

A Novel Design Assistant for Analog Circuits

Markus Wolf, Ulrich Kleine

Frédéric Schafer

Institute for Measurement Technologies and Electronics
Otto-von-Guericke University of Magdeburg
PO Box 4120
D-39016 Magdeburg, Germany
Tel: +49 391 67 14625
Fax.: +49 391 5 61 63 58
e-mail: mwolf@ipe.et.uni-magdeburg.de

Institut für Automation und Kommunikation e. V. Magdeburg
Steinfeldstraße
D-39179 Barleben, Germany
Tel: +49 39203 810 22
Fax: +49 39203 81100
e-mail: henning@ifak.fhg.de

Abstract — This paper presents a new design assistant for analog integrated circuits. The interactive tool is implemented in the Design Framework II of Cadence and supports the designer during circuit design. With the help of this new assistant analog designers can create ad hoc layouts of their circuits. These layouts are automatically extracted, and the updated netlist of the circuit is used for further simulation and optimization steps. Thus the optimization is speeded up and the reliability of the design is improved due to the more accurate modeling of the parasitic circuit elements. The use of the design assistant will be demonstrated by various examples.

I. INTRODUCTION

In general, the synthesis process of analog circuits starts with the selection of an appropriate circuit topology or an existing circuit for a similar application. In a second step this circuit is iteratively optimized by circuit simulations to meet the desired specifications. Due to technology scaling, the electrical parameters of MOS transistors get worse for analog circuits and therefore, the design window, in which the dimensions of elements must fit, decreases. Hence, the amount of optimization will increase in the future, especially for low voltage applications. For a reliable optimization, the knowledge of electrical parameters caused by the layout (e. g. parasitic capacitances, RC time constant of gates) is required. Therefore, the layout of the circuit is already necessary during circuit design for high performance circuits.

Today, most analog layouts are hand-crafted by specialists which is a very time consuming process. Consequently, the parasitic elements are normally only roughly estimated during the design and optimization phase without having a layout. On the other hand, several tools have been developed to automate the generation of analog layouts [1]-[4] by bypassing the designer. Usually, these tools apply a top down approach taking the already optimized circuit netlist as input and generating the layout for this circuit.

Applying the new aid analog designers can iteratively generate the layout during the optimization phase to get a more precise estimation of the parasitic elements. This enables the designer to consider the influence of parasitics already during the circuit optimization. Due to analog constraints like matching requirements, it is usual to build more or less complex clusters of devices, hereafter called modules. In recent years, several module generators have been presented [5]-[7]

for MOS and bipolar circuits. In contrast to bipolar circuits, the geometrical parameters of MOS-Transistors (width, length, number of foldings) can vary in a wide range for analog applications. Therefore, module generators must be available which provide optimal layout topologies even for that wide range of parameterization. Depending on parameter values, different optimal topologies can result from geometrical constraints like latch up rules or from electrical properties like matching and amount of parasitic elements. In addition to this, the selection of an optimal topology depends heavily on the special circuit application. This increases the complexity of module construction drastically. For reusable modules the description must be technology independent. The flexibility of the modules can be further improved by additional application dependent parameters which will be described in more detail in Section II. An additional property is a clear description language [8] which allows an easy maintenance and reusability of the modules. With the module generator environment [9] complex analog generators with these features can be created. Thus the module library is kept reasonably small and is maintainable.

In practice, the module creation is often limited to non optimal single device modules [3] and thus the placement and routing problem is increased by necessary analog constraints like neighborhood and symmetry conditions. In contrast to the "Device Level Editor (DLE)" of Cadence in which one element of the schematic is translated into a single layout, in the presented environment elements can be grouped in order to create more complex modules than single device modules. Due to the use of complex modules the remaining placement and routing problem is simplified and does not influence the parasitic calculation drastically because the modules already support inner cell connections.

