A Layout Approach for Electrical and Physical Design Integration of High-Performance Analog Circuits

Mohamed Dessouky and Marie-Minerve Louërat

Université Pierre et Marie Curie. Laboratoire LIP6-ASIM.
4, place Jussieu. 75252 Paris Cedex 05. France.
E-mail: *Mohamed.Dessouky@lip6.fr*

Abstract

This paper presents a layout generation tool that aims to reduce the gap between electrical sizing and physical realization of high performance analog circuits. The procedural layout approach is shown to be best suited for this kind of methodologies. Once captured, the procedural description can be used several times to calculate both rapidly and accurately all parasitics that appear during physical realizations without layout generation. Efficient algorithms are developed to take into account analog layout constraints such as matching, parasitic control, shape and reliability considerations. This allows to account for these effects early in the design which guarantees the fulfillment of the required performance specifications, permits to optimize various design aspects in the presence of parasitics and shortens the overall design time by avoiding laborious sizing-layout iterations. An example of a high performance OTA is presented at the end to illustrate the effectiveness of the approach.

1. Introduction

During the design of high performance analog circuits, device matching, parasitics, reliability design rules, thermal and substrate effects must all be taken into account. All of these effects can be controlled with a good layout design performed either manually by an expert layout designer or using a dedicated automatic tool. However, the nominal values of performance specifications are subject to degradation due to a large number of parasitics which are generally difficult to estimate accurately before the actual layout is complete. Over-estimation of layout parasitics results in wasted power and area, while under-estimation of parasitics leads to circuits that do not meet the required specifications.

Various layout automation tools have been reported in order to automate the layout generation phase [1–6]. They can be classified into two main groups: knowledge based approaches and optimization based ones.

In the first group the circuit topology is always fixed. A *sound* topological arrangement for the building blocks of the circuit is stored based on traditionally accumulated design experience. Knowledge storage can either be in the form of a procedural layout [1] or through the use of topology libraries [2], by employing a *design by example* principle (layout templates) [3].
The second group of approaches employs an optimization algorithm to generate a suitable placement configuration followed by a routing phase [4]. They are fully automated and strive to take a large number of specific analog constraints into account. More recently, a performance-driven layout methodology has been introduced [5]. Performance specifications are mapped onto a set of constraints for critical parasitics which are then used to drive the layout tools. In [6], performance constraints are used to drive directly the layout tools.

As the knowledge based approaches offer a short generation time, and a reuse of the expert knowledge (which seems to be indispensable to the analog domain), they suffer from their high design cost and thus are best suited for frequently used circuits. On the other hand, the optimization based approaches offer an *automatic* layout generation which tries to optimize the layout, but they suffer from the complexity of the optimization problem and the determination of the appropriate cost function, this is besides a long generation time. They are thus best suited for circuits with small number of devices.

All of the systems cited above consider the layout as a step which *follows* the design synthesis process. The layout generation tool does not interact during the design synthesis. So the circuit sizing tool has *no* information about the parasitics that the circuit is going to generate during the layout phase.

In this paper a procedural layout tool that incorporates a fast area optimization algorithm and accurate parasitic calculation is presented as a solution that couples both circuit sizing and layout generation.

Different module layout styles can be used, their corresponding parasitic contribution is evaluated and sent back

to the circuit sizing program. Layout techniques that minimize parasitic capacitances on certain nets and enhance the overall performance can be further exploited in order to optimize certain design aspects. For example, in an opamp, folding large transistors allows to decrease their source and drain diffusion capacitances. This can be used to optimize transistor sizes and to reduce the current consumption for a given frequency and noise specifications.

This paper is organized as follows, section 2 presents the overall design methodology. In Section 3 various analog layout constraints are discussed and solutions that are incorporated in the tool are proposed. Section 4 describes the implementation of the layout tool. An example of parasitic calculation during circuit sizing is given in section 5. Finally, conclusions are summarized in section 6.

