A Python-Based Layout-Aware Analog Design Methodology For Nanometric Technologies.

Stéphanie Youssef, Farakh Javid, Damien Dupuis, Ramy Iskander and Marie-Minerve Louerat
University Pierre et Marie Curie (UPMC), LIP6 Laboratory, 4, Place Jussieu, 75005 Paris, France
Email : stephanie.youssef@lip6.fr

Abstract—**This paper presents a methodology for procedural layout-aware design for nanometric technologies. A Python-based layout generation tool generates different layout styles for the same basic analog building blocks. Moreover, layout dependent parasitic parameters such as stress effects are easily computed and compared for different layout styles. The procedural layout description is written using a Python API that ensures layout portability over different technologies. A main focus is on how the layout generation tool addresses both geometric and parasitic-aware electrical synthesis. This is made possible through an internal loop that links circularly both the sizing phase and the layout generation phase. The proposed design methodology assists the analog designer in exploring electrical and physical trade-offs. At the end, we present synthesis and characterization results that prove the effectiveness and speed of the proposed methodology.**

I. Introduction

The demanding of low-power design and a reduced time to market are the key of success for microelectronics industry. There are many CAD tools in the digital domain that continue to improve and ensure the security of the future of the digital designers. In analog domain this is more complex. The main obstacle is that unlike the digital circuits, analog circuits are very sensitive to small variations and the performance may be affected easily. This is why most of analog designers traditionally size manually their circuits and perform a full custom layout which is laborious and error-prone. Another important point is that with the migration to deep-sub-micron (DSM) technologies, providing less area, less power consumption and higher speed integrated circuits, two important constraints related to the layout of the circuit have to be taken into consideration:

a) The problem of analog device matching: Due to the manufacturing process, it is hard to accurately control large transistors' widths. Transistor folding technique is commonly used to reduce parasitic capacitance and gate resistance [1], [2] allowing more accurate geometries and providing better electrical performance. Interdigitation and mirror styles are usually used to equally distribute process gradients along the device.

b) The Shallow Trench Isolation (STI): The DSM technologies use Shallow Trench Isolation (STI) for its accurate dimension control when compared with LOCOS isolation [3]. STI is implemented in the form of trenches etched into the wafer and filled with silicon dioxide to isolate the active area of the transistors. Although STI provides some degree of latch-up protection [4], this isolation technique induces mechanical stress on the transistor and hence degrades its performance [5]. As shown in [6] and [7], this mechanical stress is highly dependent on the layout style being used. To reduce the impact of mechanical stress, the layout must be designed so that all the transistors of the device are affected in the same way.

To keep a hard link between the electrical and the physical view, the analog designer needs to have a precise estimation of the impact of the layout on transistor characterization. This makes the design more complex to be performed using the manually traditional flow.

The automation of fundamental analog design steps, by providing a two way-communication between the electrical and the physical views, is extremely relevant for the success of a project. Several works, such as [8], [9], [10] and [11], target the automation of the layout generation of analog circuits.

To our knowledge, very few number of tools provide the designer with a fast and accurate way to realize different layouts for the same analogue atomic function. A remarkable progress has been made by CIRANOVA [12] which develops parametrized cells in Python Language, known as *PyCells*. PyCells deliver physical views in OpenAccess [13] which is Cadence standard and interoperable database. Many EDA startups have invested in developing OpenAccess native applications. OpenAccess is currently being pushed as a standard database for the EDA industry. OpenAccess assures interoperability, speed, usability. Yet, interoperability standards still need to be agreed upon for this to be true.

In this paper, we will focus on developing a library of analog basic building blocks called *devices* that supports different layout styles (interdigitated, mirror, M2 module and 2D common centroid). We propose a methodology that allows to size and bias a device and generate different layout styles seamlessly. The idea is to well characterize the electrical and the physical parameters of the device to meet functional and robustness constraints [14]. Based on [15], sizing and biasing operators are integrated in a loop with very fast nanometric layout generation tool that allows to describe device layouts in Python. The sizing operators propose sizes to the layout generation tool. This in turn realizes the layout for a given style. Then it computes physical sizes, stress effects as well as layout dependent parameters. These are then fed back to the sizing operator to be taken into account in the next iteration. We show that the flow is very simple and achieves satisfactory results in a matter of seconds. The advantage of the proposed flow is to couple seamlessly and in a procedural manner both transistor sizing and nanometric layout generation, with strong focus on the device intrinsic performance.