In the next two sections the examined electrical properties will be presented and the design assistant will be explained in more detail. Section IV will give an example of a layout for a folded cascode amplifier for low voltage applications.

II. ELECTRICAL PROPERTIES IN ANALOG LAYOUTS

In this section electrical properties influenced by the layout are described. It is very important to regard these electrical properties during circuit design because they affect the functionality of the circuit directly. There are two kinds of

properties which are considered: Firstly, the conditions that need to be fulfilled in order to guarantee the functionality of a circuit must be considered. An example of such a property is the electromigration which defines the maximal current density in a metal wire. For the functionality of the circuit, it is important to check that the width of a metal wire is at least as large as the minimal width corresponding to the maximal current density. If the rule is fulfilled the layout will not be affected. In contrast to this, the second kind of properties determines the performance of the circuit directly. For instance, the bandwidth of an amplifier is influenced by parasitic capacitances caused by the layout. In this case the circuit performance can be optimized by a proper design taking into account parasitic capacitances.

A. RC Time Constant of MOS Transistor Gates

The polysilicon gate of a transistor can be regarded as a resistor including a capacitance to ground. Thus, the gate is a RC-network with a time constant which can prevent the signal from reaching the opposite side of a gate simultaneously at high frequencies. In this case, the layout of a module must be changed in order to reduce the RC time constant. In this subsection, the -3 dB corner frequency of the gate is calculated with a distributed RC-network model [14].

In Fig. 1 a longitudinal section of a MOS transistor is depicted. The width and the length of the transistor are indicated in this figure. The gate voltage is applied at point A or B.

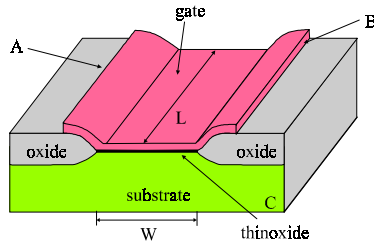


Fig. 1: Longitudinal section of a MOS transistor

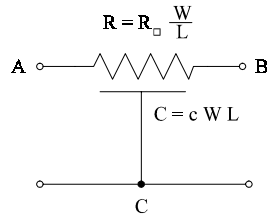


Fig. 2: Distributed RC representation of gate

The adequate distributed RC network of the gate is shown in Fig. 2. R_{\square} is the resistance per square and c defines the gate oxide capacitance per area. With these two technology constants the resistance R and the capacitance C of the gate can be calculated according to the equations in Fig. 2. The equivalent lumped π network [14] for the distributed RC line is shown in Fig. 3 with the admittance Y calculated by (1) and with P given by (2) using the common complex frequency variable p . The periodic transfer function is given by (3).

With the help of the transfer function (3) a maximal frequency for a signal applied to the gate of a MOS transistor can be obtained numerically by calculating the -3 dB corner frequency. If the gate is contacted only at one side, C_0 is the capacitance of the gate extension over the transistor. It is calculated by $C_0 = cL \cdot \text{endcap}$ with the extension of the gate over the transistor *endcap*. If the gate is connected at both sides of the transistor, C_0 is zero and only the half gate width is relevant for the RC network.

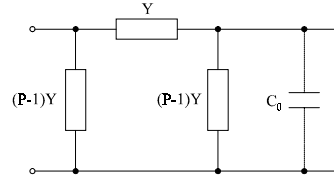


Fig. 3: Lumped π network for the gate

$$Y = \frac{\sqrt{p \frac{c}{R}}}{\sinh \sqrt{sRC}} \quad (1)$$

$$P = \cosh \sqrt{pRC} \quad (2)$$

$$H(s) = \frac{Y}{Y + (P-1) + pC_0} \quad (3)$$

For the verification of equation (3), different simulations have been performed with 1, 2, 10 and 100 connected lumped π networks in order to obtain the -3 dB corner frequencies of the circuits. In Fig. 4 the simulated magnitude frequency responses are displayed for the four different connections of π networks for a single transistor ($W = 200 \mu\text{m}$, $L = 2 \mu\text{m}$). The calculated transfer function is also shown in this figure. The calculated and the four simulated corner frequencies for these simulations are depicted in Tab. 1. As can be seen, the simulation of 100 π networks is a rather good approximation for the real transfer function. Thus the solution of (3) is a good method to calculate the RC time constant.

