Analog Device Layout Procedural Generation

Jean-François Naviner*, Patrick Loumeau**, Omid Oliaei**,
Hervé Petit**, Lírida Alves de Barros Naviner*, Elmar Melcher*

(*) Departamento de Engenharia Elétrica - CCT
Universidade Federal da Paraíba
Av. Aprigio Veloso, 882, Bodocongó
58105 - 970, Campina Grande, Pb, Brazil
naviner@dsc.ufpb.br, lnaviner@dsc.ufpb.br, elmar@dee.ufpb.br
Tel. ~55 83 310 1135- Fax. ~55 83 310 1418

(**) Département Électronique
Ecole Nationale Supérieure des Télécommunications
46, rue Barrault - 75 634 Paris cedex 13, France
loumeau@elec.enst.fr, oliaei@elec.enst.fr, petit@elec.enst.fr
Tel. ~33 1 45 81 78 43 - Fax. ~33 1 45 80 40 36

Abstract:

To reach the desired performances, the design of analog circuits requires most often a very careful sizing and then layout synthesis. To address this task, an original set of analog device layout generators has been designed to be used in the context of an existing commercial framework. A new formalism has been developed to define easily new generators and to fit them in accordance to the technological rules. The set of generators covers the most common devices: resistors, capacitors, inductors, various transistor shapes, differential pairs and current sources. In this paper, the context and the objectives of this work are described. The formalism developed to define a technology and the generators is shown. Finally, three examples of device layout generators are presented. This set of generators has been successfully used to design several circuits.

I. Introduction

During the last few years, the synthesis tools for analog or mixed-signal integrated circuits (ICs) have achieved a lot of significant improvements [1]. There are several factors increasing this tendency:

- To perform a complete system integration on a chip, the implementation of analog and digital parts on a same circuit is required. The very competitive market of microelectronics induces the need of methods and tools to design all parts of the system. A lack of tools for one part means a significant overcost for the final product.
- A large variety of mature tools for digital circuit design gives a huge productivity gain without counterpart in the analog domain where often a circuit design is still based on full custom methods using only some simulation and analysis tools. Some methods and algorithms used to solve digital problems may often be adapted and partially re-used for other applications.
- The number of parameters which may influence the performances or more generally the implemented function itself is big for analog circuits. The low cost computation power today available simplifies the development and use of specific algorithms to solve complex problems of analog design in a reasonable time.

The analog layout synthesis addresses two different problems: the cell layout synthesis which consists in the creation of masks from a transistor-level schematic under a set of constraints and the assembly of cells or macrocells, that is, the placement and routing of the whole analog circuit. This paper is focused on the study of analog cell synthesis.

Analog cell layout synthesis aims to produce masks under a set of constraints and desired performances. The result should be as good as a full custom design realized by an expert designer, however with a much shorter development time. In the following section, the main approaches proposed to reach this objective are presented.

Then, the analog cell layout generator approach is discussed.

A set of novel and efficient cell layout generators has been developed. The layout strategy and the context of the development are presented. Finally, three examples are described.

II. Analog layout strategies

The approaches to design analog layout have evolved from nearly digital strategies to analog performances oriented strategies. The first approaches developed were based on some analog device generators and place-and-route tools derived from the existing algorithms implemented for digital circuits [2]. Another strategy was to develop basic cell generator programs based on a rich device generator library. When only few changes (orientation, sizing, etc.) have to be realized in the layout, the generation is fast and satisfactory [3]. A considerable designer effort is generally required to perform the assembly of the generated structures in order to respect the analog constraints which are not directly included in the synthesis process. A strong interactivity is then necessary. The migration of a rich generator library is also an heavy task.

More recently, various tools have been developed including analog properties into the synthesis of the layout [4-6]. Generators are also used but only for very basic structures. Complex structures are constructed with powerful algorithms during the layout synthesis process avoiding *a priori* choices. Based on a small set of basic generators, they are sometimes unable to reach some optimized configurations. In some critical applications, the use of these optimized device layouts is necessary. The goal of these tools is to complete in a fully automated way the optimized synthesis of an analog circuit from a high level set of constraints and specifications. One consequence is that interactivity should not be necessary. In fact, some algorithms used (simulated annealing for example) do not allow human interaction.

All strategies are therefore based on the use of generators, simple or complex depending mostly on the assembly algorithms. Like some recently described systems, the work presented in this paper uses a set of generators of medium complexity: optimized shaped transistors, basic active or passive devices [7-8].