2. Layout-oriented design methodology

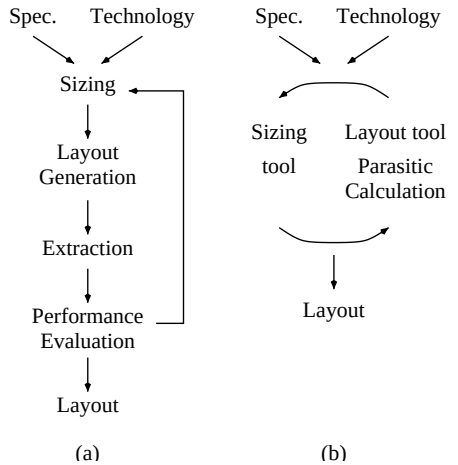

Figure 1. Design Flow: (a) traditional and (b) proposed

The problem of compensating layout parasitics is usually solved as demonstrated by the design flow shown in Fig. 1(a). The design process follows laborious iteration loops during which circuit sizing is followed by generating the layout, extracting the circuit netlist with layout parasitics and evaluating the effect of those parasitics in order to compensate for them by re-sizing the circuit. This re-sizing modifies the parasitics and the loop is repeated till a satisfying performance is obtained.

Fig. 1(b) shows the proposed layout-oriented methodology. The approach is an extension to that first presented in [7] by considering a more detailed parasitic extraction and analog layout constraints while treating circuit reliability conditions in the same time. Layout information is passed to the circuit sizing tool early in the design phase. Multiple calls to the layout tool in a *parasitic calculation* mode are allowed as the design progresses. This approach guarantees a circuit that satisfies the performance specifications even in the presence of circuit parasitics. The accuracy is largely dependent on the precision of parasitic calculations by the layout tool, as well as its capability to take analog layout constraints into consideration. Physical layout constraints such as the global aspect ratio and circuit reliability design rules can be taken into account *during* circuit sizing.

In order to be used in the proposed design methodology, the layout generation tool must satisfy the following conditions:

- It must be fast as it is normally called several times during circuit sizing.
- It must support an accurate method for parasitic calculation.
- It must support conventional analog layout constraints.
- In order to explore various design space points, it must support different layout options for each device.

It is clear from the first condition that optimization-based layout generation approaches [4–6] can't be used due to their high computational cost. On the other hand, the knowledge-based approach seems to be attractive for its short layout generation time. The procedural approach has been thus chosen for reasons of flexibility and generality. The use of procedural module generators and routing allows to predict parasitics before the layout is done. This is achieved through a dedicated layout language that allows to easily describe relatively both module placement and routing.

3. Analog layout constraints

In addition of being fast, analog layout effects must be carefully treated in the layout tool. Layout constraints taken into account are presented hereafter together with the algorithms developed to control them.

3.1. Parasitics constraints

All transistors are built using a single motif generator that produces one of the following motifs: A single-module transistor *M1*, a double-module transistor *M2*, a single-module transistor with a dummy one *M1D* and a dummy transistor *MD* all shown in Fig. 2. A bulk contact accompanies each motif to maintain the bulk potential. Motifs are stacked or interleaved in order to build-up transistors. This motif generator provides total control over transistor terminals and wires. This gives an additional degree of freedom

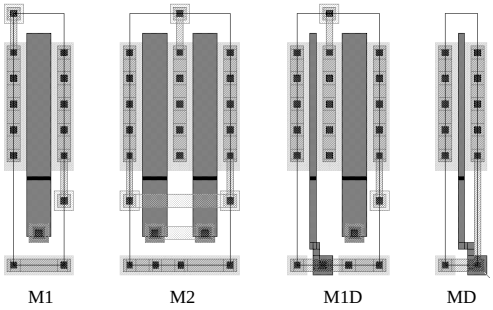

Figure 2. Motifs used in building transistors

to control coupling parasitic capacitance between wires inside transistors according to the application [8]. Fig. 3 shows two different implementations of the same transistor. In 3(a) the gate and source are superimposed, while the drain passes over both of them. This module configuration can be used in a biasing network or a low frequency application. In 3(b) the gates are joined by the first metal layer close to the active region to reduce RC effects, the source passes over it using the second metal layer and the drain is separated downwards. This module is best suited to high frequency applications. Both versions could thus be used in two different contexts depending on the application and module routing. Both of them are generated using the *same* module generator.

Very wide transistors are also generated on multiple stacks. This allows to insert more bulk contacts in order to avoid latch-up and to reduce substrate coupling noise.

