AIDA-C: Analog IC Design Optimization Embedding NGSpice as the Evaluation Engine

Catarina Aleixo
catarina.aleixo@tecnico.ulisboa.pt

***Abstract*— Two distinct approaches can be followed when automatizing analog IC sizing: knowledge-based or optimization-based. The latter is subdivided into two categories: equation-based and simulation-based. Simulation-based sizing uses a circuit simulator as evaluation engine, which is the case of AIDA-C, an Analog Integrated Circuit Design Automation environment. The work developed throughout this paper aims at integrating an open source circuits simulator, namely the NGSpice, into AIDA, which currently uses ELDO as its simulation engine. Multiple netlists were run in NGSpice in order to check which simulation commands should be invoked, to guarantee that the corresponding outputs match with ELDO's outputs. Integrating NGSpice into AIDA required the development of an interface, since the latter has a specific input format which is not compatible with NGSpice's output. This interface is written in Python and is able to run multiple netlists in parallel. The parallelism allows to increase the interface performance, which had to overcome NGSpice limitations, particularly when running corners. The results obtained were analysed statistically, where 95% confidence intervals for the relative errors of each performance measure were obtained, proving that NGSpice can be AIDA's simulation engine. AIDA's outputs always differ among optimizations, even using the same simulator, due to the algorithm used, which is based on a stochastic process. Taking this into account, to fairly compare both simulators the points simulated in ELDO were then simulated in NGSpice, and vice-versa.**

***Keywords*—Electronic Design Automation, Computer Aided Design, Sizing Optimization, Open source circuits simulator, Corners.**

I. INTRODUCTION

In the context of this paper, EDA is a category of products and processes to support analog IC design at circuit-level. This concept emerged in the 70's and revolutionized the way how engineers design products. Nowadays, integrated circuits (ICs) are mainly implemented through digital circuits, although radio-frequency and analog circuits are still the basis for many interfaces and transceivers. Nevertheless, despite most of design stages of digital circuits being automated, the lack of CAD tools for electronic design automation of analog and radio frequency ICs contributes to error-prone designer's intervention, during the entire course of the design process, which impacts in the overall system of development. These divergences among digital and analog design are related to the fact that the latter is not as systematic as the former, since different designs produce different circuits, which entails that most of the designed analog blocks are not usually reused in other circuits. In addition, in analog design, the optimization is fully heuristic and requires a larger knowledge base from the designer. Once the tendency is mixing digital and analog systems, the lower reuse rate of analog circuit blocks increases the production period of developing mixed-signal integrated circuits. Commonly, an analog project may require weeks or months to be manually developed. Considering that market demands are increasing, automate the developing process is extremely important, using EDA tools.

There are two different approaches with respect to automate analog IC sizing, namely knowledge-based and optimization-based. Focusing on the early automation systems, the initial strategy applied did not use any optimization, on the contrary, the knowledge-based approaches try to systematize the design deriving a design plan through specific knowledge [1]. On the other hand, the optimization-based approach, presented in Figure 1, apply optimization techniques to automate IC sizing and they can be divided in two main different subclasses: equation-based or simulation-based, which is defined according to the method used to evaluate circuit's performance. Equation-based techniques use analytic design equations, while simulation-based use a circuit simulator. The latter benefits from a faster execution time, which makes it preferable since equations are expensive to maintain and the results are not so accurate.

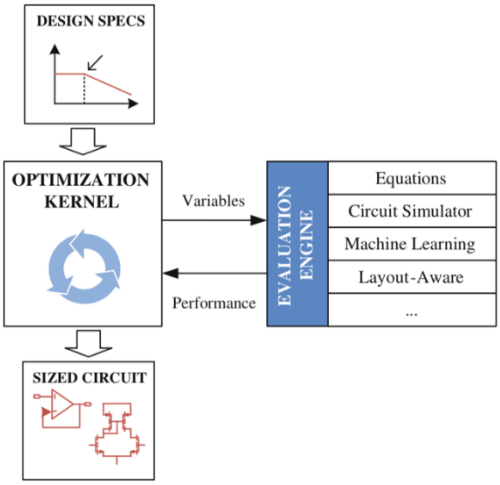

Fig. 1: Optimization-based Circuit Sizing [1]