III. Parameterized cells under the Opus-Cadence framework

The set of generators has been developed with the facilities (layout editor and parametrized cells) proposed in the Opus-Cadence framework. The generators may be defined graphically or writing the corresponding code using the Cadence development language called Skill [9]. When defined graphically, the code file may be obtained. In this case, it was preferred to directly write in the development language because of the high level of parameterization necessary: none of the parameter values is hardcoded. The main transformations available for parameterized cells (pcells) are: stretch of polygons, repetition, inclusion of parameterized labels, inclusion of parameterized levels of masks, conditional inclusion of shapes, repetition along shapes, inheritance of parameters and parameterization of properties. The pcells may be instanciated as any other cell but the parameters may be specified each time. In the Opus framework, the instanciated version of a pcell remains a dynamic link.

IV. Development strategy

A. Technology Definition

When designing a layout synthesis tool, a major preoccupation is the technology independence. That is, ideally, a technology change must not require to modify generator programs. Only the new characteristics of the technology have to be entered. In practice, for major technology evolutions, a partial rewriting may be necessary. In this context, the way used to define a technology is essential. A proper technology description without link with the Opus format was preferred from the following considerations:

- with proper rule definition, the generator program is immune with respect to some technology design-kit alterations and the requirements of other tools;
- moreover, a rewriting in a more common language is simpler if there is no strong link with the host system.

The technology data are stored in a structure composed of basic parameter values (minimum pitch and technology name) and lists. The lists group the layers (group `Layer`) or geometrical rules (groups `Width`, `Space`, `Overlap` and `Intersection`) or electrical characteristics (groups `Resistor` and `Capacitor`). A group `Other` is used to described all properties not covered by the previous groups.

A rule is a set of keywords and an associated value placed in a list. The elements are in order: the group name, then usually symbolic names of layers and finally the value. That is, for each mask layer level is defined one symbolic name or more may be defined. For example, the rules:

```
´(Layer       poly1  "poly" )
´(Layer       gate   "poly" )
```

define two symbolic names (`poly1` and `gate`) corresponding to the layer "poly". After the symbolic layers definition, the other rules are described. For instance, the overlap of polysilicon over a contact may be defined as follows:

```
´(Overlap     poly1  contact      0.30 )
```

Generally, only one or two symbolic layer names are necessary to characterize a rule. Sometimes, however, a greater number is required. The use of keywords not related to layers is also possible. For example, the rule:

```
´(Other       contactOnCapacitor  t )
```

specifies that the implementation of contacts on the top plate of a capacitor is allowed.

B. Files

Several kinds of code are distinguished: the core of the program and functions for general use, the generator code, the technological and generator parameter specifications and some global environment specifications. Normally, a custom designer only needs to define its own global environment specifications: directory paths, program version, and so on. The technology and generator specifications are only modified when a technology rule change occurs. Most often, it is not necessary to alter the generator specifications even when entering a new technology if the same symbolic layer names are used.

C. Generator description

A generator is a program which builds shapes from a set of user defined variables and a set of technology dependent predefined parameters. For example, for a MOS transistor generator, the channel width and length are user variables and the minimum width and the minimum length are technology dependent parameters. Each generator is described in a separated file. They are all described with the following organization:

- Declaration of the variables: Name and default value.
- Definition of local variables from the values of necessary technological rules (layers, geometrical and electrical rules).
- Computation of some local variables necessary to generate the shapes from the variables and the previous local variables.
- Generation of the shapes.

D. Generator parameters

The code of the generators is entirely defined using variables or parameters. There is no direct reference to a numerical hard coded value. When a technological parameter is needed, the generator set of rules is first examined. All rules may be redefined for a particular generator. If not found, the rule is searched in the technology rule set. Finally, if not found, a default value is assumed. Some data are only searched in the generator parameter set, for instance, the layers used.

V. Generators Overview

The development of a generator library is a cumbersome task: While a large variety of generators including some elaborated structures is suitable to cover the main needs and give a high gain with respect to fully manual design, complex generators may involve a loss of universality and flexibility. That is why a lot of automatic analog synthesis tools often use a very small set of device generators considering that more complex configurations may be reached through optimization algorithms [4]. Even so, some structures are really difficult to obtain automatically. That is why a set of generators should contain the basic structures and some others more elaborated. These are particularly important to optimize some electrical performances or to reduce area.