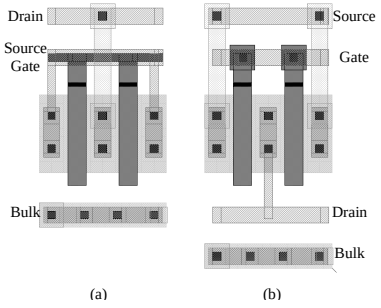

Figure 3. Different transistor overlapping terminals

Transistor folding reduces the diffusion-bulk parasitic capacitance (drain-bulk and source-bulk capacitances). This is due to the sharing of these diffusion areas between folds. The total effective diffusion width W_{eff} is usually a fraction F of the transistor width W ($W_{eff} = F.W$), where F is the capacitance *reduction factor*. In case of a non-folded transistor $F = 1$. While for a folded one, F depends on the number of folds N_f and the position of the

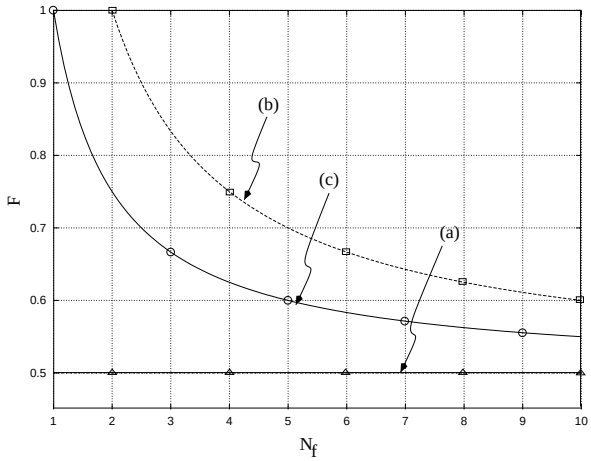

Figure 4. Capacitance reduction factor F

diffusion (for alternate source/drain diffusions) as follows:

$$F = \begin{cases} \frac{1}{2} & \text{if } N_f \text{ even \& internal diffusion (a),} \\ \frac{N_f+2}{2N_f} & \text{if } N_f \text{ even \& external diffusion (b),} \\ \frac{N_f+1}{2N_f} & \text{if } N_f \text{ odd (c).} \end{cases} \quad (1)$$

As shown in Fig. 4, this reduction factor F decreases significantly for the first few folds for cases 1(b) and 1(c). It is clear that this parasitic capacitance can be minimized on a given net by controlling the number of folds of the transistor connected to this net to be even, and by connecting it to the internal diffusion (case 1(a)). This parasitic control is used by the tool to enhance the frequency characteristics of the layout.

3.2. Matching constraints

Special layout styles of transistors are used in order to minimize device mismatch based on considerations of process gradients, temperature gradients, anisotropic effects, and boundary effects. Interleaving and common centroid configurations are shown to be effective in reducing the mismatch due to linearly varying parameters across the chip surface [9]. Combined with parasitic constraints, several configurations of critical transistors in the circuit could be investigated and a good compromise between matching and parasitic effects could be found.

The mismatch between transistors is also dependent on their relative channel orientation. Consider two MOS transistors Mi and Mj, respectively split into n_i and n_j modules, all in the same stack and carrying the same nominal current I. The *current mismatch* F_{ij} between transistors

Mi and Mj is given by [10]

$$F_{ij} \triangleq \frac{\epsilon_I}{I} \left| \frac{\Delta n_i}{n_i} - \frac{\Delta n_j}{n_j} \right| \quad (2)$$

where ϵ_I is the maximum error of the difference between currents flowing through channels with opposite orientations, $\Delta n_i(\Delta n_j)$ is the difference between the number of motifs oriented in opposite orientations of transistor $Mi(Mj)$. For N transistors in the same stack, the *current mismatch* F_N is defined as

$$F_N = \sum_{i=1}^{N} \sum_{j=i+1}^{N} F_{ij} \quad (3)$$

An algorithm dedicated to the layout of tightly matched transistors has been developed. It takes into account channel orientation and guarantees maximum interleaving between transistors all centered (as much as possible) around the stack mid-point (common-centroid).

The algorithm is based on the *M2* and the *M1D* motifs shown in Fig. 2. For these two motifs, if the source is assigned to the external diffusion and the drain to the internal one, then they can be freely interleaved by sharing the source diffusion area.

Current mirrors are a special case where tight matching between transistors is usually essential. Given the current ratio of a mirror with N transistors, the first step is to assign for each transistor the appropriate motifs that minimize F_N given by equation 3. Each transistor Mi is thus composed of nm_{1i} motifs of type *M1D* and nm_{2i} motifs of type *M2*. The total number of motifs in each transistor is $nm_{ti} = nm_{1i} + nm_{2i}$ and the total number of transistor modules is $n_i = nm_{1i} + 2.nm_{2i}$. Since the *M2* motif has two transistor modules with opposite channel orientations while *M1D* has only one module oriented to the right, then by definition $\Delta n_i = nm_{1i}$. This motif assignment is done by an exhaustive trial of all possible motif combinations. A trivial solution that leads to $F_N = 0$ is to take all motifs of the type *M1D*, i.e. all transistors has the same channel orientation with dummy transistors inserted in between. This solution however increases the distance between transistors which is another important matching factor. It also requires overall excessive area. Thus solutions of more than one transistor with *all* channel orientations in one direction are rejected. Another possibility is to force n_i to be even. In this case $\Delta n_i = nm_{1i} = 0$ which leads to $F_N = 0$. However, this results in stacks of non-practical aspect ratios and to transistor modules with small widths. As a consequence, mismatch problems due to small channel area arise [10].