The paper is organized as follows: Section II introduces the principle of the automation of analog layout generation tool and shows how it interacts with the electrical view. Section III presents our analog layout graphical interface and an application for the loop between physical and electrical view. Finally, section IV concludes the paper.

II. Exploring electrical and physical trade-offs for analog design

In this section we show how a *layout description language* may help to generate a parametrized layout for analog basic building blocs. We present how our layout generation tool provides layout dependent parameters (LDP). We show also how these parameters are sent to electrical view to take into consideration their effects during the sizing phase.

A. Smart analog basic blocs

In this work we focus on developing smart parametrized generators for analog basic blocks. Each basic block contains an electrical view and a physical view.

1) Electrical View: To offer a large possible choice to the designer, the device comes with an electrical API. This API consists of a set of operators. These operators have two goals: the first set is dedicated to size the device according to some input specification and the second one is dedicated to analyze the electrical behavior of the device, taking into account all the details of physical realization.

Sizing and biasing operators are based on the transistor compact model equations. Operators are used in both *sizing* and *analysis* phases. In the sizing phase, the operator computes unknown widths and biases (Table I, where $V_{EG} = V_{GS} - V_{TH}$) according to input parameters set by the designer. A sizing operator computes either $W = f_W(Temp, I_{DS}, L, V_{GS}, V_{DS}, V_{BS})$, or $V_{GS} = f_{V_{GS}}(Temp, W, L, I_{DS}, V_{DS}, V_{BS})$, where f_W and $f_{V_{GS}}$ are two partial inverse functions of the compact model $I_{DS} = f_{MODEL}(Temp, W, L, V_{GS}, V_{DS}, V_{BS})$. *MODEL* is a standard transistor model like BSIM3V3, BSIM4, PSP, EKV. Sizing operators use simulator encapsulation [15], ensuring accurate computed results. During the analysis phase, the OPIDS operator is used to compute the current as well as the small signal parameters taking into account the layout dependent parameters.

2) Physical View: The device comes also with a physical API based on Python language. This choice was motivated by the fact that Python is an easy to learn, object-oriented, portable and interpreted language. This allows the designer to write concise and simple code to describe complex layouts. The physical view has the following features:

- A layout description language based on python code that eases and simplifies the coding of the device.
- Each analog basic block has different layout styles. Their layout dependent parameters can be computed for different technologies.

TABLE I
Class definition of sizing & biasing operators

Operator	Definition
$OPVS(V_{EG}, V_B)$ ...	$(Temp, I_{DS}, L, V_{EG}, V_D, V_G, V_B) \mapsto (V_S, W, V_{TH})$ ...
$OPVG(V_{EG})$ ...	$(Temp, I_{DS}, L, V_{EG}, V_D, V_S) \mapsto (V_G, W, V_{TH}, V_B)$ ...
$OPVGD(V_{EG})$ ...	$(Temp, I_{DS}, L, V_{EG}, V_S) \mapsto (V_G, V_D, W, V_{TH}, V_B)$ ...
$OPW(V_G, V_S)$ ...	$(Temp, I_{DS}, L, V_D, V_G, V_S) \mapsto (W, V_{TH}, V_B)$ ...
$OPIDS(V_G, V_S)$ ...	$(Temp, W, L, V_D, V_G, V_S) \mapsto (I_{DS}, V_{TH}, V_B)$ ...

- The analog basic building blocks are fully parametrized and reusable.
- There is a hard link between physical and electrical views.

B. Stack object

As previously mentioned, folding technique is commonly used in analog circuits. Since this structure is essential, we have defined a 'Stack' object in our layout generation tool. To create the layout of a complete stack, the designer of parametrized analog devices (folded transistors, differential pair and current mirror, etc ...) simply calls `createStack()` method with well specified input parameters.

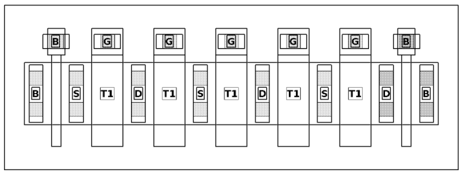


Fig. 1. Layout stack example W=2.0μm, L=0.2μm, NFs=7, Type=NMOS and NDummies=1.