Today, the set of generators includes:

- folded transistors;
- capacitor arrays;

- differential pairs;
- current sources;
- waffle shaped transistors;
- inductors;
- resistors;
- contacts.

VI. Current Source generator

A current source is one of the most used structures in analog circuits. The use of a generator is particularly interesting because an optimized shape with large channel widths requires to fold the transistors and to merge drains and sources. In order to obtain a similar shape from basic transistor generators with a general synthesis tool, a lot of computational effort is required. The main properties of the implemented MOS current source generator are the following:

- unlimited number of transistors;
- possibility to specify a different channel width for each transistor;
- possibility to fold the transistors (the folding factor is unique for all transistors of a current source);
- possibility to implement or not source, drain or gate connections;
- the lateral contacts may be omitted in order to allow an easier merge with other structures.

Some examples are given below.

A. Example of a current mirror

The transistors have the same channel width and length. The following configuration is assumed:

```
transistorNumber     2
foldFactor           4            ;(an even number)
polycontact          2            ;(transistor with a drain-gate connection)
sourceConnected      true
drainConnected true
gateConnected        true
```

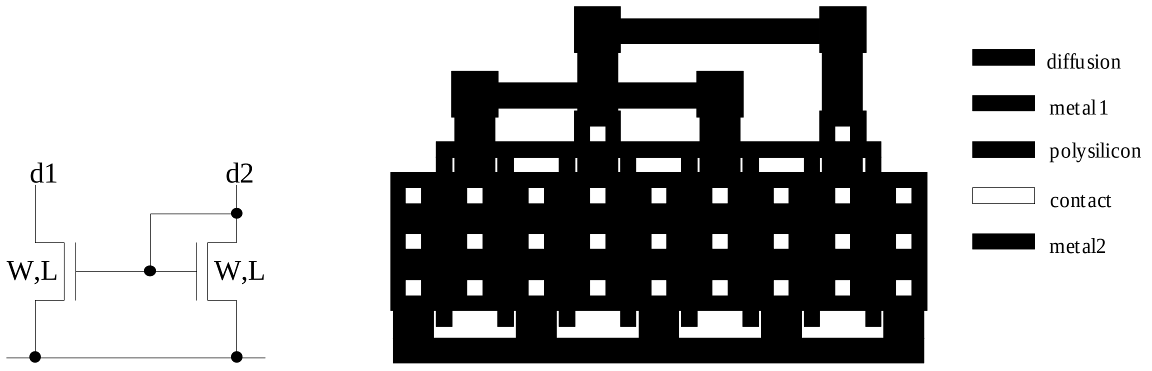

Figure 1. Current Mirror layout

B. Example of a 3-transistors current source

In this example, a current source formed by three transistors of different channel widths is considered. The first one is implemented with a connection gate-drain. The configuration used is the following:

```
transistorNumber     3
foldFactor           4
polyContact          1
transistorWidth      ´(20  32  24);(transistors widths)
sourceConnected      true
drainConnected true
gateConnected        true
```

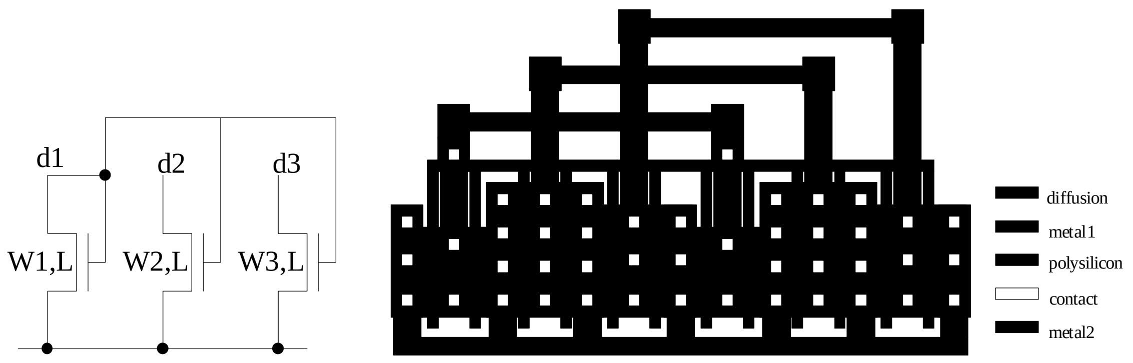

Figure 2. 3 transistors current source

VII. Waffle Shaped transistor

The waffle shaped transistor configuration is well suited to implement transistors with a very large channel width over length ratio while keeping low source, drain and gate resistors. The reduced areas of drain and source allow low junction capacitors on these terminals. The side capacitance effect is minimized. A typical waffle-shaped transistor is represented on the figure below: the gate is subdivided in segments distributed horizontally and vertically forming a grid. Source and drain areas are connected by diagonal metal lines.