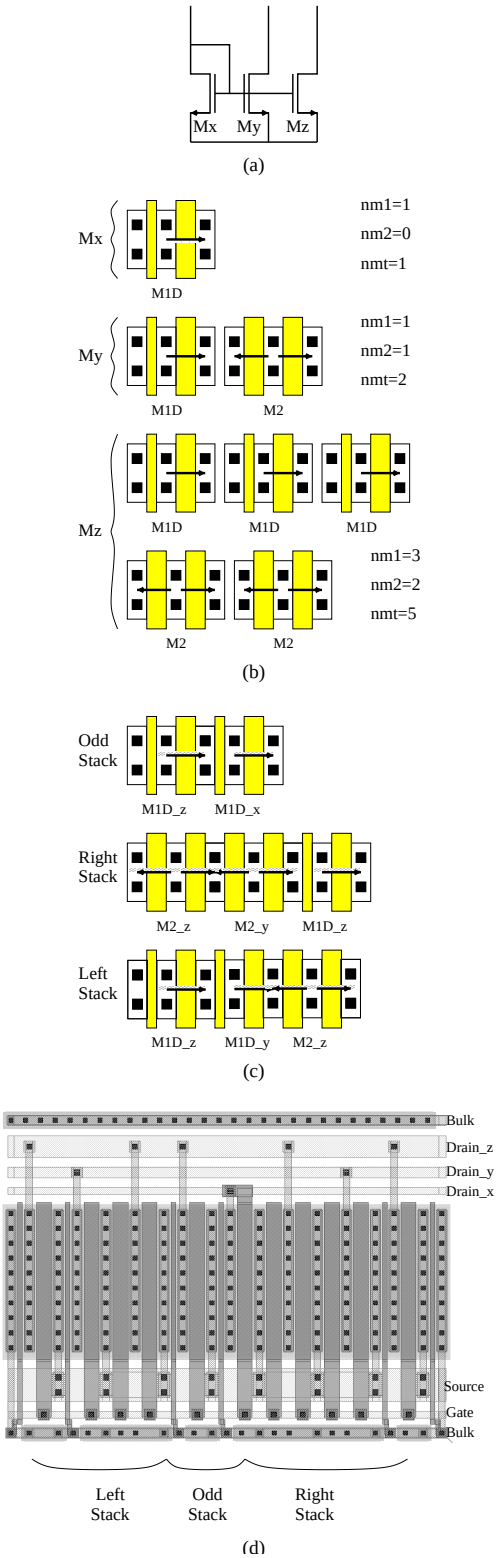

Figure 5. Current mirror, (a) schematic, (b) transistor motifs, (c) elementary stacks, and (d) final layout

The second step is to interleave the motifs of all transistors while centering each group of motifs belonging to a given transistor around the middle of the stack. In order to achieve this, three elementary stacks are constructed: An *odd* stack containing one motif from each transistor with an odd number of motifs nm_t, and two other stacks (a *left* and a *right* stack) constructed by placing one motif alternatively from each transistor till all motifs are exhausted. This ensures maximum interleaving between transistors. The required current mirror stack is then composed of the *odd* stack placed at the middle, and the other two stacks abutted one at each side. This places the centroid of all transistors near the middle of the final stack.

As an example, consider a current mirror composed of three transistors $Mx : My : Mz = 1 : 3 : 7$ shown in Fig. 5(a). Arrows show the direction of current flow. Applying the previous algorithm, the following motifs are found: $nm_{1x}/n_x : nm_{1y}/n_y : nm_{1z}/n_z = \Delta n_x/n_x : \Delta n_y/n_y : \Delta n_z/n_z = 1/1 : 1/3 : 3/7$ which minimizes equation (3). The assigned motifs of each transistor are shown in Fig. 5(b). Since the number of motifs of Mx and Mz (nm_{tx} and nm_{tz}) are both odd, one motif from each is placed in the *odd stack*. The other two stacks are then composed by taking one motif alternatively from each transistor as shown in Fig. 5(c). Fig. 5(d) shows the physical layout of the current mirror stack after abutting the three elementary stacks shown in 5(c). The gates of all dummy transistors are connected to the bulk terminal to ensure their OFF state.

