A Layout-Educated Analog Design Flow

Vincent Bourguet, Laurent De Lamarre and Marie-Minerve Louërat
University of Paris VI, LIP6-ASIM Laboratory,
4, Place Jussieu, 75252 Paris, France
Vincent.Bourguet@lip6.fr

***Abstract*— In this paper, we present a new flow for analog design automation. It is an electrical and layout co-design flow which is based on the precise knowledge of the process electrical models and layout. This flow is as fast as a layout-aware flow since the layout informations aren't given by an external extractor, but are measured while generating the layout. But it is also as accurate as a layout-inclusive flow seeing that it goes until the layout generation. It is so a layout-educated flow which, by a strong separation from the target process, allows the technology migration for commonly used designs. All the expertise of the proposed flow is contained in the device generators, which are the leaf cells of the flow hierarchy**

I. Introduction

Many papers state the increasing part of analog circuitry required in the design of mixed-signal integrated systems on chip [1]. Although there has been a lot of improvement in the topic of analog CAD, analog designers are still reluctant to use design automation, since there are always some steps of the design that they can not control. Today, migrating an existing function to a new set of specification for the same process, or to a new process with the same or new specifications is considered as a very difficult and major issue. Furthermore, with the growing part of analog circuitry in manufactured chip, it is compulsory to find how to reuse existing design and knowledge, which means that designers are required to define and use some kind of analog libraries of commonly used analog functions [2]. However, specification and technology migration of an analog circuit is still considered as requiring a lot of designer's tuning.

Here we present simple analog function generators, called devices. With a library of such generators at his disposal, the analog designer is able to create complex hierarchical function generators. Although these devices greatly facilitate the design by automating it, they still provide the designer the entire control of the circuit generation. We will also explain how these generators can be independent of the technological process thus enabling process and specification migration.

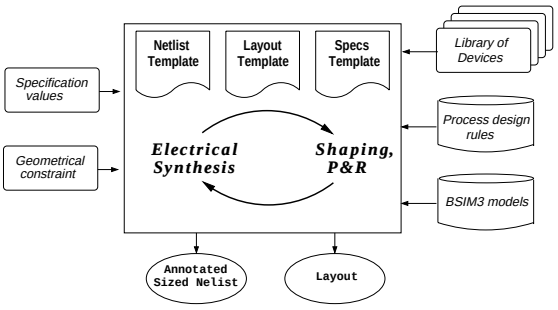

Fig. 1. The proposed design flow

II. Analog IC Design Flow

A. State of the Art

Studies on analog design of integrated circuits agree that layout information has to be taken into account as soon as possible in the electrical sizing phase of the circuit [1], [3]. Different approaches are used to meet this goal. They differ by the way they estimate the parasitics resulting from the layout (i.e. knowledge based estimation, layout generation with or without external extractor, look up tables of pre-characterized devices) and by the way they take advantage of the parasitics information in the electrical sizing phase. Our device generators allow the design to be as fast as layout-aware flows [4]–[6] since the layout informations aren't given by an external extractor, but are measured while generating the layout. Meanwhile, it can be as accurate as the layout-inclusive [7] flow seeing that the layout generation is actually performed in the sizing loops.

B. The Proposed Flow

According to our flow (fig. 1), the circuit is described by a functional tree. The nodes of this tree are called modules and are created by module generators. A module is based on two hierarchical templates : an electrical one (the netlist template) and a physical one (the layout template). The netlist template, as shown on fig 2-a., is the unsized netlist of the design, described in a hierarchical way. The leaf cells of this electrical tree are the devices. All electrical pertinent components are enclosed in the devices. In order to predict parasitics resulting from layout, we have chosen an approach using layout templates. The relative placement of instances inside a module is described by a container tree. A container defines the layout template of the module and is composed of abutted containers placed besides each other in a specific order (fig 2-b). There exists vertical and horizontal containers. The leaf cells of the container tree correspond to the devices. Otherwise, there is an injective relation between the electrical tree and the physical tree. For each node in the electrical tree, there exists a corresponding container in the physical one, but there could be more containers than module since the containers are used to describe the layout topology inside a single module.