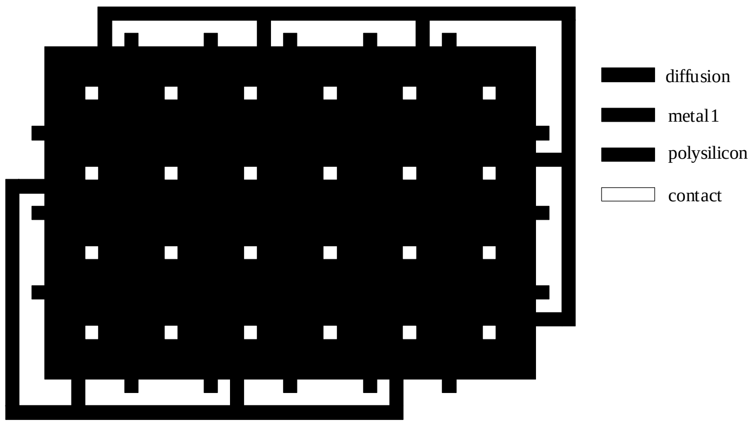

Figure 3. Waffle shaped transistor

A drawback of this structure is that drain and source connections overcross gate introducing parasitic capacitances. It may be noticed also that this structure is not well characterized: because of the crossing of gate segments, the total channel width of the transistor is difficult to evaluate. In this section, are described how to design the component and the main properties of the developed generator.

The following notations are used:

- `W` total channel width;
- `L` channel length;
- `hg` number of horizontal gate segments;
- `vg` number of vertical gate segments;
- `dg` distance between two parallel gates;
- `hw` horizontal external width of source or drain;
- `vw` vertical external width of source or drain;
- `gc` factor to take into account the crossing of gate segments ($0 \leq gc \leq 1$);
- `dd` coefficient to compute distances with diagonal connections (`dd >` $\sqrt{\ }$).

A. Distance between gate segments

It is interesting to minimize the distance between gate segments in order to maximize the gate density and to minimize the source and drain areas. Hence, the parameter `dg` is computed from the geometric rules of the technology. In most cases, the diagonal constraint on metal1 determines its value.

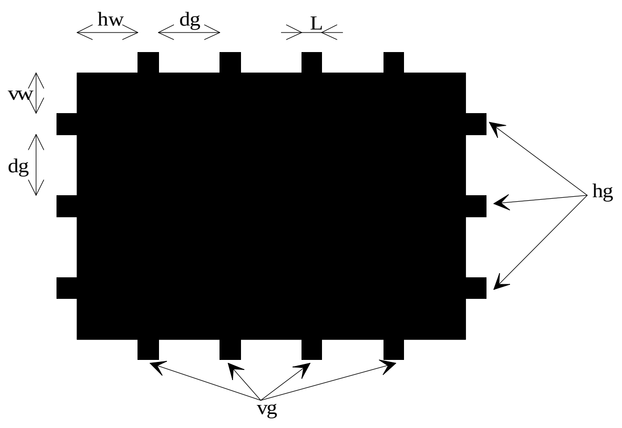

Figure 4. Notations

B. Total width of the transistor

It is assumed in this section that the number of horizontal and vertical gate segments are parameters. The expression of the total width as a function of these parameters is deduced.

