Simulation-Based Hierarchical Sizing and Biasing of Analog Firm IPs

Farakh Javid, Ramy Iskander and Marie-Minerve Louërat
LIP6-SoC Laboratory, University of Paris VI, Paris, France

Abstract— **This paper presents a simulation-based hierarchical sizing and biasing tool for analog integrated circuits design. The tool allows the designer to express the sizing procedure in terms of sizing and biasing operators. These operators are technology independent, hence the documented procedure can be easily ran over different technologies. A procedure has been proposed for a single-ended two-stage operational amplifier and evaluated over 130nm, 65nm and 45nm technologies. The results prove the efficiency of the proposed tool.**

I. Introduction

The design of analog integrated circuits is still a complicated task compared to digital one. Technology migration particularly implies redesigning the whole circuit whereas digital circuits have complete automatic "silicon compilers" design flow tools. Thus analog design is very time consuming, which raises the problem of reducing time to market. Considering mixed-signal circuits amount will increase in next years, the development of accurate and rapid analog automation tools becomes crucial.

Since decades efforts have been made in that direction. Migration problem is highlighted in [1] where authors implement an automatic circuit resizing engine. Another important topic, DC modeling, is investigated in [2]. Generally we now distinguish three types of analog design tools : simulation-based, knowledge-based and hybrid. The first category includes MAELSTROM [3], ANACONDA [4] or AMIGO [5]. These tools couple an optimizer with a simulator in a loop (Fig.1) : designs are refined by trial-and-error using an iterative time-consuming approach. The simulator acts as performances evaluator. Second category encloses tools like OASYS [6], FEEDS [7] or DONALD [8] where simulator is replaced by inside approximate transistor model. Third category gathers CAIRO+ (Creating Analog IPs Reusable and Optimized) [9] and OCEANE [10] in which inside transistor model is the standard transistor model of the simulator. In the two last categories designs are analysed by experienced designers who extract equations that describe circuit behaviour and evaluate performances. This method is first time-consuming due to equations extraction, then it becomes rapid and robust due to the execution of an automated procedure including these equations.

In this paper we present a new simulation-based hierarchical sizing and biasing tool that employs a commercial electrical simulator. Thus we are first ensured about the accuracy of computation due to the very precise transistor model included in the simulator. Secondly we can reuse previously developed sizing and biasing procedures over newer technologies. Moreover these procedures are technology independent since they abstract the technology by a given set of predefined operators.

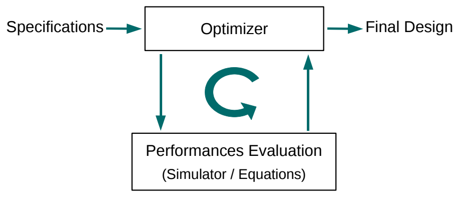

Fig. 1. Abstract model of analog synthesis.

The paper is organized as follows : section II describes hierarchical sizing and biasing method applied to analog design. In section III we present our tool architecture. In section IV we apply this tool on the sizing and biasing of an amplifier with different technologies and transistor models. Conclusion is given in section V.

II. Hierarchical Sizing and Biasing

A. Hierarchical Methodology

The concept of hierarchy is one of the key element in modern digital design efficiency. Indeed a large circuit is divided into smaller sub-circuits, hiding lower-levels details to make circuit understanding easier. In [11] and [12] authors point out the importance to apply this concept to analog design.

The hierarchical sizing and biasing method, described in [13] and [14], is used to automatically compute the DC operating point and generate suitable design plans for an analog circuit. A circuit is built as a hierarchy of sub-circuits, leaf sub-circuits are called *devices* and higher-level sub-circuits are called *modules*, each sub-circuit is represented by a dependency graph. The dependency graph expresses electrical dependencies of sub-circuit DC parameters on a selected set of design parameters. The dependency graph of the analog circuit is constructed, in a hierarchical bottom-up approach, by merging graphs of children devices and modules. The graph is converted to a directed acyclic graph (DAG) which is a directed graph with no directed cycles. The resulting DAG is the design plan of the analog circuit. Upon construction, the DAG is executed in a top-down approach in order to compute the DC operating point and the dimensions of the transistors.