Nowadays, there are plenty of evaluation engines, commercial and open-source. The main advantage of an open-source simulator is being able to offer flexibility and freedom of integration. Nevertheless, open source simulators also have

limitations, namely the lack of support and possible instability of versions. NGSpice [2] is being developed and maintained for more than fifteen years, which makes it popular among free spice-based simulators. This paper aims at integrating an open source simulator, namely, NGSpice simulator, in AIDA, in order to make it completely independent from any commercial circuit simulator and free for dissemination among the academic community.

The paper is organized as follows. In section II, some optimization-based EDA tools are described, and AIDA Framework is presented. In section III, the most used electrical simulators nowadays are identified, characterized and their features compared. In section IV, all NGSpice required features are evaluated in order to understand how this open-source simulator will be integrated in AIDA, comparing NGSpice with ELDO, which is AIDA's currently evaluation engine. In section V, the interface which integrates NGSpice in AIDA is presented and, in section VI, the results from two optimizations are analysed. Finally, in section VI, the conclusions are drawn.

II. Related Work and AIDA Framework

Optimization-based approaches are subdivided into two categories: equation-based and simulation-based. With the increase in computer power, simulation-based optimization became more advantageous. Simulation-based sizing uses a circuit simulator, e.g., SPICE [3] as evaluation engine, which is the case of AIDA-C [4].

A. Optimization-based EDA Tools

There are plenty algorithms that use optimization techniques, starting by the early approaches where local optimization around a "good" solution was used, in simulation-based automatic sizing, being SA [5] (simulated annealing) the most common optimization technique used. This technique is part of other optimization algorithms, such as FASY [6] and Kuo-Hsuan et al. [7], where an aproximate solution using equations is derived, using SA to improve the circuit design, or Cheng et al. [8], which follows a similar approach, using SA as a starting point and adding constraints to the problem through considering transistor bias conditions.

Genetic algorithms [9] are another class of optimization methods, which simulate the natural evolution of the current population, obtaining new solution vectors, after applying genetic operators of mutation and crossover. Barros et al. [10] [11] optimizes circuit sizing through genetic algorithms, using the circuit simulator and a support vector machine (SVM) to evaluate the populations. In Santos-Tavares [12], MAELSTROM [13] and ANACONDA [14], the parallel mechanism that allows to share the evaluation load in between multiple computers, reduces the time to simulate the population.

Other approach to optimize circuit sizing, that also uses evolutionary methods, consists of generating both the circuit topologies and the device sizes simultaneously. Koza [15], Sripramong [16] and Hongying [17] suggested a design procedure to generate new topologies, starting from low abstraction level. Although this is an innovative approach, designers are septic about it due to the generated structures being quite different from the well-known analog circuit structures.

In addition, swarm intelligence algorithms [18], which uses many simple agents to lead an intelligent global behaviour, are also methods to optimize analog circuit sizing. The most common ones are ant colony optimization (ACO), applied in [19] and particle swarm optimization (PSO) [20].

B. AIDA Framework

This thesis is related to AIDA, an Analog Integrated Circuit Design Automation environment, more properly on AIDA-C, an EDA tool for analog integrated circuit optimization at circuit-level, which uses state-of-the-art mutli-objective multi-constraint optimization techniques, including robust design requirements, since considers corner analysis, together with commercial simulators, such as ELDOTM, developed by Mentor Graphics, or HSPICE, from Synopsys.

The optimization outputs are a set of Pareto Optimal Fronts which are the non-dominated solutions, meaning a feasible solution, where there is not another feasible solution which is better than the present one in a certain objective function, without worsening another objective function.

AIDA-C was thought to be flexible enough to integrate any simulation tool. However, this interaction between a simulator and the optimizer is only possible defining an interface capable of writing files, executing commands and reading output files, which is named ACEI - AIDA-C Custom Evaluation Interface.