The input parameters of the `createStack()` method are:

- **Type:** the type of the transistor NMOS or PMOS.
- **W:** the overall width of the transistor.
- **L:** the length of each finger.
- **NFs:** the number of stack's fingers (including dummies).
- **NDummies:** the number of dummies at each stack ends.

Fig. 1 presents an example of a generated stack layout. The routing is not shown for clarity. The labels "T1" on the fingers represent the transistor to which the fingers belong. Once a stack object has been created, it can be queried for useful layout distances as shown in Fig. 2.

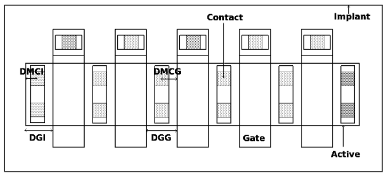


Fig. 2. Useful distances provided by the Stack object.

The distances provided by the stack are :

- **DMCI:** distance from the middle diffusion contact till the isolation edge.
- **DMCG:** distance from the middle diffusion contact till the gate edge.
- **DGG:** distance between two successive gates. This is equal to $2 \times DMCG$.
- **DGI:** distance from the edge of the end gate to the isolation edge. This is equal to $DMCI + DMCG$.

Each distance has a method to query it in the stack object.

C. Layout Dependent Parameter Computation Methods

A dedicated Python API has been developed to describe the device layout. In addition to the methods describing the layout, three special methods have been developed to compute the layout dependent parameters of the MOS transistor model. The first one computes the area and perimeter of the source and drain zones, the second one computes the stress effect parameters as introduced in BSIM4 [16] and the third one computes the capacitance of the routing wires.

D. Stress effect parameter computation

1) The stress effects for a transistor: In the BSIM4 model [16] the stress effect parameters are SA, SB, SD as shown in Fig. 3.

- **SA:** Distance from the first left gate edge at the left end of the stack till the isolation edge at the left end of the stack. This is computed using:

$$SA = DGI + NB_{dummy} \times (L_{dummy} + DGG) \quad (1)$$

  where NB_{Dummy} is the number of dummies and L_{dummy} is the dummy transistor length.
- **SB:** Distance from the first right gate edge at the right end of the stack till the isolation edge at the right end of the stack. This is computed using:

$$SB = DGI + NB_{Dummy} \times (L_{dummy} + DGG) \quad (2)$$

  where NB_{Dummy} is the number of dummies and L_{dummy} is the dummy transistor length.
- **SD:** Distance between two successive gates. This is set equal to DGG.

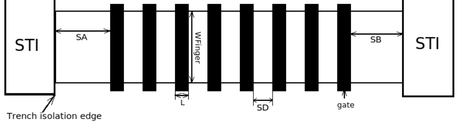


Fig. 3. The stress effect parameters for the transistor.

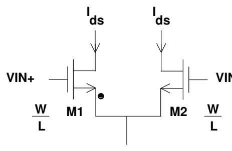


Fig. 4. Differential Pair Schematic View.

2) The stress effects for a Differential Pair: When considering the effects of mechanical stress on a differential pair, the analysis differs significantly from the standalone transistor. For the case of a standalone transistor, all the fingers belong to the same device. On the other hand, a differential pair requires the matching of two different transistors. The calculation of the stress effects parameters becomes more complicated as it deals with matched fingers from different transistors. In this case, the calculation of stress parameters for a differential pair depends on the layout style chosen for the differential pair. Fig. 5 shows a differential pair consisting of transistors T1 and T2. In the following, we discuss the stress effects calculations for the layout styles: *mirror* and *interdigitation*.

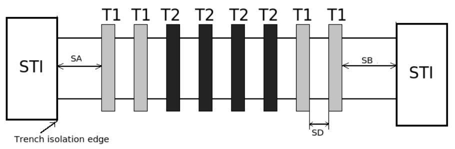


Fig. 5. The stress effect parameters for the mirror style.