TAB. 1: CORNER FREQUENCIES

nr. of π networks	3 dB frequency
1	223,3 MHz
2	253,7 MHz
10	270,7 MHz
100	271,5 MHz
calculated	273,3 MHz

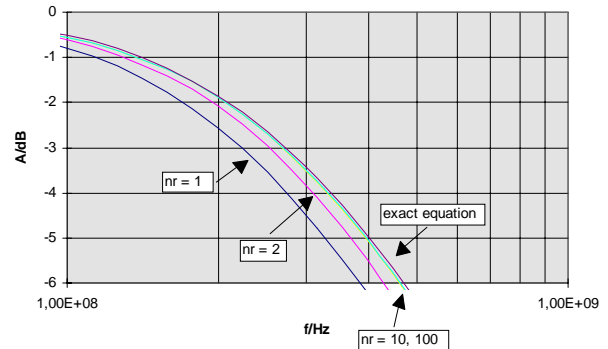


Fig. 4: Computed magnitude frequency responses

Equation (3) has been integrated into the module generator environment in order to check the RC time constant of MOS poly gates. The designer can specify a maximal frequency for a module or single transistors of a module and the environment will check this restriction during module generation. If the RC time constant of a transistor gate is too great to meet the defined frequency, a warning occurs. With this information the module can be changed by modifying the number of gates of a transistor in order to decrease the width of a transistor in an interdigital module, for example. This evaluation can also be performed automatically. The automatic calculation and check of the RC time constant of gates is an example

for a condition which must be fulfilled in analog layouts. Therefore, the design safety of an analog integrated circuit is increased.

B. Parasitic Capacitances

Overlapping rectangles on different electrical potentials cause excess parasitic capacitances [11] in analog layouts. These excess parasitic capacitances in a module may or may not be desired, depending on the function of the module. In a current mirror of a bias network for example, high parasitic capacitances are desired for stabilizing the voltage. In contrast to this, excess parasitic capacitances must be avoided in a current mirror for an active load. In order to keep the number of different modules small the parasitic capacitances can be controlled by a capacitance sensitivity matrix [10] in the module generator environment. Hence, one module description can be used even for different circuit applications. With the help of this matrix, which is defined graphically as an external input for the generator, overlaps between rectangles on certain electrical potentials can be (i) allowed, (ii) allowed only as a simple line cross, (iii) completely forbidden, (iv) completely forbidden with minimal distance, or (v) desired. The benefit of this approach is to enable the analog designer to control the parasitic capacitances easily during module generation. Examples for the use of this matrix will be presented in the following section.

The conditions required for controlling the overlap are realized by automatically generated constraints for the compactor creating the modules [9]. The condition for desired overlaps, which is often applied in manual layouts to fill up empty spaces in order to create excess ground capacitances, is realized by a special filling algorithm which will be presented in this subsection. This algorithm regards electrical connectivity and the design rules automatically because it utilizes the internal compactor routines of the module generator environment and all its features [9].

Fig. 5a illustrates the function for scaling up a rectangle in all directions. The parameters of this function are the rectangle which is to be expanded and the direction which is omitted during expansion. The direction is necessary for the recursive call of this function in order to avoid endless loops due to re-investigation of the rectangles previously expanded. During the first function call, this parameter is omitted in order to expand the original rectangle in all directions. For each direction, the function for scaling up a rectangle calls the function which calculates the allowed expansion (Fig. 5b) of a rectangle in a given direction and inserts an appropriate geometry.