III. Evaluation engines

The main advantage of simulation engines is their ability to afford practical feedback when designing real circuits. These tools allow the user to explore the trade-offs among different designs without actually physically building the circuit. As a result, the inherent costs of developing a system decrease, reinforcing the powerfulness of these engines.

Firstly, all commercial simulators have a common feature: they are all based on SPICE (Simulated Program with Integrated Circuit Emphasis). However, SPICE has some limitations, and one of them is related to not considering the noise, interference and crosstalk, unless that behaviour is not added to the circuit. So, each simulation is as accurate as the behaviours modelled in the SPICE devices created for it. These simulators present consistent results and have plenty of support which is valuable to users. The main limitation is imposed by the number of licenses available and the costs associated.

Apart from the commercial simulators, there are also free evaluation engines, e.g., NGSpice, Gnucap [21], Qucs [22] or Xyce [23]. These engines have more limitations than the commercial simulators, mainly due to the lack of support provided. Apart from the manual and the online forums, there is no sources to search for any doubt, which sometimes inhibits its use. However, the time spent investigating how these simulators behave could be worth it, considering that there are no costs

associated with it and the unlimited number of licences available, which reinforces the powerfulness of these simulators and their capability of being integrated within other tools, which could be a great asset nowadays.

Looking at Table 1, is clear that all simulators have the same functionalities, except for NGSpice that does not support sweep analysis explicitly. Although there is no command to directly perform sweep analyses in NGSpice, there is a way of implementing it, using loops and variables inside the netlist. Despite that disadvantage of NGSpice, is the only simulator that is free, which makes it preferable in this particular case, since the main idea is turning AIDA completely independent of any simulation engine, in particular, independent of commercial ones. In addition, there are plenty of tools that use NGSpice for circuit simulation, which highlight even more its potential.

Table 1. Simulators Comparison

	Simulators			
	Eldo	Sepctre	HSpice	NGSpice
Developed By	Mentor	Cadence	Synopsys	Open source
Analog/Digital/Mixed Signal	Mixed Signal	Mixed Signal	Mixed Signal	Mixed Signal
Paid/Free	Paid	Paid	Paid	Free
Transient, DC and AC analyses	x	x	x	x
Monte Carlo	x	x	x	x
Multi-thread	x	x	x	x
Sweeping analysis	x	x	x	
Supports verilog	x	x	x	x
Noise, transfer function, sensitivity and pole-zero analyses	x	x	x	x

IV. NGSPICE FEATURES

Firstly, is important to understand how simulators interact with AIDA. The schema presented in Figure 2 shows, from a high perspective, how each simulator can be integrated in AIDA:

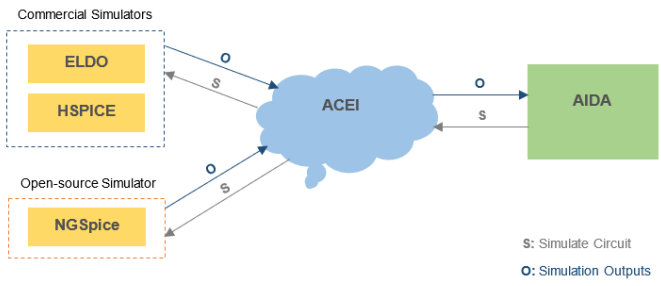

Fig. 2: Interaction between AIDA and simulators

Basically, the communication between AIDA and the simulators is performed through an interface, which not only executes simulations, but also collects the corresponding outputs which will then be delivered to AIDA.

A. *NGSpice Execution Modes*

NGSpice can be executed in two different ways: batch mode or interactive mode. Although, the interactive mode is more suitable when a user can interact with the simulator, this mode also allows to run a script, the netlist, without having to perform any interaction, using control sections along with the quit command. These sections have many advantages that will be highlighted further ahead. When using the batch mode, the behaviour of control sections used to get the measures needed is unpredictable, which inhibits its use when using these sections. In short, the command line that will be used to run circuits has the following structure:

ngspice netlist.cir > output.dat.

This command line saves the simulation standard output, in interactive mode, to the specified file.