$$\underbrace{\begin{bmatrix} \mathrm{Temp} \\ \mathrm{V_{IN,i}} \\ \mathrm{V_{eg,i}} \\ \mathrm{I_{B,i}} \\ \mathrm{V_{OUT,i}} \\ \mathrm{L_i} \end{bmatrix}}_{\substack{\text{Design}\\ \text{Parameters}}} \Rightarrow \underbrace{\begin{bmatrix} \mathrm{W_1} \\ \mathrm{V_{gs1}} \\ \mathrm{V_{ds1}} \\ . \\ \mathrm{W_n} \\ . \end{bmatrix}}_{\substack{\text{Biases+}\\ \text{Sizes}}} \Rightarrow \underbrace{\begin{bmatrix} \mathrm{g_{m,1}} \\ \mathrm{g_{ds,1}} \\ \mathrm{C_{gs,1}} \\ . \\ \mathrm{g_{m,n}} \\ . \end{bmatrix}}_{\substack{\text{Small -- Signal}\\ \text{Parameters}}} \Rightarrow \underbrace{\begin{bmatrix} \mathrm{A_d} \\ \phi_\mathrm{m} \\ \mathrm{F_t} \\ . \\ . \\ \mathrm{PSRR} \end{bmatrix}}_{\substack{\text{Linear}\\ \text{Performances}}}$$

Fig. 2. Parameter mappings in CAIRO+ design space.

B. Design Parameters

Transistor width is traditionally used as the variable to optimize during synthesis. This leads to a very large design space and waste of time in the evaluation of unfeasible circuits [4]. This is the reason why we argue that width parameter is not the best adapted choice concerning design parameters, i.e. parameters fixed at the beginning by the designer. On the contrary voltages, currents and lengths have a narrower range that strongly reduces the design space and makes computation faster (some minutes versus some hours [3], [9]). Moreover since widths are calculated from these variables (Fig.2), we are assured of the feasibility of the resulting circuit. Thus CAIRO+ employs voltages, currents and lengths as design parameters.

C. Sizing and Biasing Operators

Operators result from the inversion of transistor model equations [15] and are applied to each *device*. Each operator has a set of input parameters that correspond to designer fixed values (first vector in Fig.2, where $V_{eg} = V_{gs} - V_{th}$), and computes several unknown parameters that are widths and biases (second vector in Fig.2) [13]. An operator computes either :

$$W = f^{-1}(temp, I_{DS}, L, V_{GS}, V_{DS}, V_{BS}) \qquad (1)$$

or :

$$V_{GS} = f(temp, W, L, I_{DS}, V_{DS}, V_{BS}) \qquad (2)$$

where f is a monotonic function solvable with Newton-Raphson algorithm. Convergence criteria are the same as those integrated into commercial simulators. To size and bias a complete circuit several operators are needed. In that case a parameter computed by an operator can be an input parameter for another operator. This dependency between parameters is expressed by the DAG that defines a design plan for the circuit (see previous subsection), i.e. the order in which the operators are called. In fact each *device* is individually sized and biased with the appropriate operator, the topology of the circuit being expressed by the DAG.

Table I shows the definition of the main five classes of the sizing and biasing operators. Let us examine one operator such as $OPVS$. The $OPVS$ operator class is *source voltage*. The table shows only two versions of this operator. The first version $OPVS(V_{eg}, V_B)$ is called whenever V_{eg} is known and the reference transistor is not bulk-source connected i.e. V_B should be fixed by the designer. This operator computes V_S, V_{th} and W, simultaneously, in terms of $Temp$, I_{DS}, L, V_{eg}, V_D, V_G and V_B. The second version $OPVS(V_{eg})$ is called whenever V_{eg} is known and the reference transistor is bulk-source connected. This operator also determines V_S, V_B, V_{th} and W, simultaneously in terms of $Temp$, I_{DS}, L, V_{eg}, V_D and V_G.

TABLE I

CLASS DEFINITION OF SIZING & BIASING OPERATORS.

Operator	Definition
OPVS(V_{eg}, V_B)	$(V_S, V_{th}, W) \Leftarrow Temp, I_{DS}, L, V_{eg}, V_D, V_G, V_B$
OPVS(V_{eg})	$(V_S, V_B, V_{th}, W) \Leftarrow Temp, I_{DS}, L, V_{eg}, V_D, V_G$
…	…
OPVG(V_{eg})	$(V_G, V_B, V_{th}, W) \Leftarrow Temp, I_{DS}, L, V_{eg}, V_D, V_S$
…	…
OPVGD(V_{eg})	$(V_G, V_D, V_B, V_{th}, W) \Leftarrow Temp, I_{DS}, L, V_{eg}, V_S$
…	…
OPW(V_G, V_S)	$(W, V_B, V_{th}) \Leftarrow Temp, I_{DS}, L, V_D, V_G, V_S$
…	…
OPIDS(V_{eg})	$(I_{DS}, V_B, V_{th}) \Leftarrow Temp, W, L, V_{eg}, V_D, V_S$
…	…

D. Proposed Methodology

We aim at implementing sizing and biasing operators on top of an electrical simulator to take advantage of very precise available transistor models, avoiding their implementation in our tool. Another key motivation is the possibility to switch technology by simply changing transistor models used by the simulator.