3.3. Reliability constraints

Reliability design rules are important for the long-term functionality of the circuit. DC current information is used to adjust wire widths inside each module as well as routing wires in order to respect the maximum current density allowed by the technology. This prevents electromigration from taking place which may lead to open circuits in wires subjected to high current densities [8]. The number of contacts are also increased for wide wires in order to decrease their resistance according to the reliability design rules. This is clearly shown in the current mirror shown in Fig. 5(d) where wire widths and contact numbers have been adjusted separately for each transistor assuming high current densities. The widest wire is that of the source where the sum of all transistor currents passes.

3.4. Shape constraint

The layout is usually driven by a shape constraint (a given height or aspect ratio). Given this constraint area optimization is performed using an efficient algorithm based on *shape functions* and *slicing structures* [3].

Shape functions calculate the dimensions of alternative shapes for each module, while slicing structures are used to store relative placement of layout modules. The procedural tool allows to place modules relatively in horizontal slices called *groups*, which in turn are placed in vertical *slices*. Shape function propagation allows to calculate the shape functions of *groups* and *slices* starting form those of the layout modules.

```
OPTIMIZE_SLICE(HS)
Phase 1:
   FIND the initial set of group heights h_i;
Phase 2:
   DO {
      FIND the widest group j (w_j = WS);
      FIND ΔH such that
         when h_j = h_j + ΔH
              w_j = f_j(h_j) < WS;
      /* Try to compensate ΔH by the other groups */
      FOR each group i ≠ j
         WHILE (ΔH > 0)
         DO {
         h_i = h_i − Δh_i such that w_i = f_i(h_i) < WS;
         ΔH = ΔH − Δh_i;}
      IF (ΔH <= 0)
      /* ΔH is compensated by the other groups */
      THEN
         Conserve the new set of heights;
      ELSE
         Exit;
   };
```

Figure 6. Slice width optimization algorithm

Given a slice height HS, let WS be the corresponding slice width, $\underline{h}$ be the set of group heights and $\underline{w}$ the set of the corresponding widths, then the problem of slice area optimization can be formulated as follows:

$$\text{given} \qquad w_i = f_i(h_i) \tag{4}$$

$$\underset{\underline{h}}{\text{minimize}} \qquad WS = max(\underline{w}) \tag{5}$$

$$\text{subjected to:} \qquad \sum_i h_i \leq HS \tag{6}$$

$$h_{imin} \leq h_i \leq h_{imax} \tag{7}$$

where f_i is the group i shape function.

The developed algorithm is summarized in Fig. 6. It is completely hierarchical such that compound slices might contain another previously defined sub-circuits which in turns include several slices.

Analog layout constraints described in the previous sections are directly reflected in the shape function of the corresponding module and affect indirectly the area optimization algorithm.

4. Implementation

The procedural tool is implemented in a form of a special language composed of a documented superset of C functions. This offers independency of any CAD vendor and takes advantage of the C language sophisticated constructs in the same time. An exisiting interactive debugger [11] is being extended to be used with the language to allow graphical interactive debugging.

4.1. Organization

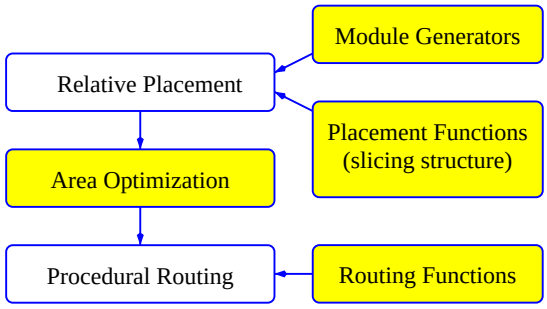

Figure 7. Language Organization

Shaded blocks in Fig. 7 show the main parts constituting the language:

- Complex module generators include transistors (section 3), multi-capacitor arrays and resistors.
- Placement functions that allow to build up the slicing structure.
- Module area optimization algorithm (section 3.4).
- Routing functions that allow relative routing description using predefined reference points. This results in a shape-independent description of the routing.