The first step of the proposed design flow consists in creating both the electrical and physical templates. Then, the second step is the design space exploration [8], [9]. Here we check that the specifications can be reached, and if possible, we compute parameters of the sub-level generators to meet them. To achieve this computation, we define a third template containing a list of selected parameters required to perform the electrical sizing. This template is called the specification template (cf. section III.-B.). Once the parameter values are obtained, we can achieve the third step which shapes the circuit layout. In fact, as we will explain in section III.-C., a design may present

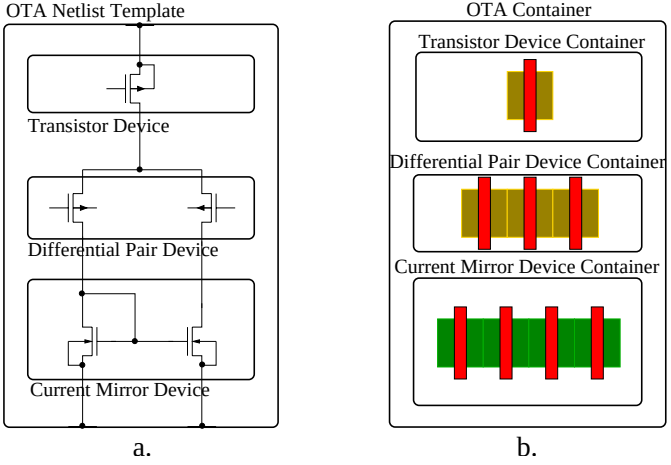

Fig. 2. Both netlist and layout templates for a simple OTA

several layouts with different shapes. Therefore, the shaping step consists in calculating all the geometrical dimensions that can be taken by a module. Each of these shapes involve different parasitics, and moreover, different performances. In order to take that into account, the layout of every device shape is generated. However, while drawing, the layout engine also measures the parasitics. After the shape selection by a geometrical constraint, the fourth step achieves the circuit placement and routing.

Although our proposed design flow goes until the layout generation, parasitics aren't computed by an external extractor. The synthesis loop is significantly improved by that, and this flow can take place at the frontier of a layout-aware and a layout-inclusive design flow.

III. Designing With Device Generators

A. Introduction

We have seen that the proposed design flow of an analog circuit is based on an existing library of devices which are made up of both electrical and physical views. These devices are not necessarily electrical basic elements such as transistor, capacitance or resistor. They can also be several elements tied by strong matching constraints, that can only be respected by a dedicated layout [10]–[12]. Typically, a differential pair is a device.

B. Electrical Sizing

The proposed method allows the designer to keep control on the sizing of the circuit. Our device generators integrate a set of functions which return the electrical characteristics of a device. Then by questioning the generator on the value of these parameters, the designer has the possibility to write a synthesis procedure, based on his own expertise of the circuit. There are two kinds of functions:

direct functions return the electrical performances of a device. The arguments taken are the biasing voltages and the dimensions of the devices

inverse functions that are able to return a dimension or a biasing voltage in order to satisfy a constraint on the circuit

The electrical synthesis step is seen as a negotiation between two levels of the module hierarchy. After specifying a number of constraints on the generator's parameters, the higher level can ask for the value of the missing parameters. An exception request is threw when the set of specifications is unrealizable. Caught by the upper level, it is decoded to know which parameter failed. For example the width of a transistor can be computed for a given drain current, length and bias voltages. If the drain current is too high and consequently the largest drawable width cannot conduct it, the upper level is informed. The latter may try another configuration. The list of valid questions for the transistor is given in table 1. To calculate the width for a given drain current, the designer may fix the gate-source voltage V_{gs} or the overdrive voltage $V_{OD} = V_{gs} - V_T$, where V_T is the threshold voltage. A function that computes both W and V_{gs} to achieve I_{ds} and V_{OD} is implemented in the transistor generator. Our library eases the task of the designer by proposing some useful built-in functions.

dimensions	
W	gate width
L	gate length
biasing voltage	
VGS	gate-source voltage
static parameters	
VTH	threshold voltage
VDSAT	saturation voltage
IDS	drain current
Qx	charge at node x
STH	thermal noise density
S1F	$\frac{1}{f}$ noise density
small signal parameters	
GM	gate transconductance
GDS	drain conductance
GMB	substrate transconductance
Cxy	capacitor between nodes x et y

TABLE I
Valid questions for the transistor generator

From the designer's point of view, the manipulation of the basic components is process independent. The same model's equations than those used in electrical simulator (BSIM3v3) have been implemented in our device generators. We can directly exploit the values of the technological parameter file, to calculate accurately the electrical characteristics of a component. As we can see in figure 3, for the gate transconductance, the parameter's values calculated by our functions match the results of the electrical simulator.