III. CHAMS : A SIMULATOR-BASED TOOL

The architecture of CHAMS (CAIRO+ Hurricane [16] AMS) is given in Fig.3. At the bottom we have an electrical

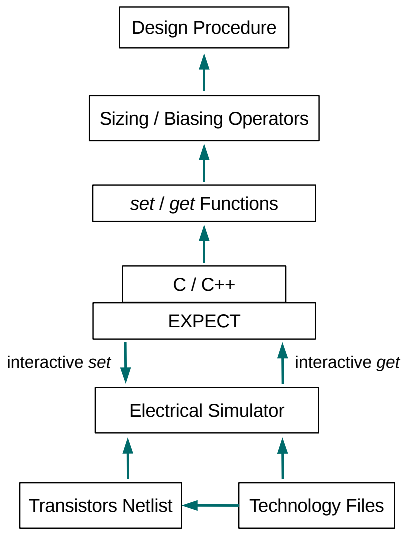

Fig. 3. Architecture of CHAMS.

netlist containing only two transistors (one PMOS and one NMOS), entirely sizable and biasable through simulator interactive commands. This netlist also specifies the desired target technology. It is used by the electrical simulator launched in batch mode. Three types of interactive commands are evaluated : *set*, *get* and *run*. The first one allows to set all transistor parameters (sizes, biases, temperature, number of fingers). The second one enables to catch all currents, voltages and small signal parameters values. After any *set* command a simulation must be run, using *run* command, to compute the DC operating point of the transistor and update all parameters values.

Then we use *expect* library [17] to encapsulate the simulator into our tool. We define a C++ control language of the simulator that allows to automate *set* and *get* commands in batch mode. At last we develop a C++ group of "set/get" functions similar to object-oriented programming ideas in order to hide simulator commands details. These functions are used by sizing and biasing operators.

All operators have been optimized to minimize the number of calls to the simulator, which can reach several hundreds during sizing. We point out that these operators are totally technology independent. Knowing that more and more complicated transistor models appear frequently, being able to use them in a short time to migrate existing circuits is a great improvement. Indeed only the netlist including the two transistors has to be changed with corresponding existing technology files.

The pseudo-code of $OPVS(VEG)$ operator is given in Fig.4 to illustrate simulator encapsulation. *computeW()* (line 36) is the function described by equation (1). *EPSILON_V* and *EPSILON_W* (lines 44, 45) define convergence criteria.

IV. Sizing and biasing examples

A design procedure using CHAMS has been written for the sizing and biasing of a single-ended two-stage operational amplifier (Fig.5). This circuit can be decomposed into six *devices*, thus we will employ six operators. For the first stage we have a differential pair (M_1, M_2), a current mirror (M_3, M_4) and a biasing transistor (M_5). The second stage is composed by a load transistor (M_6) and a biasing transistor (M_7). M_8 is the biasing circuit transistor. We start with 130nm technology then migrate to 65nm and 45nm.

We performed all computations with a temperature of $27°C$, and we have :

$$K = \frac{I_{DS,M7,M6}}{I_{DS,M5}} \tag{3}$$

We assume the amplifier is designed to minimize the systematic offset voltage V_{off} :

$$\begin{array}{lllll} V_{D,M1} & = & V_{D,M2} & = & V_{G,M6} \\ L_{M3} & = & L_{M4} & = & L_{M6} \\ L_{M8} & = & L_{M5} & = & L_{M7} \end{array} \tag{4}$$