As an example consider the folded cascode OTA shown in Fig. 8. The three dark areas correspond to three horizontal slices (*groups*) chosen for the corresponding slicing structure. Fig. 9 shows the main sections of the corresponding language code for generating the layout.
In section 1 a netlist *spice* file is opened where device sizes as well as special comments for additional device layout options (number of stacks, transistor current, ...) can be seized. This file is normally generated by the sizing tool.
A device declaration section follows which calls the required *devices* with special layout options. A differential pair *DP1* is called in the first line with dummy transistors at the ends and with the drain capacitance minimization option. Layout styles concerning terminal positions (section

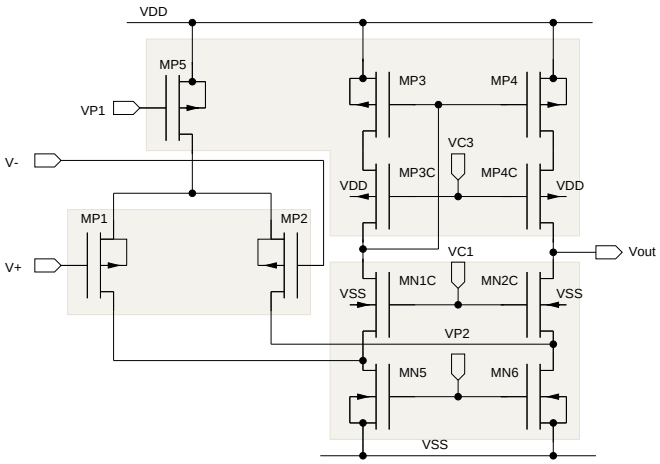

Figure 8. Folded Cascode OTA

3.1) can be changed in an external default style file or directly in the language code as shown in the second line where the second metal level is used for gate connections inside the *DP5* device.
Section 3 constructs the slicing structure with the following functions: the *CAIRO_ADD_DEVICE()* function which builds the *groups*, the *CAIRO_ADD_GROUP()* function which builds the *slices* and the *CAIRO_ADD_SLICE()* function which adds the constructed *slices* to the current module. Area optimization is performed using the *CAIRO_RESHAPE()* function in section 4.
Sections 5 and 6 contain the routing and module terminal definitions. Routing functions support muli-layer routing with appropriate via placement. The width of each wire is determined according to the corresponding layer type and the current of the modules connected to it.

Fig. 10 shows the generated layout. As can be seen from the layout, all transistor folds are chosen such that *drains* are internal diffusions to minimize drain capacitance and enhance the frequency behavior. The input differential pair is interleaved in a common centroid style with dummy transistors placed at the end in order to avoid boundary effects.

4.2. Parasitic extraction

In the *parasitic calculation* mode, after the determination of the shape of each module in the area optimization step, each module calculates the values of parasitic components in a predefined parasitic model which defines the following:

- Transistor layout style. This includes the number of folds for each transistor and their widths, the number of source/drain diffusions which are external, internal to the transistor or shared with other transistors. This allows exact calculation of diffusion capacitances.

```
#include <cairo.h>
/* OTA */
main(argc,argv)
int argc;
char **argv;
{
/***** (1) Open a SPICE file  *****************/
CAIRO_OPEN_SPICE_FILE(argv[1]);

CAIRO_OPEN_MODULE("OTA");

/***** (2) Device Declaration *****************/
CAIRO_DIFFPAIR_SPI("DP1",NTRANS,"MP1","MP2",B_O,
       "DUMMY","DIFF_CAP",MIN_D,C_END);
CAIRO_TRANSISTOR_SPI("DP5",NTRANS,"MP5",B_S,
      "GATE_TYPE_H",ALU2,"DIFF_CAP",MIN_D,C_END);
...

/***** (3) Placement (slicing structure)  *****/
/***** (3.1) Building Groups ******************/
CAIRO_ADD_DEVICE("TP5","group_0","DP5",SYM_X,C_END);
CAIRO_ADD_DEVICE("TP1","group_1","DP1",ROT_P,C_END);
...
/***** (3.2) Building Slices  *****************/
CAIRO_ADD_GROUP("group_0","slice_0","TO",2*PITCH,C_END);
CAIRO_ADD_GROUP("group_1","slice_0",C_END);
...
/***** (3.3) Building Module  *****************/
CAIRO_ADD_SLICE("slice_0",C_END);

/***** (4) Area optimization  *****************/
CAIRO_RESHAPE("OTA",H,argv[2],GENERATION_MODE);

/***** (5) Routing  ***************************/
CAIRO_BEGIN_ROUTE("OTA","OTA");
CAIRO_WIRE3("2",ALU2,ALU2,ALU1,CURRENT_W,CURRENT_W,
        CURRENT_W,"TP1","source",0,"TP5","drain",0,
        CAIRO_GET_Y("MP5",TRA,REF),VER);
...

/***** (6) Defining the module interfaces *****/
CAIRO_PLACE_CON_H("TP5","gate",0,"evp1",ALU2,CURRENT_W);
...
CAIRO_END_ROUTE("OTA");

/***** (7) End module *************************/
CAIRO_CLOSE_MODULE("OTA");
}
```

Figure 9. Language description of the OTA circuit shown in Fig. 8

- Parasitic routing capacitance including coupling capacitance between wires.
- Exact well sizes so that floating well capacitance can be calculated.