The stress effect parameters for differential pair can be calculated as the transistor device by treating each transistor (T1 and T2) separately using the equations (3)-(6).

$$Inv_{SA,T1} = \sum_{i=0}^{(NFS-1)} \frac{\delta i}{SA + 0.5 \cdot L_{drawn} + i \cdot (SD + L_{drawn})} \quad (3)$$

$$Inv_{SB,T1} = \sum_{i=0}^{(NFS-1)} \frac{\delta i}{SB + 0.5 L_{drawn} + i \cdot (SD + L_{drawn})} \quad (4)$$

$$SA_{eff} = \frac{1}{Inv_{SA}} \quad (5)$$

$$SB_{eff} = \frac{1}{Inv_{SB}} \quad (6)$$

We have improved the BSIM4 model to take into account that the calculations of one transistor change according to the position of its fingers. Note that the summation is carried out on the range from 0 to $NFS - 1$. The presence of the fingers of the other transistor alternating to the calculated transistor's fingers creates what we called holes . This requires the introduction of a new parameter δ in the initial formula of BSIM4. The parameter δ takes the value of 1 when pointing to a finger of the transistor considered, and the value of 0 when pointing to a finger of the other transistors.

The stress effects affect model parameters such as the effective mobility μ_{eff}, the velocity saturation V_{sat} and the

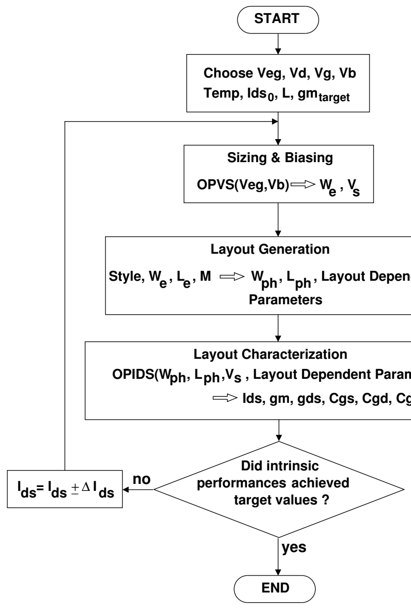

START
Choose Veg, Vd, Vg, Vb
Temp, Ids0, L, gmtarget
Sizing & Biasing
OPVS(Veg,Vb) ⟹ We , Vs
Layout Generation
Style, We , Le , M ⟹ Wph , Lph , Layout Dependent Parameters
Layout Characterization
OPIDS(Wph, Lph,Vs , Layout Dependent Parameters)
⟹ Ids, gm, gds, Cgs, Cgd, Cgb
Did intrinsic performances achieved target values ?
no
Ids= Ids ±Δ Ids
yes
END

Fig. 6. Sizing and layout generation design flow

threshold voltage V_{th} [17], [18], [19]. To reflect the influence of SA_{eff} and SB_{eff}, the parameter "α" is defined as follows:

$$\alpha = \frac{1}{\frac{1}{2\dot{S}A_{eff}} + \frac{1}{2\dot{S}B_{eff}}} \tag{7}$$

Notice how stress effects, quantified by $1/\alpha$ decrease by increasing NF.

E. The design flow

Figure 6 illustrates how the operators and the layout method are used to implement a device while respecting specifications. In the first step, given the temperature, the biasing current I_{DS}, the overdrive gate voltage V_{EG}, the drain voltage V_D, the gate voltage V_G, the bulk voltage V_B and the transistor length L, the operator OPVS is used to compute the electrical width W_e, the source voltage V_S and the threshold voltage V_{TH}. The width W_e, length L_e, number of fingers M and layout style are given to the layout generator. Once the layout is generated, the actual physical width W_{ph} and length L_{ph} as well as the layout dependent parameters (diffusion zone, stress and routing) are available. An accurate characterization, including the actual physical realization, is performed using the OPIDS operator that provides the actual I_{DS} and the small signal parameters for the purpose verification.

Here, we choose to take the g_m as a specification for the case of the differential pair. If the g_m value does not meet the specification, a loop is set to adjust the biasing current till the specifications are achieved. After convergence, the final layout is then realized.

Note that in the flow, all parameters varies, except the layout style and the number of fingers that are kept invariant in this loop.