B. *NGSpice and ELDO Netlists*

Despite the output values, given by NGSpice, being very similar to the ones given by ELDO, the syntax of the netlists run in the former are quite different from the ones in the latter, as shown in Figure 3, starting by how both the overdrives and margins of each transistor are obtained (marked with a blue dashed line). There are a set of parameters that have to be extracted from the OP analysis, more precisely, the voltage from gate to source (Vgs), the threshold voltage (Vth), the voltage from drain to source (Vds) and the voltage from drain to source, when the transistors enter the saturation region (Vdsat).

```
* ELDO Netlit

********** Analysis ***************************
.TEMP 25.0
.AC DEC 200 1 10G
.OPTION BRIEF=0
***********************************************

********** Performance Measures ***************
.EXTRACT DCAC label=IDD   ((-1(vdc))-(100u))
.EXTRACT AC   label=GDC   YVAL(VDB(OUTPUT),1)
.EXTRACT AC   label=GBW   XDOWN(VDB(OUTPUT),0,start,end)
.EXTRACT AC   label=PM    MIN(VP(OUTPUT),start,Extract(GBW))
.EXTRACT AC   label=FOM   (((GBW/1000000)*6)/(IDD*1000))
***********************************************

********** OVERDRIVES *************************
.MEASURE AC vov_mnm0    = param('ABS(VGS(xinova.mnm0)-lv9(xinova.mnm0))')
.MEASURE AC vov_mnm1    = param('ABS(VGS(xinova.mnm1)-lv9(xinova.mnm1))')
***********************************************

********** MARGINS ****************************
.MEASURE AC delta_mnm0    = param('ABS(VDS(xinova.mnm0)-VDSAT(xinova.mnm0))')
.MEASURE AC delta_mnm1    = param('ABS(VDS(xinova.mnm1)-VDSAT(xinova.mnm1))')
***********************************************
```

```
* NGSpice Netlist

********** Analysis ***************************
.TEMP 25.0
.OPTION BRIEF=0
***********************************************

.control

set units = degrees
AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

OP

let IDD = (-@vdc[i]) - 100u
print IDD

let vov_mnm0 = @m.xinova.mnm0[vgs] - @m.xinova.mnm0[vth]
print vov_mnm0

let vov_mnm1 = @m.xinova.mnm1[vgs] - @m.xinova.mnm1[vth]
print vov_mnm1

let delta_mnm0 = @m.xinova.mnm0[vds] - @m.xinova.mnm0[vdsat]
print delta_mnm0

let delta_mnm1 = @m.xinova.mnm1[vds] - @m.xinova.mnm1[vdsat]
print delta_mnm1

.endc
```

Fig. 3: Comparison between ELDO and NGSpice netlists

Using both simulators, inside the netlist, the user can use explicitly order to calculate both the overdrives and the margins, subtracting in each transistor Vgs - Vth and Vds - Vdsat, respectively. However, in NGSpice, both overdrives and margins have to be assigned to a variable that can be later printed to the standard output, while in ELDO with the .MEASURE command the same measures are immediately printed.

Another important difference is the command .EXTRACT used to get performance measures in ELDO simulator, which does not exist in NGSpice. To obtain these measures, NGSpice uses the .MEASURE command, differing here in the way these values are obtained. These differences are also highlighted in Figure 3, with a green dashed line.

Finally, in NGSpice, both the analyses and measures are enclosed in a control section. This section enables the use of interactive mode without the need of the user to interact with it.

C. *NGSpice Corners Simulation*

Corners are variations in the values of fabrication parameters, more properly, represent the extremes of these parameter variations, which corresponds to varying increments of environmental conditions, such as voltage, clock frequency or temperature, simultaneously or not. The main idea of simulating corners is to verify the behaviour of circuits, testing its limits.