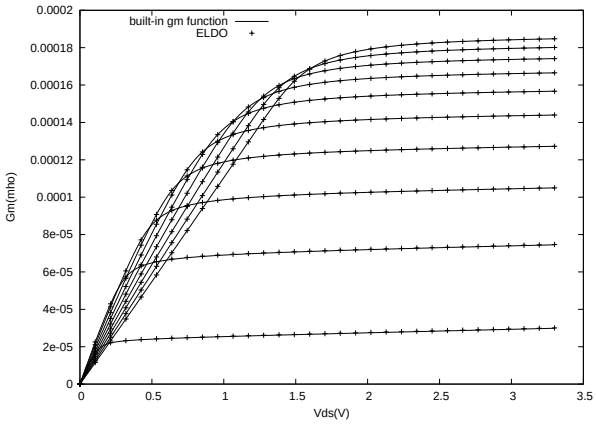

Fig. 3. Gate Transconductance computation compared to Eldo results

Parasitics elements are accurately estimated during this step since it gets the value of all geometrical informations, set by the layout generation step. For a transistor, the number of fingers, the diffusion area and perimeter are taken into account during the synthesis process.

C. Shaping and Layout Generation

One of the main difficulties in analog layout automation is to manage the wide range of aspect ratios that can be presented by a same design. In fact, as illustrated on fig 4, the layout area of a specific analog function can easily double between two sets of specifications. On that figure, two layouts of a differential pair are shown. The first results from a desired $I_{ds} = 0.1mA$, the second from $I_{ds} = 0.2mA$. Of course, migrating this function on a new process leads to the same problem.

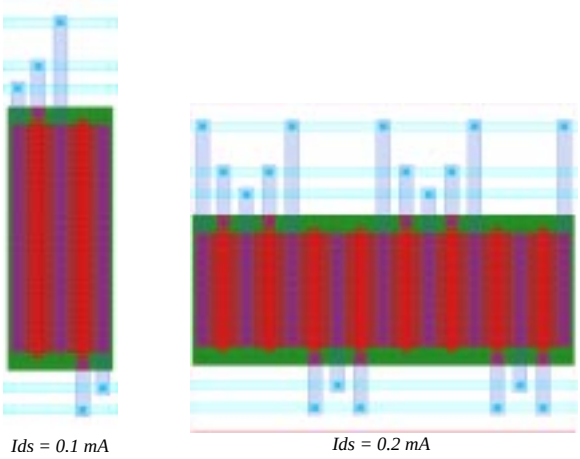

Fig. 4. Inter-digitated differential pair layout from two different I_{ds}

The device generator presented here tries to give an answer to that problem by having several layout aspect ratio for electrical known specifications on a target process. For example, a transistor with a gate width W can be drawn as M parallel transistors with a gate width $W_f = W/M$ while W_f respects the minimal width of the process. This technique is well known as transistor folding [10]–[12]. These different aspect ratios are represented by a shape function which gives for example the width as function of the height, as presented on fig 5. Unit is the λ with $\lambda = 0.5\mu m$. Thanks to that, the designer can obtain a compact layout.

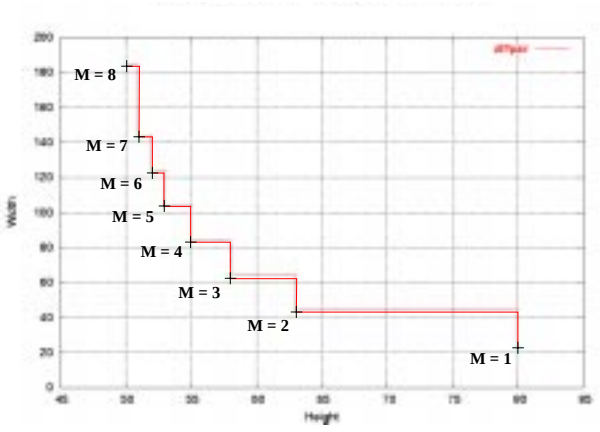

Fig. 5. Shape function of an inter-digitated differential pair

The devices achieve this shaping by building the layout as an abutment of simple patterns. If we consider a folded transistor, patterns would be each of the "little" parallel transistors, also called transistor fingers. If we now consider a capacitor matrix, pattern would be a unit capacitor. Thus, the placement of these patterns gives the global layout aspect ratio. Fig. 6 shows two different aspect ratios -that means two different pattern placements- of the same transistor, using respectively two and four patterns.

