAND gate, there are different targets to be considered depending on the use of this cell. For instance different users would need a NAND optimized for area, speed or power consumption. In a traditional approach, cells are pre-designed and made available in a cell library. However, due to the design and characterization cost for each cell, the number of cells available in a cell library is limited. The solution to reduce the design and characterization cost is to use automatic generators. Even if the quality of the generated cells is not as good as those generated handily, the availability of a greater number of cells certainly compensate this drawback.

Another advantage in using automatic cell generators is the technology migration cost. As technology changes become more and more frequent, it is necessary to redesign cell libraries for every target process. Also the time-to-market issue is gaining importance from an economic point of view. For this reason, automatic cell generators play an essential role in circuit design.

Leaf cell generation is not limited to digital cells, analog circuits as well as MEMS devices can also be provided, which require a great deal of expertise for one handle their layout building.

The manager tool also includes didactic visualization features. These features are an effort to make the designed cells more understandable to the user. The goal is to teach layout structure by showing to the user 2D cuts and 3D views of the generated cells. For these reasons the manager includes translation from layout to 2D cuts and 3D views. Some works that translate layouts to 3D views [2] and 2D cut views [3] were already presented, but they were not integrated in a single tool.

2. Tool Objectives

This project has three main goals: to offer a platform independent freeware environment for layout synthesis, to offer 2D and 3D didactic visualization features, and to offer extensibility to plug new layout generators at this manager.

As far as the authors know, there is no freeware, platform independent, environment for automatic cell generation available. The availability of such environment would provide designers with a tool for automatic (re)construction of standard cell libraries. Moreover, many different generators, each one with its own generic parameters to generate specialized layouts like digital cells, MEMS, etc. This tool will help to perform technology migration. When the target process change, the generators can retarget cells to the new technology.

Visualization tools providing 2D, 3D and cut views are a key point to understand the structure of CMOS technologies. Therefore, these tools should be used in the education of people that will work with physical design of leaf cells. For MEMS this visualization can be even more important, by offering a view close to real implementation of the circuit model working.

As there are many different possible layout generators, the environment should stay open to allow new generators to be plugged in the environment as well allowing new technologies to be configured for the existing generators. This ability gives to the software a dynamic evolution, because the environment can aggregate many generators and processes, offering thus a suite of cell development tools not attached to a specific process.

3. Tool Features

The implementation of the proposed tool involved many decisions, being the first one the choice of the implementation language. Java is a new and modern programming language that is having an increasing grown due to portability needs. Today Java is largely utilized in the internet and in intranets. Two important
characteristics of Java are the graphical portability (of interfaces) and the reusability given by object orientation. The major Java disadvantage is the low speed due to code interpretation: codes created in Java are in general very slower than codes created in other languages, specially compiled ones. But this situation may change with the availability of new Java compilers and interpreters. Although Java is a slow language it supports integration with other languages through some patterns of distributed programming such as CORBA. This resource can be used to translate a slow part of the code to other language and integrate it with the Java part of the software. In conclusion, Java has been elected because it offers a portable graphic interface, it is largely used, and it also offers the possibility of code reuse.

By using Java it is possible to design three kinds of software: stand alone, applets without server, and client-server software:

- Applets without a server are applications that run over the net (they may also be local) and are attached to a WEB page. This kind of software has many security restrictions to avoid the remote executing of intrusive programs, for instance the impossibility of saving files. Another property of this kind of software is that it can easily be translated into a stand-alone program.

- The client-server approach runs with one or more server software offering services (files, processing, etc.) to the client software, which are frequently more than one. This makes the installation, configuration and maintenance more expensive because it is necessary to maintain the server and the client working well. The advantages of this approach are the absence of security restrictions, because clients can be authenticated, and no need of file replication, because the configuration files are located in the server side.

- The last approach is the stand-alone. In this approach, the software will run on a stand-alone machine without the need of a net neither of a server to be completely useful. This approach is the simplest to implement because there is no need to maintain other software but the stand-alone software. In some cases this approach makes the data replicated over a net because every software installation must have all the configuration data to run.

The software that was implemented is based on a stand-alone version that is able to get data over the net, for instance, processes and generators. This way it is easy to maintain and avoid file replication.

3.1 Layout Generator Management

This feature enables the user to chose the layout generator to be used. Figure 5 and 6 shows the window of the manager tool. The chosen generator was a nand from group logic and subgroup simple. Other group implemented was the MEMS group, in which the user can generate comb-drive devices from geometric parameters. This window offers the user the possibility to generate the cell once the parameters are set up. It also offers standard 2D view of the generated cell, as shown in the figures. Besides, the manager window is the interface to call other features (2D cut section view, 3D view, CIF code generation), and to see the log viewer.

3.2 Log viewer

The log viewer is a window where messages of all events occurring in the software are displayed. This feature helps the user to see the software processing flow in order to verify the actions that were already performed. Moreover, the log viewer shows the eventual errors and bugs that occur in the process. The log viewer appears as a sub-window in the manager window.

3.3 Parameter configuration

The parameters needed for layout generation configuration are specific to each generator. Every selected generator will recreate the panel of data acquisition to show its own parameters. These parameters appear in the manager window according to the selected layout generator. This feature adds to the software the flexibility needed to deal with different kinds of tools, by passing to the generators the control of the acquisition and processing of the generic parameters needed to create the cell.
3.4 Process Configuration

The process configuration window has the objective of making the software configurable to any process by saving the information needed to configure the whole process layer by layer. This way, a user with knowledge about a layout building process can configure the software to support the desired processes.