A linear performance parameter, the DC gain, is evaluated with CHAMS using the approximate expression :

$$Ad_0 = -\frac{g_{m1}}{g_{ds2}+g_{ds4}}\frac{g_{m6}}{g_{ds6}+g_{ds7}} \tag{5}$$

```
1   OPVS_VEG (trName, nfing, temp, l, ids, veg, vg, vd, & w,
2           & vth, & vs, & vb, wmin, wmax)
3   {
4       // Declaration of intermediate variables
5       wp = 0.0 ; vsp = 0.0 ;
6       vgs = 0.0 ; vds = 0.0 ; vbs = 0.0 ;
7
8       // Set all input parameters in simulator environment
9       setTEMP(temp) ; setNFING(nfing) ;
10      setL(l) ; setW(10*wmin) ;
11      setVDS(vds) ; setVBS(vbs) ; setVGS(vgs) ;
12
13      // Call the simulator to run a DC simulation
14      run() ;
15
16      // Call the simulator to get a parameter
17      vth = getVTH(trName) ;
18
19      vs = 0.0 ; w = wmin ;
20
21      do {
22          // Update all intermediate variables
23          wp = w ;
24          vsp = vs ;
25          vb = vs = vg - veg - vth ;
26          vgs = vg - vs ;
27          vds = vd - vs ;
28          vbs = vb - vs ;
29
30          // Call the simulator to set all intermediate variables
31          setVGS(vgs) ;
32          setVDS(vds) ;
33          setVBS(vbs) ;
34
35          // Call the simulator to compute W
36          w = computeW(trName, temp, l, w, nfing, ids, vgs,
37                  vds, vbs, wmin, wmax) ;
38
39          setW(w) ;
40          run() ;
41
42          vth = getVTH(trName) ;
43      }
44      while( |vs - vsp| >= EPSILON_V
45          or |w - wp| >= EPSILON_W)
46  }
```

Fig. 4. Pseudo-code of $OPVS(VEG)$ sizing operator.

A transient parameter, slew rate, is also computed with the following expression :

$$SR = \frac{I_{DS,M5}}{C_c} \tag{6}$$

with $C_c = 2.9pF$. Equations (3), (4), (5) and (6) express designer knowledge.

We start by computing all widths and $V_{G,M5}$ (which is equal to $V_{G,M7}$, $V_{G,M8}$, and $V_{D,M8}$). Then we enter these values into the electrical netlist of the complete amplifier and run a DC simulation. To validate CHAMS results we compare fixed currents and voltages with those computed from simulation.

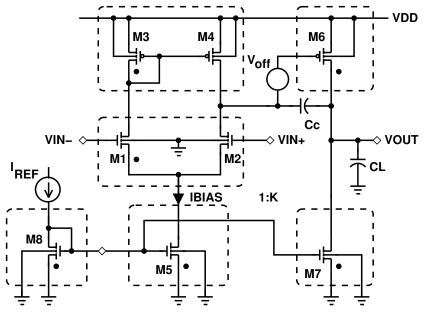

Fig. 5. Single-ended two-stage operational amplifier ($V_{off} = 0V$). Each dashed-box is a *device*.

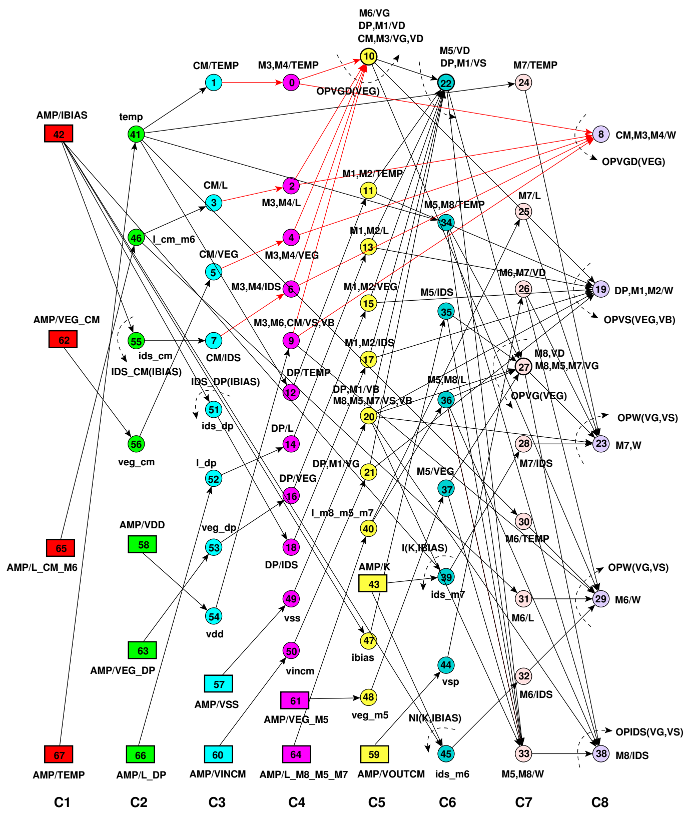

Fig. 6. Electrical parameters dependency graph for single-ended two-stage operational amplifier : rectangle nodes are the known parameters, bold circle nodes with dotted arrows (8, 19, 23, 27, 29, 38) are the operators and other nodes are temporary variables used for parameter propagation.