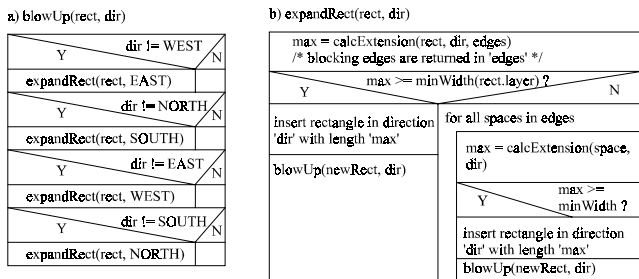


Fig. 5: Structogram for the scale up (a) and expansion (b) function

The function for expanding a rectangle has two parameters, one for the rectangle and one for the direction of expansion. At first, the maximal possible extension of the rectangle is calculated with the function call `calcExtension`. This function uses the compaction algorithm of the module generator environment in order to obtain this value. Rectangles on the same electrical potential are not regarded and the design rules are evaluated automatically. The blocking edges, up to which the rectangle can be extended, are returned in the variable `edges`. If the calculated extension is greater or equal to the minimal width of the expanded layer, a rectangle with the calculated length is inserted and this new rectangle becomes the start rectangle for the further expansion by calling recursively the function `scaleUp`. If the calculated extension is smaller than the minimal width, all free spaces between the blocking edges are investigated along the rectangle to determine if it can be expanded between these edges. A simple example for this expansion algorithm is depicted in Fig. 6.

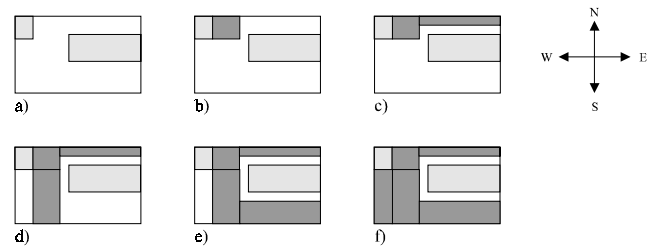


Fig. 6: Example of the filling algorithm

In this example, two rectangles of different potentials exist (Fig. 6a). The left rectangle shall be expanded to fill up the empty area. Firstly, a rectangle is inserted up to the blocking edge of the second rectangle (Fig. 6b). This inserted rectangle then becomes the starting rectangle of the subsequent expansion call. Of course, the new expansion into direction east results in the width of the inserted rectangle smaller than the minimal width. Therefore, the spaces between the blocking edges are investigated and in Fig. 6c the result of this step is shown. Since this new rectangle cannot be extended in any other direction, the first inserted rectangle is extended into direction south (Fig. 6d) after backtracking. Fig. 6e shows the subsequent extension of this rectangle and in Fig. 6f the final filled up layout is depicted.

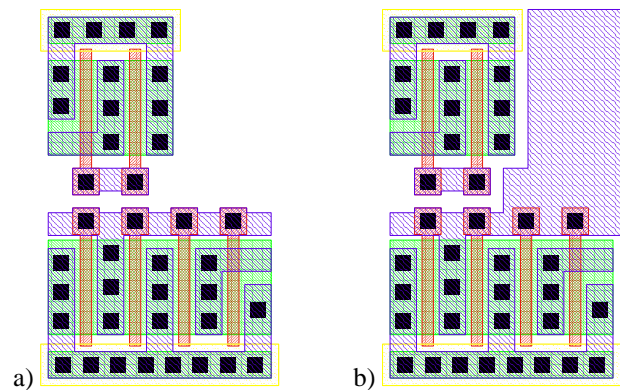


Fig. 7: Example for scale up routine

An example in which this algorithm has been applied to a module is shown in Fig. 7. Both modules consist of a current mirror at the bottom and a folded transistor on the top. In contrast to Fig. 7a the gate rectangles of the module in Fig. 7b have been scaled up.