A set of information are required for the tool, not only traditional process data. The information about the order of deposition of every layer in the process, needed to describe how to build a 2D cut section, is a information saved with a process. The software is capable of building a 2D cut view with the information on the deposition order and other information about the relations and different kinds of layers.

![Figure 1: Process Configuration.](image)

Many kinds of relations between the layers can be expressed in the configuration. A layer may be generated of the result of a union of other layers, a intersection with other layers and a subtraction of other layers, and a combination of this functions. Another configuration feature is the information of the 2D cut view, a layer can be a deposited layer, a implanted layer, or a layer that appears in the normal 2D view by not in the cut view. As examples of these features, in the configuration of the AMS process the pDiff layer is a copy of Diff, subtracted the Poly and intersected with the N-Tub layer, and the Hole is a layer that appear in the 2D normal view but not in the cut view. These features offers a very powerful mechanism of configuring any kind of integrated circuit conception process assured by its flexibility.

The 3D view is build by the same way, information about the order and the sizes of each layer are saved. The process configuration includes also a color palette to allow users to configure layer coloring in an easy way.
3.5 Cut Section View

The possibility of a cut section view is a good help to a student learning the process of creation of integrated circuits. This feature can be also found in another related work integrated to a commercial software [3]. The 2D cut view is specially useful in the case of MEMS, because MEMS design involves also many mechanical considerations in addition to the normal structure of usual fabrication technologies. The cut view is made with the information saved in the configuration process. The real deposition that occurs in normal circuit process is shown by the software.

![Cut Section View](image)

**Figure 2:** Cut section view of a Nand 2 ports.

3.6 VRML Conversion

The virtual Reality Modeling Language (VRML) is a standard language for 3D modeling, used to build virtual reality worlds. This language allows descriptions of 3D geometric objects, and the version 2.0 of this language supports interaction with the users. Another feature of this version is that it includes the 4th dimension of time that allows the addition of movements in the modeled 3D worlds.

There are several possible applications using this language in the field of physical layout design and simulation. This is a powerful tool for the MEMS designers that want to see their structures working, as in the real world. This is also a powerful tool for designers and students that want to browse in a good model of a layout. Not only is possible to make the 3D view of a transistor but the current flow may be shown by using the time modeling in the VRML [4]. VRML becomes a standard language for 3D description. In the field of integrated circuits, it was used already to represent circuit layouts with good approximation [2].

The current implementation of the tool covers only the static conversion of a layout to VRML, this feature is especially useful to teach/learn the layout structure of cells. The code generated by the tool can be viewed in any VRML 2.0 browser.
3.7 CIF coding

The ultimate goal of any layout generator is to produce useful layouts. For this reason the manager is able to output the leaf cells in CIF format [5]. This format is accepted by professional design automation tools as well for foundries that produce the integrated circuits. This feature enables the use of the leaf cells for placement, routing in order to design bigger circuits as well to proceed to an electrical evaluation in order to let this information available if the cell is used to compose a library.

4. How to extend the software

There are two ways in which this software is extensible. First one is extending its internal functions and tools. The second one is the addition of new layout generators. This could be done by any layout developer that knows Java programming. An interface to offer this easy extension was made and will be explained below.
The manager works in two steps. The first one mounts the manager graphical interface and after that the second step loads the generators that are in a specific directory. The generators are extensions of a generator class having virtual methods that must be implemented in the extended class (the implementations of each generator). As an example the inverter generator shown in the pictures of this paper is an extension of the class generator and implements some methods that get parameters, return its name, group, subgroup, and implements the generate method that constructs and returns the polygons representing the inverter layout. The group and subgroup fields in the manager are used to organize the access to generators providing best browsing.

5. Examples of generators

Generators are in essence programs that, with a given set of design parameters, can obtain relations of size and distances between polygons. Two examples of generators were developed to show the potentiality of the mechanism offered by the software.

The first one is a common MEMS design, a Comb-Drive that has many generic parameters to be set up. The parameters are: the number of teeth that the device will have, the length of the teeth, the length and weight of the mass and the length of the elastic stems. All the parameters are checked before the generation of the device to avoid rule errors and assure the correctness of the layout generated.

![Applet Viewer: wfg/MainApplet](image)

The second example is a simple NAnd gate where the width and the length of the n and p transistors can be set up by the designer. This generator allows the generation of Nands with different physical characteristics. Some characteristics may vary automatically according to other parameters, for instance the number of contacts will vary with the length of the transistor.

![Figure 5: the Comb-Drive.](image)
One of the objectives of this software is to show that other layout generators with more powerful features can be created, for instance a Nand that is made with electrical parameters like the fanout or maximal current desired, or with parameters of transparency of metal 1 and 2 and many other interesting layouts can be made, visualized and exported to other commercial tools.

6. Conclusions and Future Works

This paper presented a manager tool for leaf cell layout generation. It manages several different layout generators and includes visualization facilities for 2D, 3D and cut section views of generated layouts. These different views of the generated layouts are of special interest for beginners to understand the physical structure of CMOS technologies. Besides, the availability of different automatic layout generators is useful for experienced designers to get more speed and flexibility in the process of leaf cell generation.

Future work can include the addition of many other features to those already implemented in this version of the software. For instance:

- Build a physical synthesizer that convert logic equations to layouts.
- Extend VRML to support movements of MEMS
- Support MEMS corrosion simulation
- Add transparency and patterns to layers
- Integration with commercial and university software

References