Results are listed in Tables III, IV, V and VI where W is the total width of the transistor and M the number of fingers. DC gain, slew rate (both computed by CHAMS and simulation), transition frequency F_T and phase margin (both obtained with simulation) are given in Tables VII and VIII.

A. Design Procedure

For the amplifier of Fig.5, the dependency between electrical parameters is expressed by the graph in [13] that is shown in Fig.6. For the current mirror, $OPVGD(V_{eg})$ computes W_3, W_4 and $V_{G/D,CM}$ in nodes (C5,10) and (C8,8). For the differential pair, $OPVS(V_{eg}, V_B)$ computes W_1, W_2 and $V_{S,DP}$ in nodes (C8,19) and (C6,22). For transistor M_5,

```
    SIZING_PROCEDURE ([design_parameters]) {

        netlist = "np_mos.netlist" ; // Netlist name

        // Transistors names in the netlist
        name_n = "X_trN" ;
        name_p = "X_trP" ;

        start() ; // Start the simulator

        // Load the two-transistors netlist
        load(netlist) ;

        // Step 1) Current Mirror : M3,M4
        [w_cm,vg_cm,vd_cm,vb_cm,vth_cm] =
            OPVGD_VEG(name_p,[design_parameters]) ;

        // Step 2) Differential Pair : M1,M2
        [w_dp,vs_dp,vth_dp] =
            OPVS_VEG_VB(name_n,[design_parameters],vd_cm) ;

        // Step 3) M5
        [w_m5,vg_m5,vb_m5,vth_m5] =
            OPVG_VEG(name_n,[design_parameters],vs_dp) ;

        // Step 4) M8
        [ids_m8,vb_m8,vth_m8] =
            OPIDS_VG_VS(name_n,[design_parameters],w_m5) ;

        // Step 5) M6
        [w_m6,vb_M6,vth_m6] =
            OPW_VG_VS(name_p,[design_parameters],vd_dp) ;

        // Step 6) M7
        [w_m7,vb_m7,vth_m7] =
            OPW_VG_VS(name_n,[design_parameters],vg_M5) ;

        // Quit the simulator
        stop() ;
    }
```

Fig. 7. Pseudo-code of the design procedure.

TABLE II
INPUT PARAMETERS VALUES.

Parameter	BSIM3V3 (130nm) & BSIM4 (65nm)	BSIM4 (45nm)	PSP (45nm)
V_{DD} (V)	1.2	0.99	1.1
V_{SS} (V)	0.0	0.0	0.0
$L_{M1,M2}$ (nm)	340	280	280
$L_{M3,M4,M6}$ (nm)	340	130	130
$L_{M5,M7,M8}$ (nm)	340	190	190
$I_{DS,M5}$ (μA)	30	25	25
$I_{DS,M7}$ (μA)	150	125	125
$V_{eg,M1}$ (V)	0.12	0.05	0.05
$V_{eg,M3}$ (V)	-0.12	-0.1	-0.12
$V_{eg,M5}$ (V)	0.1	0.08	0.08
$V_{in,CM}$ (V)	0.6	0.495	0.55
$V_{out,CM}$ (V)	0.6	0.495	0.55

TABLE III

CHAMS VERSUS SIMULATION RESULTS (BSIM3V3 130NM).

Parameter		M_1, M_2	M_3, M_4	M_5	M_6
$I_{DS}\ (\mu A)$	CHAMS	15.0	-15.0	30.0	-150.0
$I_{DS}\ (\mu A)$	Simulation	14.999	-14.999	29.999	-150.0
$V_{eg}\ (V)$	CHAMS	0.12	-0.12	0.1	not given
$V_{eg}\ (V)$	Simulation	0.12	-0.119	0.1	-0.12
$V_{GS}\ (V)$		0.481	-0.473	0.453	-0.473
$V_{DS}\ (V)$		0.607	-0.473	0.118	-0.6
$V_{BS}\ (V)$		-0.118	0.0	0.0	0.0
$V_{th}\ (V)$		0.361	-0.353	0.353	-0.353
$V_{dsat}\ (V)$		0.119	-0.119	0.105	-0.119
$g_m\ (mA/V)$		0.190	0.190	0.398	1.902
$g_{ds}\ (\mu A/V)$		2.271	2.227	68.667	17.756
$C_{gd}\ (fF)$		1.038	2.588	3.615	24.557
$W\ (\mu m)/M$	CHAMS	2.072/4	6.320/2	6.640/4	62.196/20

