DESIGN SPACE EXPLORATION FOR ANALOG IPS USING CAIRO+

Ramy ISKANDER, Laurent de LAMARRE*, Andreas KAISER**, Marie-Minerve LOUËRAT**

**Université Pierre et Marie Curie, LIP6-ASIM Laboratory,4 Place Jussieu, 75252 Paris, France*
***IEMN-ISEN, 41 Bld. Vauban, 59046 Lille Cedex, France*
email: ramy.iskander@asim.lip6.fr

***Abstract*—*In this paper, a new methodology for design space exploration (DSE) and knowledge capture is presented. The methodology presents analog IPs as a hierarchy of devices and modules. The hierarchy is implemented using an analog language called CAIRO+. A mechanism, which is established between devices and modules, allows modules to question devices about their electrical behavior. The electrical properties propagate upward along the hierarchy in order to guide higher level modules. This methodology is used to capture and document the designer's knowledge. Once captured, an executable generator is produced. This generator is then used to synthesize analog IPs with different specifications. The same generator can be used to migrate analog IPs between different technologies. As an illustrative example, a fully-differential two stages common-mode feedback (CMFB) amplifier is synthesized and migrated using CAIRO+.*

I. INTRODUCTION

During the past decade, as the design process of the analog part of a mixed-signal chip became a bottleneck, researchers became interested in the synthesis of analog cells. Despite the big efforts to develop CAD tools to automate the design process of analog cells, analog designers are still reluctant to use them. Their main argument is the unexplained compromises and tradeoffs demonstrated in those tools, as well as their incompleteness. Analog designers prefer to perform complicated hand analysis rather than using automated tools. This way, they control all the electrical and physical aspects of the design, based on their accumulated experience. Capturing the designer's knowledge and integrating them into CAD tools have been an active research area for many years. OASYS [1] uses a hierarchical decomposition of the cell topology and executes design plans to select each level's topology and sizes the corresponding devices to meet the performance specifications. BLADES [2] tries to mimic designer's knowledge in an expert system in the form of "IF-ELSE" rules. Nowadays, EDA companies, like Barcelona Design [3], developed CAD tools that allows designers to specify the equations of analog circuits, at the system, block and device levels - solving them simultaneously to arrive at optimal device sizes that satisfy performance and yield specifications.

A main difficulty is related to the inherent nature of the analog world. This is mainly due to the lack of parameterized cell generators. As we know, the notion of soft IPs exists in the digital domain. The same IP will be reused many times in many different designs through synthesis. The existence of complete synthesis flow from VHDL down to silicon allows technology migration to be performed systematically. In the analog world, such a flow does not exist. Due to the continuous nature of the analog signals, the devices have to be manually resized to meet the specifications in the new technologies. Thus, the full-custom approach is the regular method for analog design.

In this paper, we follow the similar ideas to OASYS and BLADES. In section A, CAIRO+ framework that is used to capture the designer's knowledge is described. Then, in section B, a fully-differential two-stages CMFB amplifier is presented. The representation of the amplifier into CAIRO+ is demonstrated. Once the DSE phase is completed, a parameterized cell generator is systematically created. The produced generator will be used for two purposes. The first one is to synthesize the amplifier for predetermined specifications in 0.6μ technology. The second purpose is to migrate the amplifier to 0.35μ technology. Finally, we draw conclusions in section C.

II. CAIRO+ FRAMEWORK

The CAIRO+ framework is described in [4].The main idea is to capture the designer's knowledge into an executable generator that is capable to produce correct device sizing for a set of predefined specifications. CAIRO+ provides the designer with an analog language to describe generators hierarchically. It is built as an extension to C/C++ programming language. It provides a library of functions to describe the *netlist template*, *layout template*, *specification template*, *design space exploration procedure*, *shape generation* and *layout generation*. In our work, we will focus on the design space exploration procedure. The following subsections describe some notions of CAIRO+.

A. Modules and Devices

Any analog IP core may be hierarchically composed of modules and devices. Higher level modules can communicate with lower level modules as shown in Figure 1. The lowest level module communicates with the device level. The device level describes passive devices: resistors and capacitors. It also describes active devices: PMOS, NMOS, differential pair, simple current mirror, ... etc. To calculate all the small signal parameters for the leaf

transistors, the device level calls the builtin BSIM3V3 transistor model to calculate the small signal parameters.