Routing parasitics are then calculated and added on circuit nodes. All parasitic calculations are done using simple geometrical methods which combine reasonable accuracy with low computational cost.

4.3. Technology independence

Technology independence is a key feature of any layout tool. A symbolic layout approach [11] is used such that layout generation passes through a technology independent symbolic layout step followed by conversion towards the target technology. A given language description is thus independent of the technology.

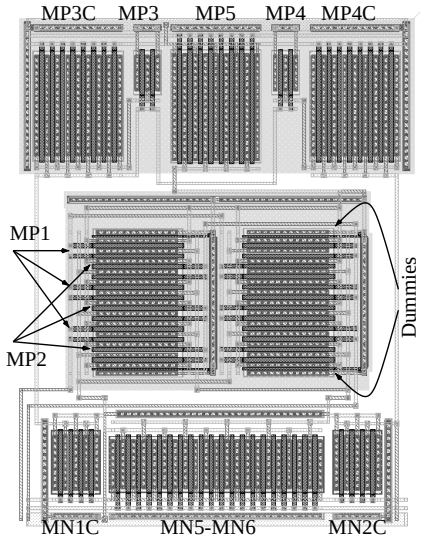

Figure 10. Folded Cascode OTA Layout

5. Example

As an example, the folded cascode OTA shown in Fig. 8 has been synthesized using different layout parasitic considerations. To perform transistor sizing a knowledge-based tool [12] has been used. The OTA is sized for a VDD of 3.3V, a GBW of 65MHz, a phase margin of 65degrees and a load capacitance of 3pF. For comparison, the input common mode voltage range as well as the output voltage range are kept the same for all cases.

Table 1 shows the obtained results, it also shows the results of simulations of the extracted netlist with all parasitics (diffusion, routing and coupling capacitances) between brackets. Final extraction has been done using a commercial design system. In case (1) no layout capacitances (neither diffusion nor routing) have been taken in consideration, only gate capacitances and transistor folding are considered. It can be seen that all dc characteristics matches the extracted layout simulation results, while for the GBW and phase margin we can notice a considerable difference. In case (2) diffusion capacitance has been taken into consideration but assuming only one fold per transistor and neglecting routing capacitance, i.e. no layout information is used during synthesis. Results show that the GBW and phase margin exceed the required specifications. In fact, as the diffusion capacitance is over-estimated, thus the obtained transistor sizes are smaller. This implies that other specifications like the input noise, the dc gain and the output resis-

Table 1. Sizing compared to extracted layout simulation results

Specification	Case (1)	Case (2)	Case (3)	Case (4)
DC gain (dB)	70.1(70.1)	55.0(56.59)	66.1(66.1)	64.7(64.7)
GBW (MHz)	64.9(58.1)	66.5(71.2)	65.0(62.6)	65.8(66.1)
Phase margin (degrees)	65.3(56.3)	65.4(72.4)	65.4(64.4)	65.15(65.4)
Slew rate ($V/\mu s$)	94.0(86.5)	103.0(98.1)	93.3(93.3)	93.0(94.4)
CMRR (dB)	100.7(100.7)	76.9(79.6)	93.9(93.9)	91.6(91.6)
Offset voltage (mV)	0.0(0.0)	0.0(-0.1)	0.0(0.0)	0.0(0.0)
Output Resistance ($Mohm$)	2.4(2.4)	0.38(0.47)	1.5(1.47)	1.23(1.23)
Input noise voltage (μV)	83.9(96.1)	101.6(85.6)	83.3(87.8)	82.7(85.8)
Thermal noise density ($nV/\sqrt{Hz}$)	7.2	6.98	7.15	7.13
Flicker noise density @1Hz ($\mu V/\sqrt{Hz}$)	1.95(3.64)	1.4(8.1)	2.59(4.85)	2.82(5.28)
Power dissipation (mW)	2.0(2.0)	2.4(2.2)	2.1(2.1)	2.1(2.1)