$OPVG(V_{eg})$ computes $V_{BIAS} = V_{G,M_5}$ and W_5 in node (C7,27) and (C7,33). Despite that V_{BIAS} is not a given design parameter, it is automatically computed in the graph. For transistor M_8, imposing the constraint : $W_8 = W_5$, $OPIDS(V_G, V_S)$ computes $I_{DS,M8}$ in node (C8,38). For transistor M_6, $OPW(V_G, V_S)$ computes W_6 in node (C8,29). For transistor M_7, $OPW(V_G, V_S)$ computes W_7 in node (C8,23).

The pseudo-code of the design procedure is given in Fig.7. *[design_parameters]* are the parameters fixed by the designer (Table II). We first declare the two-transistors netlist and the name of N and P transistors in this netlist. Then we start the simulator and load this netlist. Operators then compute lists of all unknown parameters. To illustrate electrical parameters dependency we have for instance $OPVG(V_{eg})$ (lines 23, 24) that uses $V_{S,DP}$ that is previously computed by $OPVS(V_{eg}, V_B)$ (lines 19, 20).

We highlight that the same design procedure is used for various technologies (characterized among other parameters by L_{min}, V_{dd}, V_{th}) and transistor models (BSIM3V3, BSIM4, PSP).

B. BSIM3V3 Model (130 nm)

We use the input parameters listed in Table II. As we can see in Table III, simulator finds same I_{DS} and V_{eg} (compared to those fixed as input parameters) with widths computed by CHAMS. All transistors operate in saturation region, that is especially crucial for $M5$ that acts as a current source. Moreover values computed by CHAMS in Table III are the same that those computed by CAIRO+ in [15] for the same circuit and input parameters.

C. BSIM4 Model (65 nm)

We use the same input parameters (first column in Table II). Gain and transition frequency are reduced (Table VII) and widths are twice smaller compared to 130nm technology. Gain curves of the amplifier are plotted in Fig.8.

TABLE IV

CHAMS VERSUS SIMULATION RESULTS (BSIM4 65NM).

Parameter		M_1, M_2	M_3, M_4	M_5	M_6
$I_{DS}\ (\mu A)$	CHAMS	15.0	-15.0	30.0	-150.0
$I_{DS}\ (\mu A)$	Simulation	15.013	-14.977	29.998	-150.009
$V_{eg}\ (V)$	CHAMS	0.12	-0.12	0.1	not given
$V_{eg}\ (V)$	Simulation	0.12	-0.119	0.099	-0.119
$V_{GS}\ (V)$		0.441	-0.307	0.430	-0.307
$V_{DS}\ (V)$		0.733	-0.307	0.158	-0.6
$V_{BS}\ (V)$		-0.158	0.0	0.0	0.0
$V_{th}\ (V)$		0.321	-0.187	0.330	-0.187
$V_{dsat}\ (V)$		0.150	-0.123	0.137	-0.123
$g_m\ (mA/V)$		0.146	0.178	0.328	1.772
$g_{ds}\ (\mu A/V)$		2.444	7.563	35.847	39.398
$C_{gd}\ (fF)$		0.208	1.086	1.166	8.479
$W\ (\mu m)/M$	CHAMS	0.768/4	3.889/2	2.536/4	35.589/20

TABLE V

CHAMS VERSUS SIMULATION RESULTS (BSIM4 45NM).