III. Results

A. Layout generation tool

In Fig. 7 we show the graphical user interface for the layout generation tool CHAMS/Pharos. The upper left widget, labeled *Device Explorer* shows the different layout parameters set by the designer. The lower left message console prints the values of all the layout dependent parameters. The script editor in the lower right side of the GUI allows editing and executing Python code.

Once the generation of the layout of the device is performed, and thanks to the layout dependent parameter computation methods, we have immediately the generic curves of the layout dependent parameters versus NF as shown in Fig.8 and Fig.9:

a) Mirror style: Fig. 8 illustrates the generated curve $1/\alpha$ versus NF for NMOS differential pair in mirror style in 65nm technology with W = 6μm and L = 0.15μm.

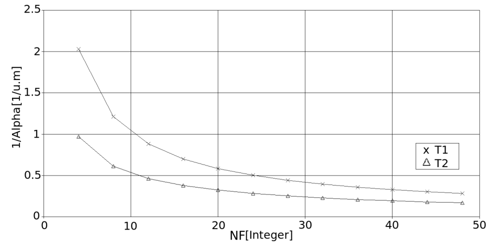


Fig. 8. The stress effect $1/\alpha$ parameter versus NF of the Mirror style.

b) Interdigitated style: Fig. 9 illustrates the generated curve $1/\alpha$ versus NF for NMOS differential pair in interdigitated style in 65nm technology with W = 6μm and L = 0.15μm. We can observe in Fig. 9 that, for the interdigitated

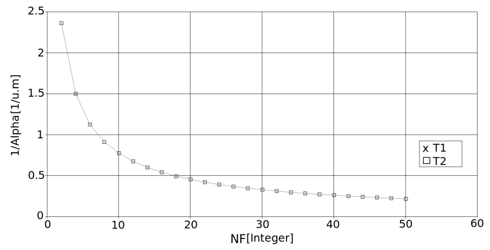
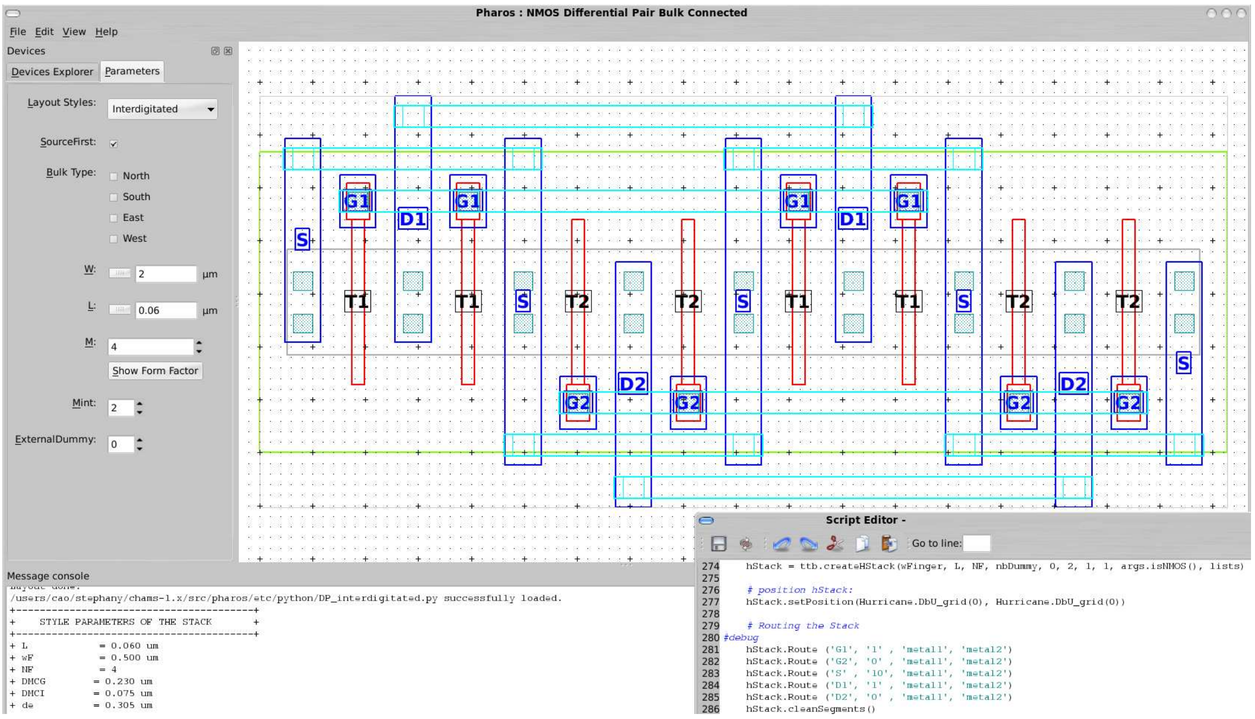