- For the core of the transistor (without the external channels):

$$W_{core} = (vg - 1) \times hg \times (dg + gc \times L) + (hg - 1) \times vg \times (dg + gc \times L)$$

- For the external channels:

$$W_{extern} = vg \times (2 \times vw + gc \times L) + hg \times (2 \times hw + gc \times L)$$
$$W_{total} = W_{core} + W_{extern}$$

C. Determination of the horizontal and vertical numbers of gate segments

The external channel widths hw and vw admit a lower bound we_{min}. That is determined by the area necessary to implement a contact on an external source or drain. In order to determine the horizontal and vertical numbers of gate segments, the previous equation of W_{total} substituting hw and vw by we_{min} is used. A set of couples (hg, vg) is determined such that:

- $W(hg, vg, we_{min}, we_{min}) \leq W_{desired}$
- $W(hg + 1, vg, we_{min}, we_{min}) > W_{desired}$
- $W(hg, vg + 1, we_{min}, we_{min}) > W_{desired}$

From this set, a specified form factor may be used to choose gate segment numbers. Then, hw and vw are adjusted to obtain $W_{desired} = W_{total}$.

D. Waffle shaped transistor generator

Currently, the implemented waffle shaped transistor generator is defined from the following parameters:

• `length`	Length of the channel;
• `horFoldFactor`	(hg) number of horizontal gate segments;
• `verFoldFactor`	(vg) number of vertical gate segments;
• `lateralHorWidth`	(hw) horizontal external width of source or drain;
• `lateralVerWidth`	(vw) vertical external width of source or drain.

From these parameters, the total width of the transistor channel is computed.

E. Parasitic capacitors

The total area and perimeter of source and drain are calculated from the parameters described in the previous section. Two cases are distinguished:

- The number of horizontal gate segments and the number of vertical ones are odd. In this case, the total area of the source is equal to the total area of the drain: $A_{source} = A_{drain}$.

- In all other cases, the drain area is related with the source area by the formula:

$$A_{source} = A_{drain} - 4 \times hw \times vw + 2 \times (hw + vw) \times dg$$

Also, the total area of source and drain is given by the formula:

$$\begin{aligned} A_{source} + A_{drain} = \; & (hg - 1) \times (vg - 1) \times dg^2 \\ & + 4 \times hw \times vw \\ & + 2 \times dg \times \{ (hg - 1) \times hw + (vg - 1) \times vw) \} \end{aligned}$$

The parasitic capacitors of the source and drain (or the parasitic diodes) may be deduced from these formulas and the electrical parameters of the technology.

The waffle shaped transistor is particularly efficient to reduce the fringe capacitor effect on the source and the drain. The length of the source or drain perimeter depends on the parity of the number of gate segments horizontally and vertically.

- If either the horizontal gate segment number is odd or the vertical gate segment is odd, the perimeters of the source and of the drain are equal: $P_{source} = P_{drain}$.
- otherwise:

$$P_{drain} = P_{source} - 4 \times (hw + vw) + 2 \times dg \times \{(hg - 1) + (vg - 1)\}$$

VIII. Square capacitor array

Switched capacitor circuits are composed of operational amplifiers, switches and arrays of capacitors. In order to obtain the best precision on the capacitor ratios, arrays of identical square capacitors are used. One difficulty when designing a generator of capacitor array is that the exact structure strongly depends on the technology rules. It is important to exploit all of the technological possibilities to optimize the performances. However, the possibilities may vary from a technology to another. For example, the use of contacts on the top plate of the capacitor is not authorized in all technologies. The defined generator includes several different structures in order to cover a large set of configurations. The generator is tolerant to technology migration.

In the designed generator, the parameters are:

- the value of a unit capacitor (an element of the capacitor array);
- the total value of the capacitor;
- the number of columns of unit capacitors in the array.

Adjusting the value and then the dimensions of the unit capacitor, a compromise may be established between area and precision. The implemented total capacitor is a multiple of the unit capacitor. The number of columns in the array defines the form factor of the array.

The dimensions of the unit capacitor are obtained by the formula:

$$L = \frac{-2C_p + \sqrt{4C_p + CC_s}}{2C_s}$$

where C_p is the fringe capacitor, C_s the capacitor per area and C the desired value.

Actually, the generator is formed by a set of sub-blocks. The basic structure is shown on the figure 6. Each sub-block of the generator is composed of the repetition, when necessary, of a smaller generator. For example, the sub-block I is composed of the repetition horizontally and vertically of the unit capacitor which is produced by a generator. Hence, five different generators are used to synthesize the capacitor array.