III. SIMULATION AND OPTIMIZATION ENVIRONMENT

In this section the simulation and optimization environment, which has been integrated in the Cadence design framework II (DFII), will be described in more detail. The graphical user interface has been implemented in SKILL, the interpreter language of Cadence. The user interface executes external programs (module generators) to create automatically the layout of defined modules. The result is displayed in the layout editor of DFII. A snapshot of this environment is depicted in Fig. 8. The schematic window (upper left window) contains the schematic of a circuit which is to be laid out and optimized. In the generator window (upper right window) a schematic view of a generator which is used for the layout generation is displayed. The layout window at the right bottom shows the corresponding module layout.

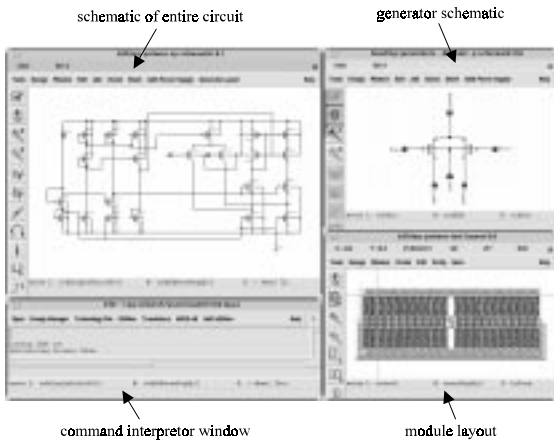


Fig. 8: Snapshot of the simulation and optimization environment

In Fig. 9 the design flow of the environment is illustrated. After entering the circuit into the layout editor of DFII, it is clustered into several modules. For this purpose, the designer selects the appropriate elements of a module in the schematic editor and chooses a generator. In the next step the elements of the circuit are assigned to the elements of the generator view.

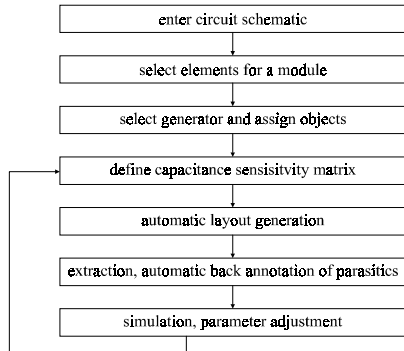


Fig. 9: Design flow in the presented environment

The parasitic capacitances of the modules are controlled with the capacitance sensitivity matrix. After the layout generation, an extraction is performed and the parasitic elements are back annotated into the schematic. With this good parasitic estimation the entire circuit is simulated. Then the parameters of the circuit can be changed in order to optimize the circuit. The optimization loop begins with a possible redefinition of the capacitance sensitivity matrix.

The following subsections describe the main steps of this design flow in more detail.

A. Select Generator and Assign Objects

The generators can be selected from a list of existing generators which have been created with a module generator environment [9]. They support the automatic layout generation even for complex modules like current mirrors or differential pairs. The use of complex modules instead of simple transistors increases the quality of the layout and reduces the subsequent placing and routing problem of these modules. For the correct application of a generator the external connections and the properties (e. g. width and length of transistors) are required. To obtain this information the selected objects of the module in the entire circuit have to be assigned to the elements in the schematic of the generator. This is done graphically by successively clicking on two matching objects in the two schematic views. Hereby the external terminals of the generator are connected with the nets of the circuits. The parameters of the generators are automatically read from the designer's circuit. If these parameters are changed for optimization purposes the generators will automatically take the new values for module generation.



Fig. 10: Defining a module

A snapshot of the dialog box for defining a module is depicted in Fig. 10. For checking the RC time constant of the gates, a maximal frequency of signals applied to the gates can also be specified in this window.