Finally, all device shapes are stored in the shape function which can be bottom-up propagated through the hierarchy to compute higher level modules shape functions. Thus, the whole design layout can also present several shapes. Although the final shape is typically selected by a geometrical constraint, the designer still can impose the shape he wants for one or several devices.

1) Pattern Generators: The pattern generators are very complex generators since they enclose all the layout expertise of the design. Moreover, they are required to be respectful of the process rules, and to guaranty maximal accuracy for a reliable analog layout. We aim to clearly divide this complex achievement into simpler tasks. The idea is that a pattern is composed by rectangles and each rectangle can be defined by five variables : X, Y, DX, DY and the Layer. With this simple set of variables, the whole pattern layout can be drawn. Nevertheless, we have to define the interface between the target process rules, which is a very complex set of rules, and the pattern's simple set of variables. This is done by a set of functions called MAPI. However, as it is explained in section III.-C.-2), the MAPI has another issue which consists in separating patterns -and thus device- and the target process. This allows to handle technology migration.

Let's see how MAPI compute the X, Y, DX, DY, and Layer variables. First, we have to understand two things :

1. Each pattern generator has his own MAPI at his disposal. That means that the MAPI is aware of the pattern drawing.
2. Pattern generators are device specifics. That implies that they are designed to be used in a known instantiating environment. For example, a pattern knows which kind of other pattern is abutted to it.

With this knowledge, the MAPI exactly knows which kind of process rules it needs. Then, it asks for the selected process informations, and achieves some computing such as maximums,

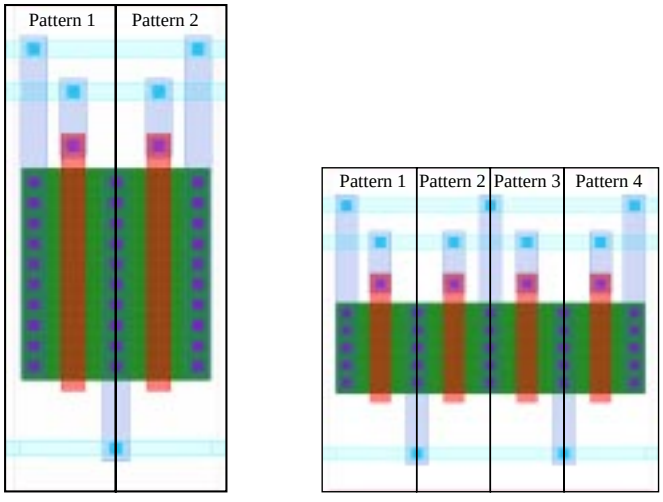

Fig. 6. Pattern placement

or sums in order to get X, Y, DX, DY and Layer of each rectangle.

2) Process Informations Access: All along the generation, informations about the process are required. Meanwhile, we saw that a good separation with the technology is important to ensure the migration. As the pattern have their MAPI, the device has a DAPI. The DAPI gives the device a direct access to the process informations. For example, it can be useful to get the W_{min}, which is involved in both electrical synthesis and shaping computation.

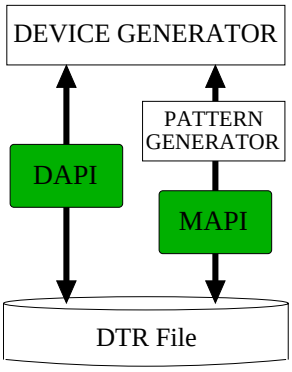

Fig. 7. Internal hierarchy of a device

Therefore, the generation is based on a description of the target process. The difficulty resides in the very heterogeneous information that is needed. We resolved this problem by using a Device Technological Rules file (DTR) to describe a selected process. The syntax of this file allows large freedom about the enclosed information and can easily be updated. Figure 7 shows the finally involved hierarchy in a device generator.

3) Parasitics Calculation: In order to perform an electrical and layout co-design, it is compulsory to communicate informations about the parasitics resulting from the layout to the electrical sizing engine. To achieve this, the pattern generators return informations about the layout they draw. For example, the transistor device is informed about the W, L and diffusion dimensions of each finger. Then, the device takes into account the information received from each of its patterns to compute global information. Therefore, the design space explorer has a back-annotated netlist at his disposal, resulting from actual layout and the electrical models can estimate the resulting performances. This enables to adjust the sizing parameters to meet the specifications.