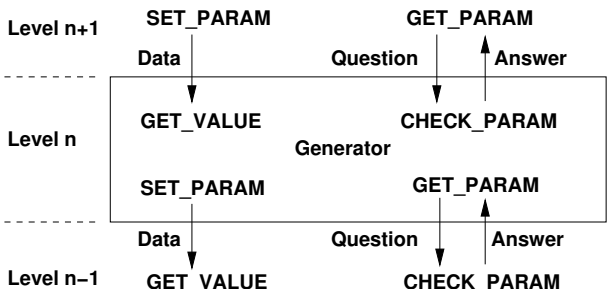

Figure 1: Communication protocol between higher-level and lower-level modules.

The higher level module $n+1$ uses SET_PARAM to set all the parameters needed to answer a question. The lower level module n (or device level) has access to these parameters through calls to GET_VALUE. Then, the higher level module calls the GET_PARAM to get the answer. The answer for the question is implemented in the lower level module in a $CHECK\_PARAM$ section. Normally, the question is the specification which the designer is interested in. The answer is given by the procedure used to calculate the specification. A question may have different answer procedures to calculate it.

B. Design Space Exploration

To write a parameterized generator, the designer must first determine the set of devices required by the given circuit topology. Then the modules are implemented hierarchically on top of the device level. The device level must provide questions/answers for the higher level module. Each lower level module must also provide another set of questions/answers to the next higher level module, and so on. We can think that each module level is a different level of abstraction in the design representation. As an example, one can define a system level($\Sigma\Delta$), a functional level(integrator), a block level(amplifier) and a device level. After determining the devices, the designer needs to identify a design plan to size all the transistors in the IP analog. The whole generator is coded using C/C++ functions that constitutes CAIRO+ framework.

III. Analog IP: Amplifier

In this section, we demonstrate the use of CAIRO+ framework to create a parameterized generator for a fully differential two-stages CMFB amplifier[5]. The amplifier is shown in Figure 2. The details of the implementation will be discussed in the following subsections.

A. Circuit Analysis

A thorough hand analysis of the small signal model of the amplifier has been developed. The analysis takes into account parasitic capacitances in the transistor model. The transistor model used is shown in Figure 3.

The precision of the developed model was verified against the ELDO simulation. It was found that the model satisfies the frequency response of amplifier in terms of static gain, poles, zeros, phase margin and high frequency gain. The model was verified

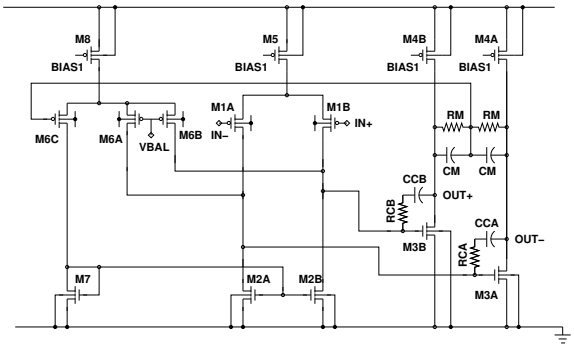

Figure 2: Fully-differential 2-stages CMFB amplifier.

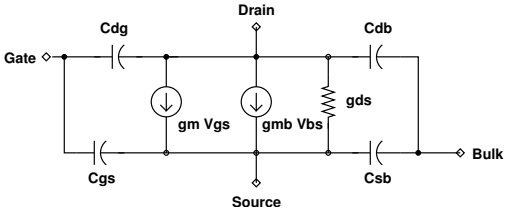

Figure 3: Small-signal transistor model including parasitics.