Input specifications: $VDD = 3.3$V, $GBW = 65$MHz, phase margin $= 65$degrees, $C_{load} = 3$pF,
Input CM range $= [-0.55, 1.84]$V, Output range $= [0.51, 2.31]$V.
Case 1: Sizing with no layout capacitances (Neither diffusion nor routing).
Case 2: Sizing with diffusion capacitance assuming single transistor folds and no routing capacitance.
Case 3: Sizing with calculation of exact diffusion capacitance and neglecting routing capacitances.
Case 4: Sizing considering all layout parasitics.
Values between brackets are obtained from layout generation, extraction and simulation.

tance could not be optimized. Note also the resulting offset voltage after folding due to the slight modification of transistor widths needed by layout grid. Case (3) shows sizing results with layout information concerning *exact* diffusion capacitance, no routing capacitance is considered. We notice only a slight difference in the GBW and phase margin between synthesized and extracted netlist simulation. However, both specifications could not be satisfied. Case (4) shows results with all parasitic capacitance information being considered during the synthesis phase. All results match the extracted netlist simulations. The layout corresponding to this case is shown in Fig. 10. Three calls of the layout tool were needed before parasitic convergence. The sizing time for each case including layout calls does not exceed two minutes.

6. Conclusions and future work

A layout tool which aims to closely couple circuit sizing and layout generation has been presented.
Procedural layout is shown to be the best suitable layout method for such methodologies due to its fast layout generation time. The procedural nature of the layout facilitates the use of a *parasitic calculation* mode where parasitics are accurately determined without any layout generation.
The proposed tool thus allows to account for constraints related to the physical implementation of a given cicruit such as parasitics and reliability *during* the design optimization phase and in the same time offers efficient solutions to improve the quality of the produced layout.

Special attention will be given to RF circuits in the future. Layout wire resistance and inductance can be accurately calculated and used to improve design accuracy. Future work also includes synthesis of larger systems as high performance A/D converters using the same methodology.

References

[1] B. R. Owen, R. Duncan, S. Jantzi, C. Ouslis, S. Rezania, and K. Martin. "BALLISTIC: An Analog Layout Language,". In *Proc. IEEE Custom Integrated Circuits Conf.*, 1995.

[2] H. Y. Koh, C. H. Sequin, and P. R. Gray. "OPASYN: A Compiler for CMOS Operational Amplifiers,". *IEEE Trans. Computer-Aided Design*, Feb. 1990, 9(2):113–125.

[3] J. D. Conway and G. G. Schrooten. "An Automatic Layout Generator for Analog Circuits,". In *Proc. European Design Automation Conf.*, 1992, pp. 513–519.

[4] J. M. Cohn, R. A. Rutenbar, and L. R. Carley. "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing,". *IEEE J. of Solid-State Circuits*, Mar. 1991, 26(3):330–342.

[5] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli. "Automation of ICL Layout with Analog Constraints,". *IEEE Trans. Computer-Aided Design*, Aug. 1996, 15(8):923–942.

[6] K. Lampaert, G. Gielen, and W. M. Sansen. "A Performance-Driven Placement Tool for Analog Integrated Circuits,". *IEEE J. of Solid-State Circuits*, July 1995, 30(7):773–780.

[7] H. Onodera, H. Kanbara, and K. Tamaru. "Operational-Amplifier Compilation with Performance Optimization,". *IEEE J. of Solid-State Circuits*, Apr. 1990, 25(2):466–473.

[8] M. Wolf and U. Kleine. "Reliability Driven Module Generation for Analog Layouts,". In *Proc. Int. Symposium on Circuits and Systems*, June 1999, pp. 412–415.

[9] J. Bastos, M. Steyart, B. Graindourze, and W. Sansen. "Matching of MOS Transistors with Different Layout Styles,". In *Proc. IEEE Int. Conf. on Mecroelectronic Test Structures*, Mar. 1996, pp. 17–18.

[10] E. Malavasi and D. Pandini. "Optimum CMOS Stack Generation with Analog Constraints,". *IEEE Trans. Computer-Aided Design*, Jan. 1995, 14(1):107–122.

[11] F. Pétrot. *Outils d'aide au développement de Bibliothèques VLSI portables*. PhD thesis, Université Pierre et Marie Curie, Laboratoire MASI, Paris, July 1994.

[12] J. Porte. *COMDIAC: Compilateur de Dispositifs Actifs*. TELECOM Paris, Sept. 1997.