IV. Application : Simple OTA

We will now illustrate an example for the generation of a simple OTA. This one is represented by a single-level hierarchy. In fact, the OTA module instantiates a current-mirror device, a differential pair device and a transistor device (fig. 2). In this example, we chose to show what happens when the GBW changes for same charge, supply and common mode voltages. The geometrical constraint is still done by $DY_{max} = 300\mu m$. Figure 8 shows the generated layout for GBW = 65MHz. Figure 9 shows that GBW = 130MHz.

V. Conclusion

We proposed a set of device generators which allows the automation of analog circuit design. These devices can be integrated in a flow which is as accurate as a layout-inclusive flow, but also as fast as a layout-aware flow. In fact, the parasitics resulting from the layout are measured and then, computed with very accurate BSIM3v3 models. Moreover, the layout generation engine allows the designer to get a compact layout on different target technologies, since it performs a shaping step. However, the designer keeps controlling every step of the generation.

References

[1] Gielen and Rutenbar. Computer Aided Design of Analog and Mixed-signal Integrated Circuits. *Proc IEEE*, 88(12):1825–1852, December 2000.

[2] Hershenson, Boyd, and Lee. GPCAD : A Tool for CMOS Op-Amp Synthesis. *ICCAD*, pages 296–303, 1998.

[3] de Ranter, Van der Plas , Steyaert, Gielen and Sansen. CYCLONE : Automated Design and Layout of RF LC-Oscillators. *IEEE TCAD*, pages 1161–1170, October 2002.

[4] Tang, Zhang, and Doboli. Layout-Aware Analog System Synthesis Based on Symbolic Layout Description and Combined Block Parameter Exploration, Placement and Global Routing. *Proc. IEEE Symposium on VLSI, ISVLSI'03*, 2003.

[5] Dessouky, Louërat, and Porte. Layout-Oriented Synthesis of High Performance Analog Circuits. *Design AUtomation and Test in Europe, DATE*, pages 53–57, March 2000.

[6] Agarwal, Sampath, Yelamanchili, and Vemuri. Accurate Estimation of Parasitic Capacitances in Analog Circuits. *Design Automation and Test in Europe, DATE*, pages 1365–1366, February 2004.

[7] Ranjan, Verhaegen, Agarwal, Sampath, Vemuri, and Gielen. Fast, Layout-Inclusive Analog Circuit Synthesis using Pre-Compiled Parasitic-Aware Symbolic Performance Models. *Design AUtomation and Test in Europe, DATE*, pages 604–609, February 2004.

[8] Hershenson. Efficient Description of the Design Space of Analog Circuits. *DAC 2003*, pages 970–973, June 2003.

[9] De Smedt and Gielen. WATSON : Design Space Boundary Exploration and Model Generation for Analog and RF IC Design. *Proc IEEE*, 22(2):213–224, February 2003.

[10] Naiknaware and Fiez. Automated Hierarchical CMOS Analog Circuit Stack Generation with Intramodule Connectivity and Matching Considerations. *IEEE JSSC*, pages 304–317, March 1999.

[11] Pelgrom, Duinmijer, and Welbers. Matching Properties of MOS Transistors. *IEEE JSSC*, pages 1433–1440, October 1989.

[12] Hastings. *The Art of Analog Layout*. Prentice Hall, 2001.

Specifications	
V_{dd}	3.3V
V_{ss}	0V
C_{load}	5.0pF
V_{ICM}	1.2V
V_{OCM}	1.2V
GBW	65MHz
Results	
GBW	64.99MHz

Fig. 8. Simple OTA : GBW = 65MHz

Specifications	
V_{dd}	3.3V
V_{ss}	0V
C_{load}	5.0pF
V_{ICM}	1.2V
V_{OCM}	1.2V
GBW	130MHz
Results	
GBW	129.98MHz

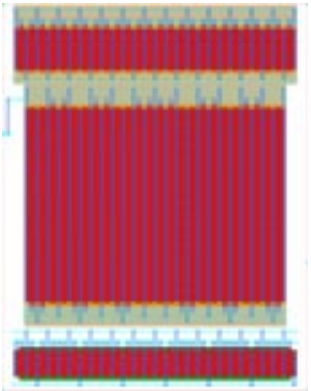

Fig. 9. Simple OTA : GBW = 130MHz