NGSpice has a command, *altermod*, which allows the user to switch between model parameters. However, the libraries used cannot have the same structure as in ELDO, since corners are not obtained in the same way. In Figure 4, the differences between both simulators are illustrated:

```
* ELDO Example

.TEMP 25.0
.AC DEC 200 1 10G
.OPTION BRIEF=0

.ALTER SS
.lib '/umc_013/models/L130E_MM_TWINWELL_V182/L130E_HGIO_RVT33_V131/L130E_HG_RVT33_V131.lib.eldo' SS
```

```
* NGSpice Example

.control

set units = degrees

AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0


altermod N_HG_33_L130E P_HG_33_L130E file /umc_013/models/L130E_MM_TWINWELL_V182/L130E_HGIO_RVT33_V131/L130E_HG_RVT33_V131.mdl.eldo


AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

quit

.endc
```

Fig. 4: Differences between running corners in ELDO and in NGSpice

Looking at both netlists, the differences are notorious, starting by the syntax. ELDO uses two commands, ALTER and .lib commands to run a corner where a library is changed, while in NGSpice only the altermod is invoked followed by the path to the corresponding library.

However, NGSpice also supports the commands used in ELDO. Nevertheless, as demonstrated in Figure 5 and Figure 6, the .lib command overrides the typical library declared at the beginning of the netlist.

Looking at the Figure 5 and Figure 6, the values highlighted with a blue dashed line prove that the typical library is being overridden by the corner library, since the outputs of Figure 6 are equal and correspond to the output of running the SS library. These results do not allow the use of the same syntax as in ELDO, since both outputs cannot be gathered using the same netlist.

```
** NGSpice running only the corner

.lib '/umc_013/models/L130E_MM_TWINWELL_V182/L130E_HGIO_RVT33_V131/L130E_HG_RVT33_V131.mdl.eldo' SS

********** Analysis *************************
.TEMP 25.0
.OPTION BRIEF=0
*********************************************

.control

set filetype = ascii
set units = degrees

AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

quit

.endc
```

```
******
** ngspice-28 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please get your ngspice manual from http://ngspice.sourceforge.net/docs.html
** Please file your bug-reports at http://ngspice.sourceforge.net/bugrep.html
** Creation Date: Mon Sep 17 17:14:21 WEST 2018
******

Circuit: ** NGSpice running only the corner


No. of Data Rows : 2001
gdc                =  5.032540e+01
gbw                =  5.687887e+07
pm                 =  6.467351e+01
```

Fig. 5: Run only corner library

```
** NGSpice using ELDO Syntax

.lib '/umc_013/models/L130E_MM_TWINWELL_V182/L130E_HGIO_RVT33_V131/L130E_HG_RVT33_V131.mdl.eldo' TT

********** Analysis *************************
.TEMP 25.0
.OPTION BRIEF=0
*********************************************

.control

set filetype = ascii
set units = degrees

AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

.endc

.ALTER SS
.lib '/Users/Catarina/Desktop/Tese/Libraries/L130E_HGIO_RVT33_V131/L130E_HG_RVT33_V131.lib.eldo' SS

.control

AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

quit
.endc
```

```
******
** ngspice-28 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please get your ngspice manual from http://ngspice.sourceforge.net/docs.html
** Please file your bug-reports at http://ngspice.sourceforge.net/bugrep.html
** Creation Date: Mon Sep 17 17:14:21 WEST 2018
******

Circuit: ** NGSpice using ELDO Syntax


No. of Data Rows : 2001
gdc                =  5.032540e+01
gbw                =  5.687887e+07
pm                 =  6.467351e+01
Doing analysis at TEMP = 25.000000 and TNOM = 27.000000


No. of Data Rows : 2001
gdc                =  5.032540e+01
gbw                =  5.687887e+07
pm                 =  6.467351e+01
```

Fig. 6: Run typical and corner libraries in NGSpice using ELDO Syntax

D. *NGSpice Alterparam*

NGSpice provides a command, *alterparam*, that allows to change the value of global parameters or in sub circuits defined by the *.param* statement. Tests were done in order to evaluate if the results were coherent. The first scenario included running a netlist, with just a few measures, assigning fixed values to the parameters, without using the *alterparam* command to change its values, as represented in Figure 7.

```
.PARAM
+_w8=1.0000e-06
+_w6=7.1200e-05
+_w4=1.5100e-05
+_w10=1.6000e-06
+_w1=1.7000e-06
+_w0=2.6000e-05
+_nf8=5.0000e+00
+_nf6=7.0000e+00
+_nf4=8.0000e+00
+_nf10=3.0000e+00
+_nf1=3.0000e+00
+_nf0=3.0000e+00
+_l8=9.4000e-07
+_l6=7.5000e-07
+_l4=6.7000e-07
+_l10=9.4000e-07
+_l1=3.0000e-07
+_l0=4.9000e-07
```

```
********** Analysis ****************************
.TEMP 25.0
.OPTION BRIEF=0
************************************************

.control
set filetype = ascii
set units = degrees

AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

.endc
```