using a symbolic analyzer to ensure the correctness of the derivation steps. The following equations fully describe the amplifier:

$$G_{01} = gds_{1B} + gds_{2B} + gds_{6A} \quad (1)$$
$$C_{02} = C_{db1B} + C_{db2B} + C_{gs3B} + C_{db6A} + C_{dg6A} + C_{dg2B} \quad (2)$$
$$G_{03} = gds_{3B} + gds_{4B} + G_L + G_M \quad (3)$$
$$C_{03} = C_{db3B} + C_{db4B} + C_L + C_M + C_{dg4B} \quad (4)$$

$$A = gm_{3B} + G_{01} + G_{03} \quad (5)$$
$$B = C_{dg1B} + C_{02} + C_{03} \quad (6)$$
$$K = C_{dg1B}G_{03} + C_{02}G_{03} + C_{03}G_{01} \quad (7)$$
$$D = G_{01}G_{03} \quad (8)$$
$$E = (C_{dg1B} + C_{02})C_{03} \quad (9)$$
$$C_T = C_{dg3B} + C_{CB} \quad (10)$$

$$A_2 = R_{CB}C_{CB}C_{dg3B} \quad (11)$$
$$A_1 = C_T - gm_{3B}R_{CB}C_{CB} \quad (12)$$

$$B_3 = R_{CB}C_{CB}(E + C_{dg3B}B) \quad (13)$$
$$B_2 = E + R_{CB}C_{CB}C_{dg3B}A + BC_T + KR_{CB}C_{CB} \quad (14)$$
$$B_1 = K + DR_{CB}C_{CB} + AC_T \quad (15)$$
$$(16)$$
$$A_d(s) = (C_{dg1B}s - gm_{1B}) \cdot \frac{[A_2s^2 + A_1s - gm_{3B}]}{[B_3s^3 + B_2s^2 + B_1s + D]} \quad (17)$$

Equations $(1) - (17)$ have been implemented in the generator to calculate the amplifier performance.

B. Modules and Devices

In order to determine the set of devices that should be implemented in the device level, we rely on the matching criterion. Any group of elementary components that are equally sized have to be matched in the layout level. This is done to reduce the effects of process variations on the overall performance. The implemented devices are shown in Figure 4. We use the appropriate devices to form each of following groups:(M_5, M_8), (M_{1A}, M_{1B}), (M_{3A}, M_{3B}), (M_{4A}, M_{4B}), (M_7, M_{2A}, M_{2B}), (M_{6C}, M_{6A}, M_{6B}), (R_{CA}, R_{CB}), (C_{CA}, C_{CB}), (R_M, R_M) and (C_M, C_M).

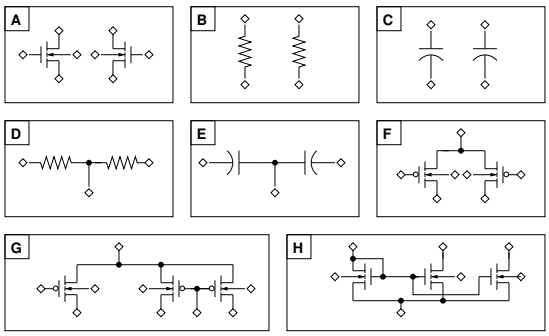

Figure 4: Low-level devices.

C. Implementation using CAIRO+

We model the amplifier, as the current module level, based on the devices already defined. In the current module, we define a design space exploration procedure that will size all the devices of the amplifier. We create one $CHECK\_PARAM$ section called $DIMENSIONS$, that when called from higher level module, will perform the sizing procedure. As an example, to perform the sizing, we choose to fix the bias currents in the two stages of the amplifier. We implement the following algorithm:

```
Set values of passive devices
Determine all drain currents in all transistors
Fix input voltages for operating point
Estimate initial value of gate voltage VGS of M1A
Loop while gate voltage VGS of M1A not converging

    Calculate W of M5 and M8 based on gate voltage VGS of M1A
    Get new estimated gate voltage VGS of M5 and M8
    Calculate W of M4A and M4B based on gate voltage VGS of M5
    Calculate W of M3A and M3B
    Get new estimated gate voltage VGS of M3A and M3B
    Calculate W of M1A and M1B based on voltages on M5 and M3B
    Calculate W of M6C, M6A, M6B in terms of M1A
    Get new estimated gate voltage VGS of M1A


End Loop
Calculate W of M7, M2A and M2B based on voltages on M3B and M1A
```