B. Capacitance Sensitivity Matrix

After having assigned each object of the module, the sensitivity of every node can be defined in order to control the para-

sitic capacitances [10]. With this option a matrix can be defined for controlling the overlaps of all nets in the resulting layout. Therefore, it is possible to evaluate different topologies of a module during circuit optimization. The matrix for controlling the parasitic capacitances is defined graphically (s. Fig. 11). The net names of the entire circuit are written in the columns and rows of the matrix. Due to the symmetry of the matrix (the overlap between net1 and net2 is the same as between net2 and net1) only the upper right part of the matrix is displayed. The possible values for the matrix are shown in Fig. 12. The value can be changed by clicking on the appropriate button. The modules are constructed using a special compaction algorithm [9]. The definitions in the capacitance sensitivity matrix are automatically transformed into constraints for the compactor.



Fig. 11: Defining the capacitance sensitivity matrix






-  no overlap allowed, minimal distance required
-  no overlap allowed
-  only cross-overlaps allowed
-  all overlaps allowed
-  overlaps desired

Fig. 12: Possible overlaps for the capacitance sensitivity matrix

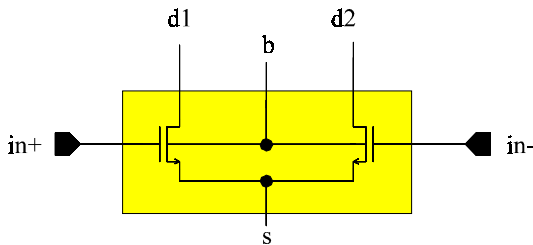


Fig. 13: Schematic of a differential pair

In order to demonstrate the influence of this matrix on the module generation two different results of a module will be presented. In Fig. 13 the schematic of a MOS differential pair is depicted. This is a very common structure for the input module of a differential amplifier. One possible layout topology for this module is a cross coupled structure with two interdigital transistors for each input transistor. The layout for this module is shown in Fig. 14 and Fig. 15, respectively. In Fig. 14, the overlap between the drain and source of each transistor has been forbidden just as the overlap (except cross overlap) between the outer source and bulk connection over the gate connections. In contrast to this, the mentioned overlaps have been allowed in Fig. 15. For this reason, the coupling capacitances increase while the area decreases. The designer has to decide which alternative is the best for his circuit. The design assistant supports the designer in assessing

the alternatives because the circuit can be simulated with different topologies including the exact parasitics obtained by an extractor.

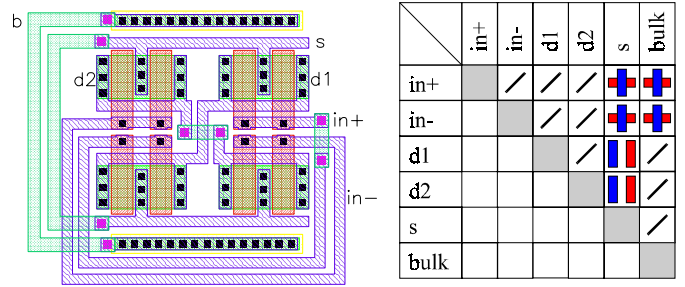


Fig. 14: MOS differential pair and appropriate capacitance sensitivity matrix (forbidden overlaps)

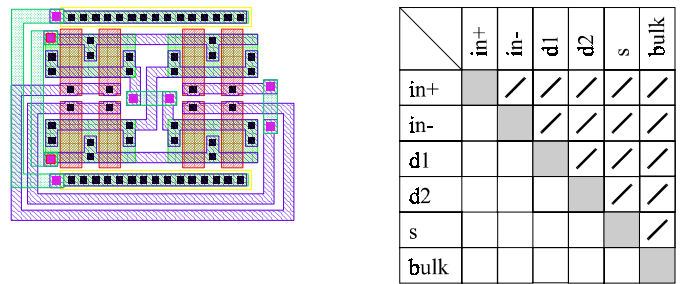


Fig. 15: MOS differential pair and appropriate capacitance sensitivity matrix (overlaps allowed)

The result of the module generation is monitored in a layout editor of DFII. The computation time for the layout of a module is only a few seconds. Hence, it can be used interactively and iteratively for evaluating modules with different parameters or topologies. The layout of operational amplifiers (s. section IV) have been generated with this method. With adequate tools, the subsequent placement and routing can be done manually or automatically. In this assistant the placement is performed by an automatic "schematic-like" placement or by an interactive relative placement of modules using the compaction command. The remaining nets between the complex modules are routed by a point-to-point router.