Fig. 9. The stress effect $1/\alpha$ parameter versus NF of the interdigitated style.

technique, the stress effect affects the two transistors the same way. While, for the mirror technique in Fig. 8, T1 transistor, placed at the extremities of the stack, suffers a more significant stress effect than transistor T2. This is due to the fact that T1 is closer to the STI. So, the interdigitated technique in the nanometer technology is more preferable unlike the older technologies that prefer the mirror techniques.

B. Case study: Design of a Differential Pair

The goal is to design a differential pair in a CMOS 65nm technology with a specified g_m equals to 0.37 mS. The assumption: V_{EG},V_D, V_G, V_B and the transistor length L (set



Pharos : NMOS Differential Pair Bulk Connected

Layout Styles: Interdigitated

SourceFirst: ✓

Bulk Type: North, South, East, West

W: 2 µm

L: 0.06 µm

M: 4

Mint: 2

ExternalDummy: 0

Message console

```
/users/cao/stephany/chams-1.x/src/pharos/etc/python/DP_interdigitated.py successfully loaded.
+---------------------------------------+
+    STYLE PARAMETERS OF THE STACK      +
+---------------------------------------+
+ L              = 0.060 um
+ wF             = 0.500 um
+ NF             = 4
+ DMCG          = 0.230 um
+ DMCI          = 0.075 um
+ de            = 0.305 um
```

Script Editor -

```
    hStack = ttb.createHStack(wFinger, L, NF, nbDummy, 0, 2, 1, 1, args.isNMOS(), lists)

    # position hStack:
    hStack.setPosition(Hurricane.DbU_grid(0), Hurricane.DbU_grid(0))

    # Routing the Stack
#debug
    hStack.Route ('G1', '1' , 'metal1', 'metal2')
    hStack.Route ('G2', '0' , 'metal1', 'metal2')
    hStack.Route ('S' , '10', 'metal1', 'metal2')
    hStack.Route ('D1', '1' , 'metal1', 'metal2')
    hStack.Route ('D2', '0' , 'metal1', 'metal2')
    hStack.cleanSegments()
```

Fig. 7. CHAMS/Pharos layout generation tool environment

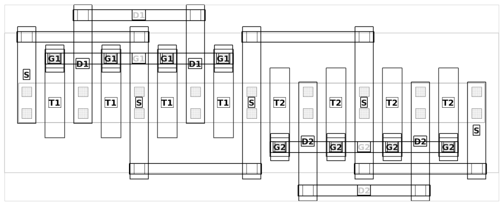


Fig. 10. Differential Pair (interdigitated layout style) 65nm

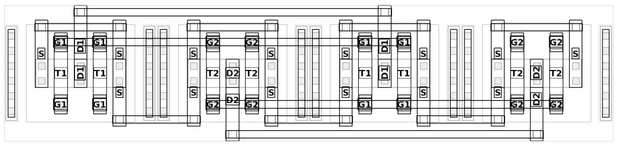


Fig. 11. Differential Pair (M2 module layout style) 65nm

to $3.L_{min}$), are known. The design is performed for 4 different layout styles with 4 fingers.

The results of the execution of the loop in Fig. 6 is presented in the case of the four layout styles, Table II for the interdigitated style, Table III for the mirror, Table IV for the M2-module style and Table V for the 2D common-centroid. The $Iter$ parameter is the number of iteration in the loop, W_{ph} is the transistor width, I_{DS} is the drain current biasing the transistor, g_m is the transistor transconductance and SA_{eff}, SB_{eff} and $1/\alpha$ are the stress parameters defined by the BSIM4 model. The corresponding layouts are presented in Fig. 10 (interdigitated), Fig. 11 (M2 module), Fig. 12 (2D common-centroid) and Fig. 13 (mirror) respectively.