Fig. 7: Netlist without alterparam command

Following that, a netlist with different initial parameter values was run and, inside the control section, the alterparam command was executed changing the initial parameter values to the ones presented in Figure 7. Both the initial parameter values and the netlist run are represented in Figure 8.

```
.PARAM
+ _w8=5.0500e-05
+ _w6=5.0500e-05
+ _w4=5.0500e-05
+ _w10=5.0500e-05
+ _w1=5.0500e-05
+ _w0=5.0500e-05
+ _nf8=5.0000e+00
+ _nf6=5.0000e+00
+ _nf4=5.0000e+00
+ _nf10=5.0000e+00
+ _nf1=5.0000e+00
+ _nf0=5.0000e+00
+ _l8=6.4000e-07
+ _l6=6.4000e-07
+ _l4=6.4000e-07
+ _l10=6.4000e-07
+ _l1=6.4000e-07
+ _l0=6.4000e-07
```

```
********** Analysis ****************************
.TEMP 25.0
.OPTION BRIEF=0
************************************************

.control
set filetype = ascii
set units = degrees

AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

alterparam _w8=1.0000e-06
alterparam _w6=7.1200e-05
alterparam _w4=1.5100e-05
alterparam _w10=1.6000e-06
alterparam _w1=1.7000e-06
alterparam _w0=2.6000e-05
alterparam _nf8=5.0000e+00
alterparam _nf6=7.0000e+00
alterparam _nf4=8.0000e+00
alterparam _nf10=3.0000e+00
alterparam _nf1=3.0000e+00
alterparam _nf0=3.0000e+00
alterparam _l8=9.4000e-07
alterparam _l6=7.5000e-07
alterparam _l4=6.7000e-07
alterparam _l10=9.4000e-07
alterparam _l1=3.0000e-07
alterparam _l0=4.9000e-07

mc_source

AC DEC 200 1 10G
meas AC GDC FIND vdb(output) at=1
meas AC GBW WHEN vdb(output)=0
meas AC PM FIND vp(output) WHEN vdb(output)=0

quit

.endc
```

Fig. 8: Netlist changing parameter values using alterparam command

The outputs of both simulations are presented in Figure 9 and Figure 10, where the blue dashed line highlights the measure's values that should match. These results validate the use of this command once the desired values are achieved, since running a netlist with only a set of parameters (Figure 7), and running a netlist with the same measures, with different values for the initial parameters, and using alterparam to change them to the ones used in the first netlist (Figure 8) gives the same result.

```
******
** ngspice-28 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please get your ngspice manual from http://ngspice.sourceforge.net/docs.html
** Please file your bug-reports at http://ngspice.sourceforge.net/bugrep.html
** Creation Date: Mon Sep 17 17:14:21 WEST 2018
******

Circuit: ** Output running netlist without Alterparam

No. of Data Rows : 2001
gdc                 =  5.078321e+01
gbw                 =  7.931982e+07
pm                  =  6.174107e+01
```

Fig. 9: Output of running the netlist without alterparam command

```
******
** ngspice-28 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please get your ngspice manual from http://ngspice.sourceforge.net/docs.html
** Please file your bug-reports at http://ngspice.sourceforge.net/bugrep.html
** Creation Date: Mon Sep 17 17:14:21 WEST 2018
******

Circuit: ** Output running netlist with Alterparam
No. of Data Rows : 2001
gdc                 =  2.980741e+01
gbw                 =  1.666461e+08
pm                  =  8.373773e+01

Circuit: ** Output running netlist with Alterparam

No. of Data Rows : 2001
gdc                 =  5.078321e+01
gbw                 =  7.931982e+07
pm                  =  6.174107e+01
```