Parameter		M_1, M_2	M_3, M_4	M_5	M_6
$I_{DS}\ (\mu A)$	CHAMS	12.5	-12.5	25.0	-125.0
$I_{DS}\ (\mu A)$	Simulation	12.495	-12.495	24.999	-125.001
$V_{eg}\ (V)$	CHAMS	0.05	-0.1	0.08	not given
$V_{eg}\ (V)$	Simulation	0.05	-0.099	0.08	-0.1
$V_{GS}\ (V)$		0.365	-0.414	0.398	-0.414
$V_{DS}\ (V)$		0.445	-0.414	0.130	-0.495
$V_{BS}\ (V)$		-0.13	0.0	0.0	0.0
$V_{th}\ (V)$		0.315	-0.314	0.318	-0.313
$V_{dsat}\ (V)$		0.115	-0.112	0.128	-0.112
$g_m\ (mA/V)$		0.153	0.140	0.245	1.397
$g_{ds}\ (\mu A/V)$		3.524	7.952	66.589	66.61
$C_{gd}\ (fF)$		0.567	0.389	1.611	3.661
$W\ (\mu m)/M$	CHAMS	1.354/4	1.029/2	1.546/4	9.750/20

TABLE VI

CHAMS VERSUS SIMULATION RESULTS (PSP 45NM).

Parameter		M_1, M_2	M_3, M_4	M_5	M_6
$I_{DS}\ (\mu A)$	CHAMS	12.5	-12.5	25.0	-125.0
$I_{DS}\ (\mu A)$	Simulation	12.499	-12.499	24.999	-125.0
$V_{eg}\ (V)$	CHAMS	0.05	-0.12	0.08	not given
$V_{eg}\ (V)$	Simulation	0.049	-0.119	0.08	-0.12
$V_{GS}\ (V)$		0.399	-0.513	0.421	-0.513
$V_{DS}\ (V)$		0.436	-0.513	0.151	-0.55
$V_{BS}\ (V)$		-0.151	0.0	0.0	0.0
$V_{th}\ (V)$		0.349	-0.393	0.341	-0.392
$V_{dsat}\ (V)$		0.13	-0.163	0.143	-0.163
$g_m\ (mA/V)$		0.242	0.162	0.407	1.616
$g_{ds}\ (\mu A/V)$		4.136	6.337	25.3	60.062
$C_{gd}\ (fF)$		0.03	0.007	0.232	0.051
$W\ (\mu m)/M$	CHAMS	5.64/4	2.748/2	4.971/4	26.952/20

D. BSIM4 Model (45 nm)

All input parameters (second column in Table II) have been lowered compared to previous values. $V_{eg,M1}$ is strongly reduced in order to cope with V_{DD} decrease.

E. PSP Model (45 nm)

We observe significant differences in widths (Tables V and VI) and performances (Table VIII) between these last two transistor models. This is due to the differences in technology parameters between BSIM4 and PSP 45nm models. Indeed the MOS transistor modeled with BSIM4 has a thinner gate oxyde and therefore lower V_{DD}, than the MOS transistor modeled with PSP.

V. Conclusion

In this paper, we presented a new simulation-based tool for analog integrated circuits design using sizing and biasing operators. The tool implements simulator encapsulation, allowing easy use of different transistor models. Sizing and biasing was successfully performed in less than one minute on a single-ended two-stage operational amplifier using a unique design procedure with BSIM3V3 (130nm), BSIM4 (65nm and 45nm) and PSP (45nm) transistor models.

TABLE VII
Performances for BSIM3V3 130nm and BSIM4 65nm.

Parameter		BSIM3V3(130nm)	BSIM4(65nm)
A_{d0} (dB)	CHAMS (eq.(5))	65.67	50.91
A_{d0} (dB)	Simulation	65.61	49.99
F_t (MHz)	Simulation	10.04	7.42
Phase Margin (degrees)	Simulation	77.90	80.93
Slew Rate (V/μs)	CHAMS (eq.(6))	10.3	10.3
Slew Rate (V/μs)	Simulation	11.9	11.8

TABLE VIII
Performances for BSIM4 45nm and PSP 45nm.

Parameter		BSIM4 (45nm)	PSP (45nm)
A_{d0} (dB)	CHAMS (eq.(5))	45.06	51.1
A_{d0} (dB)	Simulation	44.77	50.93
F_t (MHz)	Simulation	7.42	11.94
Phase Margin (degrees)	Simulation	79.4	74.7
Slew Rate (V/μs)	CHAMS (eq.(6))	8.6	8.6
Slew Rate (V/μs)	Simulation	10.1	10.2