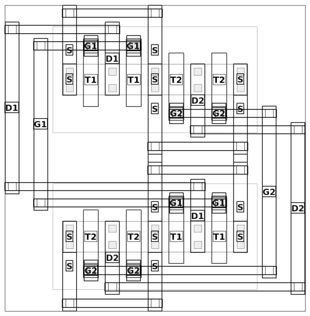


Fig. 12. Differential Pair (2D common centroid layout style) 65nm

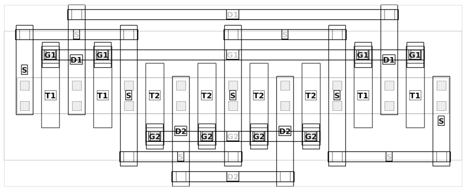


Fig. 13. Differential Pair (mirror layout style) 65nm

Let us examine Table V for the differential pair with 2D Common Centroid style . After all the loop iterations to get the required g_m, we can get many information about the electrical and the physical parameters such as the electrical intrinsic capacitances: $C_{gs} = 2.39fF$, $C_{ds} = 1.16fF$, $C_{gd} = 0.558fF$, $C_{gb} = 0.37fF$, etc ... Also the physical parasitic routing capacitances: $C_{g_{ph}} = 0.0581fF$,$C_{d_{ph}} = 0.0937fF$,$C_{s_{ph}} = 0.03.89fF$, etc ... related to the net gate

TABLE II
RESULTS 65NM INTERDIGITATED (T2 TRANSISTOR)

Iter	W_{ph} (μm)	I_{DS} (μA)	g_m (mS)	SA_{eff} (μm)	SB_{eff} (μm)	$1/\alpha$ (μm^{-1})
1	0.9	29.37	0.245	2.22	0.81	0.842
2	0.93	29.98	0.25	2.22	0.81	0.842
27	1.43	43.64	0.364	2.22	0.81	0.842
28	1.45	44.19	0.369	2.22	0.81	0.842

TABLE III
RESULTS 65NM MIRROR (T2 TRANSISTOR)

Iter	W_{ph} (μm)	I_{DS} (μA)	g_m (mS)	SA_{eff} (μm)	SB_{eff} (μm)	$1/\alpha$ (μm^{-1})
1	0.9	29.60	0.247	1.99	1.99	0.5025
2	0.93	30.30	0.253	1.99	1.99	0.5025
26	1.41	43.62	0.363	1.99	1.99	0.5025
27	1.43	44.18	0.368	1.99	1.99	0.5025

TABLE IV
RESULTS 65NM M2 (T2 TRANSISTOR)

Iter	W_{ph} (μm)	I_{DS} (μA)	g_m (mS)	SA_{eff} (μm)	SB_{eff} (μm)	$1/\alpha$ (μm^{-1})
1	0.9	28.24	0.236	0.48	0.48	2.08
2	0.94	29.07	0.243	0.48	0.48	2.08
29	1.485	43.18	0.362	0.48	0.48	2.08
30	1.505	43.74	0.366	0.48	0.48	2.08

TABLE V
RESULTS 65NM 2D-COMMONCENTROID (T2 TRANSISTOR)

Iter	W_{ph} (μm)	I_{DS} (μA)	g_m (mS)	SA_{eff} (μm)	SB_{eff} (μm)	$1/\alpha$ (μm^{-1})
1	0.9	28.92	0.241	1.59	0.48	1.356
2	0.93	29.52	0.247	1.59	0.48	1.356
28	1.45	43.43	0.363	1.59	0.48	1.356
29	1.47	43.96	0.368	1.59	0.48	1.356

'G', drain 'D' and source 'S' respectively. We can calculate similarly the transition frequency defined as:

$$F_t = \frac{g_m}{2\Pi(C_{gs} + C_{gd} + C_{gb} + C_{g_{ph}} + C_{s_{ph}})} = 17.6GHz$$

The overall computation time for the optimization loop is around 6s (1s for the sizing, 5s for layout generation). The average number of iterations is less than 30.

We conclude from the tables that the target g_m can be achieved under the influence of stress effect for the different layout styles. This has to be compromised with matching requirements in order to choose the final layout style.