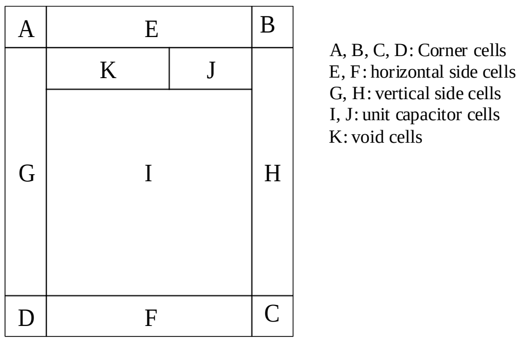
A
E
B
K
J
G
I
H
D
F
C
A, B, C, D: Corner cells
E, F: horizontal side cells
G, H: vertical side cells
I, J: unit capacitor cells
K: void cells

Figure 6. Structure of the capacitor array generator

Some topological options are:

- to connect or not on the top of the capacitors;
- to use contacts on the top plate of the capacitors;

- to implement a unique bottom plate for the array or a different square bottom plate for each unit capacitor.

IX. Conclusion

A set of layout generators for basic devices has been presented. Written in the Skill language the Opus framework, they are easily adaptable for compatible CMOS technologies by redefining a parameter file. They have been used to design the layout of various analog integrated circuits in a double-polysilicon, double-metal CMOS technology (Operational Transconductance Amplifier, Temperature Transducer, Current Multiplier and Switched-Current Sigma Delta Modulator) [10-14]. The use of the generators reduces the time required to layout a circuit and allows to experiment rapidly various configurations in order to obtain the best performances. Taking advantage of the formalism defined, new generators may be created easily. Currently, the generators are being rewritten in C language. Future works are the extension of the use of the generators to the synthesis of predefined analog circuits (sigma-delta switched-current structures) and the development of a tool to synthesize cell masks from a sized netlist description. This tool will take into account analog characteristics (sensibilities, voltages, currents) to optimize the layout.

X. References

[1] L. R. Carley, G. G. E. Gielen, R. A. Rutenbar, W. M. C. Sansen, “Synthesis Tools for Mixed-Signal ICs: Progress on Frontend and Backend Strategies”, in proceedings of *the 33rd Design Automation Conference*, 1996.

[2] J. Rijmenants, *et al*., “ILAC: An automated layout tool for analog CMOS circuits”, *IEEE JSSC*, Vol. 24, n^{o} 4, pp. 417-425, April 1989.

[3] J. Kuhn, “Analog Module Generators for Silicon Compilation”, *VLSI System Design*, May 1987.

[4] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, L. R. Carley, “Analog Device-Level Layout Automation”, Kluwer Academic Publishers, 1994.

[5] K. Lampaert, G, Gielen, W. M. sansen, “A performance-driven placement tool for analog integrated circuits”, *IEEE JSSC*, Vol. 30, N^{o} 7, pp. 773-780, July 1995.

[6] P. Miliozzi, I. Vassiliou, E. Charbon, E. Malavasi. A. L. Sangiovanni-Vincentelli, “Use of Sensitivities and Generalized Substrate Models in Mixed-Signal IC Design”, in proceedings of *the 33rd Design Automation Conference*, 1996.

[7] J. D. Bruce, H. W. Li, M. J. Dallabetta, R. J. Baker, “Analog Layout Using ALAS!”, *IEEE JSSC*, Vol. 31, N^{o} 2, pp. 271-274, february 1996.

[8] B. R. Owen, “BALLISTIC: An Analog Layout Language - Beta+ Version Reference Document”, University of Toronto, september 1995.

[9] Cadence - Opus Framework On-Line Documentation.

[10] J. Porte, F. Yang, K. Azadet, J.-F. Naviner, “Continuous-Time Current-,Mode Filters Using Normal Biquad Structure”, in proceedings of the *37th Midwest Symposium on Circuits and Systems*, pp. 994-997, august 1994.

[11] O. Oliaei, P. Loumeau, “A low-input resistance class AB CMOS current conveyor”, in proceedings of the *39th Midwest Symposium on Circuits and Systems*, august 1996.

[12] R.C.S. Freire, G.S. Deep, J.F. Naviner, P. Loumeau, “A p-n junction-based precision temperature transducerintegrated circuit”, in proceedings of the *Int. Workshop on Thermal Investigations of ICs and Microstructures*, september 1996.

[13] O. Oliaei, P. Loumeau, “A CMOS Class AB Current-Multiplier”, to be published at ISCAS, july 1997.

[14] O. Oliaei, P. Loumeau, H. Aboushady, “A Switched-Current Class AB Sigma-Delta Modulator”, to be published at ISCAS, july 1997.