The code is written in C/C++ using CAIRO+ framework. It is then compiled into an executable generator by the framework.

```
CAIRO_CHECK_PARAM("DIMENSIONS")

    ...

    // ------------------------------------
    // Initialization of passive devices
    // ------------------------------------
    CAIRO_SET_PARAM( LIBRES_4C, "BIRES_4C_POLY", "res_rca_rcb" , "RVAL", rc) ;
    CAIRO_SET_PARAM( LIBCAP_4C, "BICAPA_4C"    , "cap_cca_ccb" , "CVAL", cc) ;

    CAIRO_SET_PARAM( LIBRES_3C, "BIRES_3C_POLY", "res_rma_rmb" , "RVAL", rm) ;
    CAIRO_SET_PARAM( LIBCAP_3C, "BICAPA_3C"    , "cap_cma_cmb" , "CVAL", cm) ;

    // ------------------------------------------------
    // Determine all drain currents in all transistors
    // ------------------------------------------------
    isd_m5  = ibias            ; isd_m8  = ibias            ;
    isd_m1a = ibias/2.0        ; isd_m1b = ibias/2.0        ;
    isd_m6c = ibias/2.0        ; isd_m6a = ibias/4.0        ; isd_m6b = ibias/4.0;
    ids_m2a = isd_m1a + isd_m6a; ids_m2b = isd_m1b + isd_m6b;
    isd_m4a = igain * ibias    ; isd_m4b = igain * ibias    ;
    ids_m3a = isd_m4a          ; ids_m3b = isd_m4b          ;
    ids_m7  = isd_m6c          ;

    // ------------------
    // Fix input voltages
    // ------------------
    v_bal = vdd/2.0;
    v_inp = vdd/2.0;
    v_inn = vdd/2.0;
```

```
    // ------------------------------------------------
    // Estimate initial value of gate voltage of M1A
    // ------------------------------------------------
    vgs_m1a = veg_p ;

    do {

        vgs_m1a_prev = vgs_m1a ;

        // ----------------------------
        // Calculate W of M5 and M8
        // ----------------------------
        vd_m5   = vdd/2.0 - vgs_m1a;
        vsd_m5  = vdd - vd_m5;
        vbs_m5 = 0.0;

        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "TEMP",temp) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "VDS", -vsd_m5) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "VBS", vbs_m5) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "IDS", -isd_m5) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "L"  , l) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "VEG",veg_p) ;

        TRY
          CAIRO_GET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "W","W(L,IDS,VEG)",w_m5);
           CAIRO_GET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m8_m5", "VGS","*",vgs_m5);
        IF_ERROR_PARAM
              exit(-1);
        ENDIF_ERROR_PARAM

        // ------------------------------
        // Calculate W of M4A and M4B
        // ------------------------------
        v_bias = vdd + vgs_m5;

        vgs_m4b = vgs_m5;
        vsd_m4b = vdd/2;
        vbs_m4b = 0.0;

        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m4b_m4a", "TEMP",temp) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m4b_m4a", "VDS", -vsd_m4b) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m4b_m4a", "VBS", vbs_m4b) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m4b_m4a", "IDS", -isd_m4b) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m4b_m4a", "L"  , l) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m4b_m4a", "VGS",vgs_m4b) ;

        TRY
           CAIRO_GET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m4b_m4a", "W","W(L,IDS,VGS)",w_m4b);
        IF_ERROR_PARAM
            exit(-1);
        ENDIF_ERROR_PARAM

        // ------------------------------
        // Calculate W of M3A and M3B