IV. CONCLUSION

In this paper, a new method for developing smart parametrized generators for analogue devices has been presented. The interaction between the transistor sizing and the layout generation of the device has been illustrated in the case of the differential pair. Four different layout styles have been compared. The tight coupling between the transistor sizing and the layout generation has been used to accurately characterize a design. The proposed method allows the designer to select the most convenient device layout style for given specifications. The results showed the efficiency of the proposed method. As a future work, the proposed method will be extended to allow a seamless coupling between circuit level sizing and layout generation, taking into account intrinsic device performance.

REFERENCES

[1] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, 2000.

[2] Ravindranath Naiknaware et al, "Automated Hierarchical CMOS Analog Circuit Stack Generation with Intramodule Connectivity and Matching Considerations." *J. Solid-State Circuits*, vol. 34, no. 3, pp. 304–317, March 1999.

[3] D. Andriukaitis et al, "LOCOS CMOS process simulation," in *Conf. Information Technology Interfaces*, Catvat, Croatia, June 2006, pp. 489 –494.

[4] W.L. Goh et al, "A comprehensive geometrical and biasing analysis for latchup in 0.18-μm C_oSI_2 STI CMOS structure." *jssc*, vol. 48, pp. 2109–2114, 2004.

[5] ——, "Latchup characterization of 0.18-micron sti cobalt silicided test structures." *Microelectronics Journal*, vol. 32, no. 9, pp. 725–731, September 2001.

[6] M. Ranjan et al, "Fast, layout-inclusive analog circuit synthesis using pre-compiled parasitic-aware symbolic performance models," in *Proc. Design, Automation and Test in Europe Conf.*, Paris, France, February 2004, pp. 604–609.

[7] V. Joshi et al, "Stress aware layout optimization," in *Proc. Int. Symp. on Physical design*, Portland, Oregon, April 2008, pp. 168–174.

[8] D. Stefanovic, M. Kayal, M. Pastre, and V. B. Litovski, "Procedural analog design (pad) tool," in *Proceedings of the 4th International Symposium on Quality Electronic Design*, ser. ISQED '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 313–. [Online]. Available: http://portal.acm.org/citation.cfm?id=851002.855973

[9] L. Lewyn and N. Williams, "Is a new paradigm for nanoscale analog cmos design needed?" *Proceedings of the IEEE*, vol. 99, no. 1, pp. 3 –6, Jan. 2011.

[10] N. Jangkrajarng, S. Bhattacharya, R. Hartono, and C. J. R. Shi, "Iprail–intellectual property reuse-based analog ic layout automation," *Integration, the VLSI Journal*, vol. 36, no. 4, pp. 237 – 262, 2003, analog and Mixed-signal IC Design and Design Methodologies.

[11] R. Castro-Lopez, O. Guerra, E. Roca, and F. Fernandez, "An integrated layout-synthesis approach for analog ics," *IEEE Trans. Computer-Aided Design*, vol. 27, no. 7, pp. 1179–1189, Jul. 2008.

[12] "http://www.ciranova.com/."

[13] "http://www.cadence.com/."

[14] T. Massier, H. Graeb, and U. Schlichtmann, "The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 12, Dec. 2008, pp. 2209 –2222.

[15] F. Javid, R. Iskander, and M.-M. Louërat, "Simulation-Based Hierarchical Sizing and Biasing of Analog Firm IPs," *IEEE International Behavioral Modeling and Simulation Conference*, pp. 43–48, Sep. 2009.

[16] W. Liu, *MOSFET Models for SPICE Simulation including BSIM3v3 and BSIM4*. Wiley Interscience, 2001.

[17] L. Yang et al, "Simulation of layout-dependent STI stress and its impact on circuit performance," in *Int. Conf. on Simulation of Semiconductor Processes and Devices*, San Diego, USA, September 2009, pp. 281–284.

[18] H. Tsuno et al, "Advanced analysis and modeling of MOSFET characteristic fluctuation caused by layout variation," in *VLSI Technology Symposium*, Kyoto, Japan, June 2007, pp. 204 –205.

[19] V. Moroz et al, "The impact of layout on stress-enhanced transistor performance," in *Int. Conf. on Simulation of Semiconductor Processes and Devices*, Tokyo, Japan, September 2005, pp. 143 – 146.