C. Extraction and Back Annotation of Parasitics

After the layout generation an extraction of parasitic elements can be performed using the extractor tool of Cadence. From this extraction, the line capacitances and transistor capacitances (drain area, drain perimeter etc.) are back annotated into the schematic. The instances for the parasitics of line capacitances are correctly connected because the net names are read from the layout which are identical to the schematic net names. With this extraction the designer gets a good estimation for the parasitics because the subsequent routing of the circuit will not increase these parasitics drastically due to the use of complex modules with inner module routing. The back annotation of the extracted parasitics values is used for a resimulation and optimization of the entire circuit. The resimulation can be done directly with the schematic includ-

ing the additional parasitics. If the designer changes a value to optimize the circuit performance, the layout is automatically regenerated and the parasitics are recalculated. Hence, a direct feedback from the layout generation is provided and the circuit can be optimized more rapidly and more accurately.

After the definition and the layout generation, the modules are stored in the database. The affected modules in the schematic window are highlighted in a specific color denoting the clustering. The module definition can be changed by several modification functions.

IV. RESULTS

In Fig. 16 the schematic of a folded cascode amplifier for low voltage applications [12, 13] is depicted. The simulated performance data of the amplifier are: Power supply voltage 3.3 V, open loop gain 60 dB, 0 dB-bandwidth for 20 pF load capacitance 28 MHz, slew rate 15 V/ μ s, output swing 1 V, and power consumption 52 mW. The partitioning of the modules is indicated by the shaded rectangles. The corresponding layout of the amplifier in a 1.2 μ -technology is shown in Fig. 17. The modules have been created automatically using the new assistant. Identical module generators have been used for the bias network and the cascode stage. The different requirements for the parasitic capacitances are considered with corresponding sensitivity matrices.