        // ------------------------------
        vds_m3b = vdd/2.0;
        vbs_m3b = 0.0;

        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "TEMP",temp) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "VDS", vds_m3b) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "VBS", vbs_m3b) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "IDS", ids_m3b) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "L"  , l) ;
        CAIRO_SET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "VEG",veg_n) ;

        TRY
          CAIRO_GET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "W","W(L,IDS,VEG)",w_m3b);
           CAIRO_GET_PARAM( LIBMOS_2T, "MOS_2T", "tr_m3b_m3a", "VGS","*",vgs_m3b);
        IF_ERROR_PARAM
           exit(-1);
        ENDIF_ERROR_PARAM

        // ------------------------------
        // Calculate W of M1A et M1B
        // ------------------------------
        v_n8d   = vdd - vsd_m5;
        vsd_m1a = v_n8d - vgs_m3b;

        if (bulk)
           vbs_m1a = vdd - v_n8d;
        else
            vbs_m1a = 0.0;

        CAIRO_SET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "TEMP",temp);
        CAIRO_SET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "VBS",vbs_m1a);
        CAIRO_SET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "VDS",-vsd_m1a);
        CAIRO_SET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "L",l);
        CAIRO_SET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "IBIAS",-isd_m5);
        CAIRO_SET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "VEG",veg_p);

        TRY
           CAIRO_GET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "W","W(L,IBIAS,VEG)",w_m1a);
          CAIRO_GET_PARAM( LIBMOS, "DP_CC", "pd_m1a_m1b", "VGS","*",vgs_m1a);
           w_m6c = w_m1a;
           w_m6a = w_m1a/2;
           w_m6b = w_m1a/2;
        IF_ERROR_PARAM
            exit(-1);
        ENDIF_ERROR_PARAM

    } while (fabs(vgs_m1a-vgs_m1a_prev) > EPSILON);

    CAIRO_SET_PARAM( LIBPD_3T, "PD_3T", "pd_m6c_m6a_m6b", "WREF",w_m6c);
    CAIRO_SET_PARAM( LIBPD_3T, "PD_3T", "pd_m6c_m6a_m6b", "LREF",l);

    // ----------------------------------
    // Calculate W of M7, M2A and M2B
    // ----------------------------------
    vds_m7 = vgs_m3b ;
    vds_m2a= v_n8d - vsd_m1a;
    vbs_m7 = 0.0;

    a = 2;
    b = 3;
```

```
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "TEMP",temp) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "VBS", vbs_m7) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "IREF",ids_m7 ) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "LREF", l) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "A",a) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "B",b) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "VDSREF",vds_m7) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "VDS",vds_m2a) ;
      CAIRO_SET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "VGS",vds_m2a) ;

      TRY
         CAIRO_GET_PARAM( LIBCM_3T, "CM_3T", "cm_m7_m2a_m2b", "WREF","*",w_m2a);
         w_m2b = w_m2a;
         w_m7  = w_m2a * a / b;
      IF_ERROR_PARAM
         exit(-1);
      ENDIF_ERROR_PARAM

      // ----------------------------
      // Flag : Dimensions ready
      // ----------------------------
      CAIRO_SET_LOCAL_VARIABLE("DIMENSIONS_EFFECTUES",true);
      CAIRO_SET_LOCAL_VARIABLE("W_M1A",w_m1a);

      CAIRO_RETURN_PARAM(1.0);

END_CHECK_PARAM
```