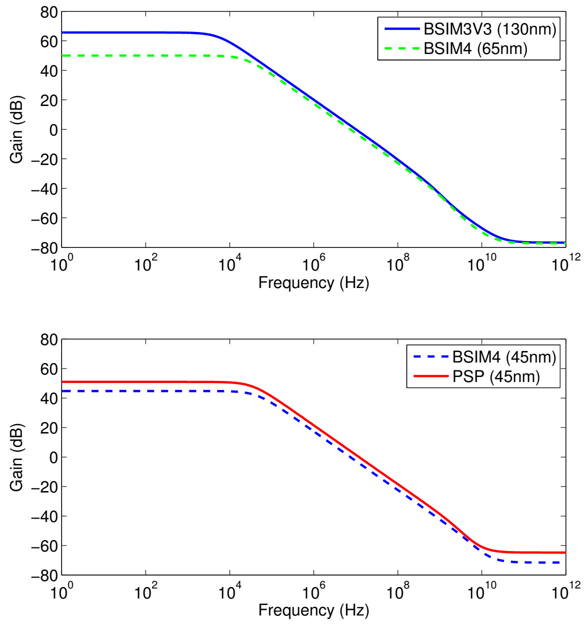

Fig. 8. Comparison of amplifier gain for different technologies and transistor models.

References

[1] S. Hammouda, H. Said, M. Dessouky, M. Tawfik, Q. Nguyen, W. Badawy, H. Abbas, and H. Shahein. "Chameleon ART: a non-optimization based analog design migration framework". *Design Automation Conference*, pages 885–888, July 2006.

[2] F. Leyn, G. Gielen, and W. Sansen. "An efficient DC root solving algorithm with guaranteed convergence for analog integrated CMOS circuits". *International Conference on Computer-Aided Design*, pages 304–307, November 1998.

[3] M. Krasnicki, R. Phelps, R. A. Rutenbar, and L. R. Carley. "MAELSTROM: Efficient Simulation-based Synthesis for Custom Analog Cells". *Proc. of Design Automation Conference*, pages 945–950, June 1999.

[4] R. Phelps, M. Krasnicki, R. A. Rutenbar, and L. R. Carley. "ANACONDA: Simulation-based Synthesis of Analog Circuits Via Stochastic Pattern Search". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 703–717, June 2000.

[5] R. Iskander, M. Dessouky, M. Aly, M. Magdy, N. Hassan, N. Soliman, and S. Moussa. "Synthesis of CMOS Analog Cells Using AMIGO". *Design, Automation and Test in Europe*, pages 297–302, 2003.

[6] R. Harjani, R.A. Rutenbar, and L.R. Carley. "OASYS: A Framework for Analog Circuit Synthesis". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1247–1266, December 1989.

[7] K.K Wee and R.J. Mack. "Towards expandable and generalised analogue design automation". *International Symposium on Circuits and Systems*, pages 359–362, June 1994.

[8] K. Swings, G. Gielen, and W. Sansen. "An intelligent analog IC design system based on manipulation of design equations". *Custom Integrated Circuits Conference*, May 1990.

[9] R. Iskander, D. Galayko, M-M. Louërat, and Andreas Kaiser. "Knowledge-Aware Synthesis Using Hierarchical Graph-Based Sizing and Biasing". *IEEE International Midwest Symposium on Circuits and Systems*, pages 984–987, August 2007.

[10] J. Porte. OCEANE, *http://www-asim.lip6.fr/recherche/oceane*.

[11] B.A.A. Antao and A.J. Brodersen. "Techniques for synthesis of analog integrated circuits". *Design and Test of Computers*, pages 8–18, March 1992.

[12] R.A. Rutenbar, G.G.E. Gielen, and J. Roychowdhury. "Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs". *Proceedings of the IEEE*, vol. 95(No. 3):640–669, March 2007.

[13] R. Iskander, M-M. Louërat, and A. Kaiser. "Automatic DC Operating Point Computation and Design Plan Generation for Analog IPs". *Analog Integrated Circuits and Signal Processing Journal*, vol. 56:93–105, August 2008.

[14] R. Iskander, M-M. Louërat, and A. Kaiser. "Hierarchical Graph-Based Sizing for Analog Cells Through Reference Transistors". *Ph.D. Research in MicroElectronics and Electronics*, pages 321–324, July 2006.

[15] Ramy Iskander. *"Knowledge-aware synthesis for analog integrated circuit design and reuse"*. PhD thesis, University of Paris 6, France, July 2008.

[16] C. Alexandre, H. Clement, J-P. Chaput, M. Sroka, C. Masson, and R. Escassut. "TSUNAMI: An Integrated Timing-Driven Place And Route Research Platform". *Proceedings of the 2005 Design, Automation, and Test in Europe*, pages 920–921, 2005.

[17] EXPECT, *http://expect.nist.gov/*.