Fig. 10: Output of running the netlist with alterparam command

However, there is an important detail when using the alterparam command, since if every time a user changes parameter values, another command, mc_source, has to be also invoked in order to changes to be effective. Although in the NGSpice manual, there is a reference to use the reset command along with alterparam, when tested the results were not coherent, as illustrated in Figure 11:

```
******
** ngspice-28 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please get your ngspice manual from http://ngspice.sourceforge.net/docs.html
** Please file your bug-reports at http://ngspice.sourceforge.net/bugrep.html
** Creation Date: Mon Sep 17 17:14:21 WEST 2018
******

Circuit: ** Output using reset instead of mc_source

No. of Data Rows : 2001
gdc                 =  2.980741e+01
gbw                 =  1.666461e+08
pm                  =  8.373773e+01


No. of Data Rows : 2001
gdc                 =  2.976015e+01
gbw                 =  1.654791e+08
pm                  =  8.376679e+01
```

Fig. 11: Output of running the netlist using reset command instead of mc_source

V. NGSPICE INTEGRATION IN AIDA

AIDA's circuit optimizer (AIDA-C) was developed to allow the integration of user's simulation tools. This integration is only possible establishing a well-defined communication protocol between AIDA-C and the user's simulation engine. Using NGSpice as AIDA's evaluation engine requires the development of an interface, named ACEI.

A. *Proposed Solution*

Considering how NGSpice behaves, the proposed solution to integrate NGSpice in AIDA is presented in Figure 12:

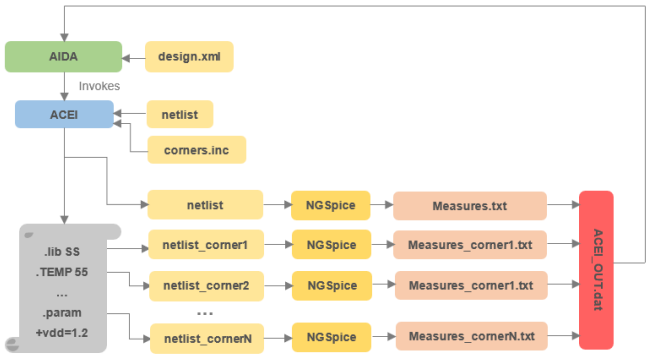

Fig. 12: Interaction between AIDA and NGSpice

As described in Figure 12, AIDA will invoke the interface (ACEI), so that the simulations run. However, before that, the interface has to generate the netlists needed, i.e., as much as the number of corners. These netlists are copies of the original one, except for the changes according to the corners.inc file. After that, for each netlist, the NGSpice will be invoked, in parallel, and each simulation will output one file containing the measures. Finally, all these files will be parsed into an output file, which will be returned to AIDA.

In short, apart from the syntax of the netlists, the user will not have to change the usual procedure when using AIDA. The interaction between AIDA, ACEI and the simulator, is simply described in Figure 13, where the ACEI developed is in between AIDA and the simulator, translating the simulation outputs to a well-define structure that AIDA understands.

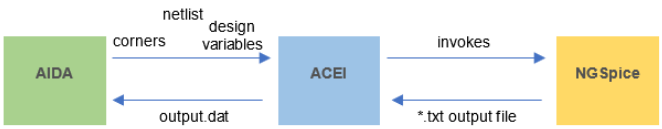

Fig. 13: Workflow between AIDA, ACEI and NGSpice

B. *Calculate Measures*

AIDA has a feature for calculating any measure that depends on other measures obtained from the same netlist, i.e., if the simulation outputs the voltage from gate to source (Vgs) and the threshold voltage (Vth) of a certain circuit device, AIDA is able to calculate the respective overdrive. However, NGSpice, as ELDO, allows to obtain these values when running the netlist. The user only has to assign the formulas of the desired measures to variables and print them to the standard output, as demonstrated in Figure 14, where both the overdrive and margin of a certain device is being calculated.

```
* Overdrive
let vov_mnm0 = @m.xinova.mnm0[vgs] - @m.xinova.mnm0[vth]
print vov_mnm0

* Margin
let delta_mnm0 = @m.xinova.mnm0[vds] - @m.xinova.mnm0[vdsat]
print delta_mnm0
```