D. Synthesis Results

To synthesize the amplifier in the 0.6μ, we set the specifications as $I_{1ststage} = 20\mu A$, $I_{2ndstage} = 200\mu A$ and $V_{eg} = V_{gs} - V_{th} = 0.4V$. We also fix the values: $R_M = 40K\Omega$, $C_M = 1pf$, $R_C = 370\Omega$, $C_C = 2pf$, $V_{dd} = 5V$, $L = 2\mu m$. After synthesis, we get the dimensions for the transistors as shown in Table 1.

Table 1: Dimensions in 0.6μ

Transistor	Dimensions
M_{1A}, M_{1B}, M_{6C}	$7.68\mu/2\mu$
M_{3A}, M_{3B}	$47.65\mu/2\mu$
M_{4A}, M_{4B}	$138.50\mu/2\mu$
M_{6A}, M_{6B}	$3.84\mu/2\mu$
M_7	$2.03\mu/2\mu$
M_{2A}, M_{2B}	$3.04\mu/2\mu$
M_5, M_8	$16\mu/2\mu$

The simulation results show that the amplifier behaves as predicted by equations $(1) - (17)$. This is shown in Figure 5.

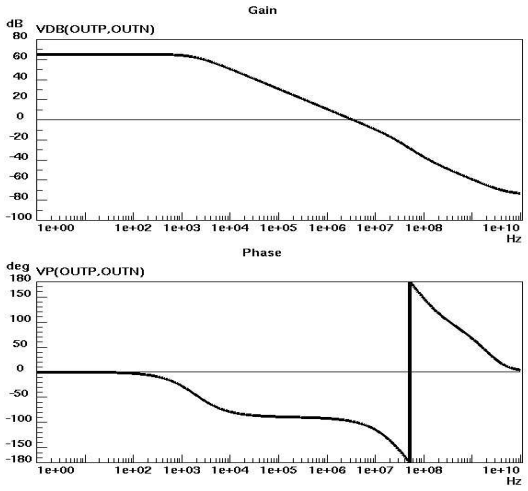

Figure 5: Simulation results for 0.6μ.

E. Technology Migration Results

To migrate the amplifier in the 0.35μ, we set the specifications as $I_{1ststage} = 20\mu A$, $I_{2ndstage} = 200\mu A$ and $V_{eg} = V_{gs} - V_{th} = 0.2V$. We also fix the values: $R_M = 40K\Omega$, $C_M = 1pf$, $R_C = 370\Omega$, $C_C = 2pf$, $V_{dd} = 3.3V$, $L = 2\mu m$. After synthesis, we get the dimensions for the transistors as shown in Table 2,

Table 2: Dimensions in 0.35μ

Transistor	Dimensions
M_{1A}, M_{1B}, M_{6C}	$19.65\mu/2\mu$
M_{3A}, M_{3B}	$143\mu/2\mu$
M_{4A}, M_{4B}	$406.5\mu/2\mu$
M_{6A}, M_{6B}	$9.8\mu/2\mu$
M_7	$4.8\mu/2\mu$
M_{2A}, M_{2B}	$7.2\mu/2\mu$
M_5, M_8	$42\mu/2\mu$

The simulation results show that the amplifier behaves as predicted by equations $(1) - (17)$. This is shown in Figure 6.

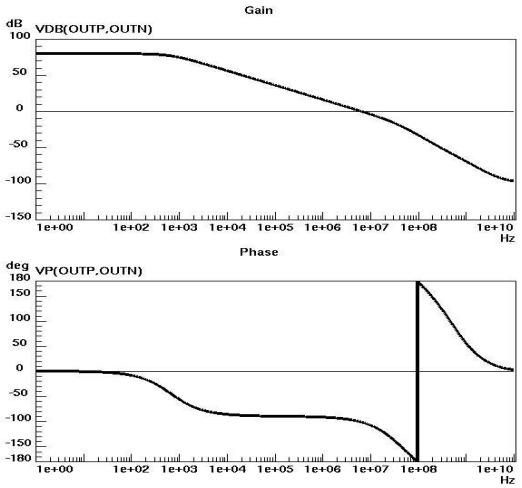

Figure 6: Simulation results for 0.35μ.

IV. Conclusions

CAIRO+ framework was used to develop an executable parameterized generator for a fully-differential two-stages CMFB amplifier. The generator was then used to synthesize the amplifier in the 0.6μ. Then, the amplifier was migrated to the 0.35μ. In both tests, the results were verified through ELDO simulations that use the device sizes generated by the parameterized generator. The framework showed to be successful in synthesizing and migrating analog IPs.

V. References

[1] R.Harjani, R.A.Rutenbar and L.R.Carley. *OASYS: a framework for analog circuit synthesis*. IEEE Trans. on Computer–Aided Design, vol. 8, no. 12, pp. 1247–1266, Dec. 1989.

[2] Fatehy El-Turky and Elizabeth E. Perry. *BLADES: An artificial intelligence approach to analog circuit design*. IEEE Trans. on Computer–Aided Design, vol. 8, no. 6, pp. 680–692, 1989.

[3] Website URL: *www.barcelonadesign.com*

[4] P. Nguyen Tuong, M.M.Louërat and A. Greiner. *Guidelines for Designing Smart and Reusable Analog IP Cores*. SAME Sophia Antipolis Microeletronics Forum, Oct. 2004.

[5] Mihau Banu, John H. Khoury and Yannis Tsividis. *Fully Differential Operational Amplifiers with Accurate Output Balancing*. IEEE Journal of Solid-State circuits, vol. 23, no. 6, pp. 1410–1414, Dec. 1988.