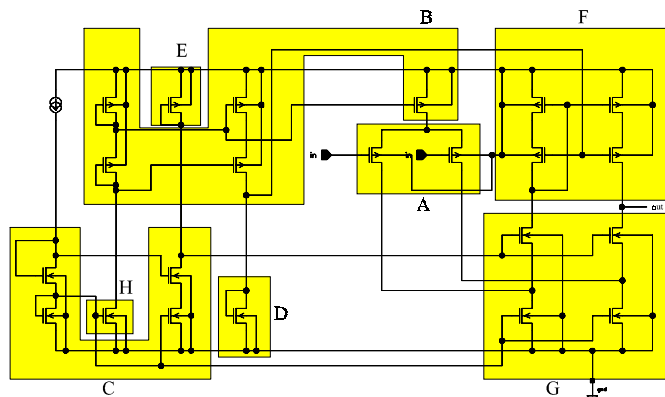


Fig. 16: Schematic of the folded cascode amplifier with partitioning

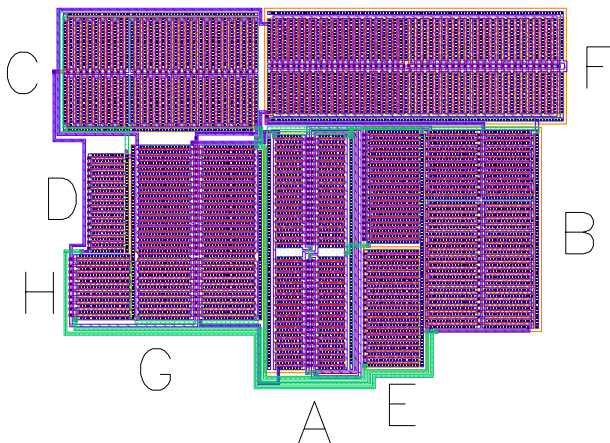


Fig. 17: Layout of the folded cascode amplifier

V. CONCLUSION

In this paper a new simulation and optimization aid for the design of analog integrated circuits has been presented. This environment supports the analog designer in circuit optimization by providing the layout and the extracted parasitic elements already in the optimization phase. Additional electrical properties caused by the layout are taken into account. One example of electrical properties are the RC time constants of gates which are normally not considered. The circuit performance is optimized by controlling the parasitic capacitances already in the design phase of a circuit. Due to the knowledge of the parasitic elements the optimization is much more effective and reliable. If parameters are changed in the schematic after a resimulation step, the affected modules are rebuilt and the parasitics are automatically recalculated. Therefore, the actual value of parasitic elements is always present in the circuit schematic. Due to parameterizable, technology and application independent modules, the number of module generators can be kept small. The usability of the environment has been demonstrated with the creation of a layout for a low voltage folded cascode amplifier.

REFERENCES

- [1] J. Rijmenants, et al., "ILAC: An Automated Layout Tool for Analog CMOS Circuits," *IEEE J. Solid-State Circuits*, Vol. 24, No. 2, pp. 417-425, April 1989.
- [2] H. Y. Koh, et al., "OPASYN: A Compiler for CMOS Operational Amplifiers," *IEEE Trans. Computer-Aided Design*, Vol. 9, No. 2, pp. 113-125, Feb. 1990.
- [3] J. M. Cohn, et al., "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing," *IEEE J. Solid-State Circuits*, Vol. 26, No. 3, pp. 330-342, March 1991.
- [4] V. Meyer zu Bexten, et al., "ALSYN: Flexible Rule-Based Layout Synthesis for Analog IC's," *IEEE J. Solid-State Circuits*, Vol. 28, No. 3, pp. 261-268, March 1993.
- [5] W. Schardein, et al., "Analog Module Generator for Effective Design Assistance," *Proc. ESSCIRC'94*, Ulm, pp. 160-163, Sept. 1994.
- [6] J. Kuhn, "Analog Module Generators for Silicon Compilation," *VLSI System Design*, pp. 75-80, May 1987.
- [7] A. Greiner, F. Pétrot, "Using C to Write Portable CMOS VLSI Module Generators," *EURO-DAC 1994*, pp. 676-681, 1994.
- [8] B. R. Owen, et al., "BALLISTIC: An Analog Layout Language," *IEEE Custom Integrated Circuits Conference*, 1995, pp. 3.5.1-3.5.4.
- [9] M. Wolf, et al., "A Novel Analog Module Generator Environment," *Proc. The European Design & Test Conference*, pp. 388-392, March 1996.
- [10] M. Wolf, et al., "Application Independent Module Generation in Analog Layouts," *Proc. The European Design & Test Conference*, p. 624, March 1997.
- [11] U. Choudhury and A. Sangiovanni-Vincentelli, "Automatic Generation of Parasitic Constraints for Performance-Constrained Physical Design of Analog Circuits," *IEEE Transactions on CAD of Integrated Circuits and Systems*, pp. 208-224, Feb. 1993.
- [12] P. E. Allen et al., "CMOS Analog Circuit Design," Holt, Rinehart at Winston, 1987.
- [13] T. C. Choi et al., "Highfrequency CMOS Switched-capacitor filters for commutation applications," *IEEE. J. Solid-State Circuits*, Vol. 18, pp. 652-664, Dec. 1983.
- [14] B. Ahuja, "Implementation of Active Distributed RC Anti-Aliasing/Smoothing Filters" *IEEE. J. Solid-State Circuits*, Vol. SC-17, No. 6, pp. 1076-1080, Dec. 1983.