Fig. 14: Calculate overdrive and margin in NGSpice netlist

Nevertheless, NGSpice has a limitation, since any measure that depends on measures from more than one analysis cannot be obtained when the simulation runs, due to NGSpice erasing from memory all vectors between different analysis. This barrier is overcome using the AIDA feature, gathering from the simulation only the measures needed to perform the calculation of the desired measure.

C. *Run ACEI*

Along with the script, to invoke the ACEI, the netlist, the option to run corners and the output file name have to be given as input: acei-ngspice.py --corners netlist.cir output-file.dat

When the ACEI is invoked, the first task is to evaluate if there are corners to run. If the option to run corners were given as input, then the netlist will be replicated for each corner and each thread will simulate one netlist at the time, until the queue that hold them is empty. Otherwise, if the --corners option was not invoked, then only the given netlist is simulated, having no reason to launch any thread. After simulating all netlists, the output file is written going through the measures dictionary.

D. *Netlists Generation to Run Corners*

Considering that using altermod to run corners entails replicating all libraries, replacing the labels defined by the values for each library and repeating the commands to extract all measures needed, for each corner, the best solution is replicating the original netlist and make all necessary changes to run each corner. The following schema exemplifies how the netlists are generated through the file that contains the corners:

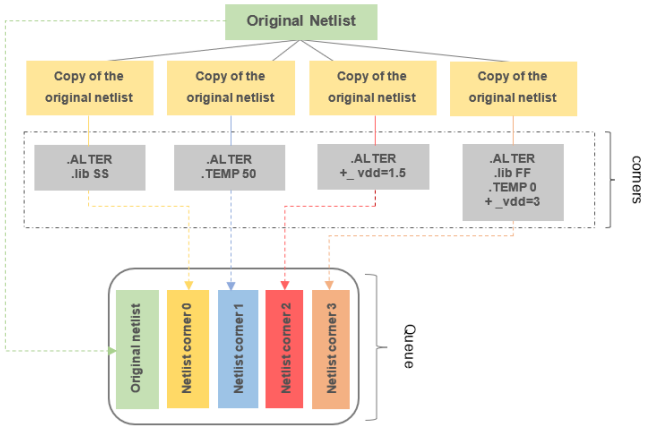

Fig. 15: Fill queue that contains the netlists to simulate (typical plus corners)

As represented in Figure 15, generating the netlists to run the corners, implies copying the original netlist. After that, according to the corner that will be simulated, each copy of the original netlist will be adapted, taking into account the type of corner:

- Change a library: the script looks for the include card, which is responsible for including the library, and replaces it for the new library;

- Change a parameter value: the script copies the design_var.inc file (which includes all circuit variables), finds the declaration of that parameter in that file and changes its value. Following that, the netlist has also to be changed to include, not the original design_var.inc file, but the file copied and changed;

- Change the temperature: the desired temperature is assigned to the .TEMP card.

Basically, to replicate the nestlits, the corners.inc file is read line by line and a dictionary holding each corner is filled. Then, for each key of this dictionary, a copy of the original netlist is performed. For each copy of the netlist, the corner changes are divided into two different lists: one that holds the temperatures and libraries to change, and another one to keep the new parameter values. For both lists, the copy of the original netlist has to be changed according to the corner. However, for the corners which modify parameter values, the design_var.inc file needs also to be cloned, since the assignment of the new values is performed in this new file. The only modification that needs to be addressed in the copied netlist is to include the new copy of design_var.inc file inside the respective netlist, instead of the original one.

Although the netlist and the design_var.inc files have nothing in common, both are changed using the same method, reusing the code. This method finds a pattern in the specified file and replaces it for a new value passed as input.

E. Run Each Corner

The queue that hold the netlists was built to allow the parallelization of the simulations. The following schema presented in Figure 16, shows how the script behaves until there are no more netlists to run.

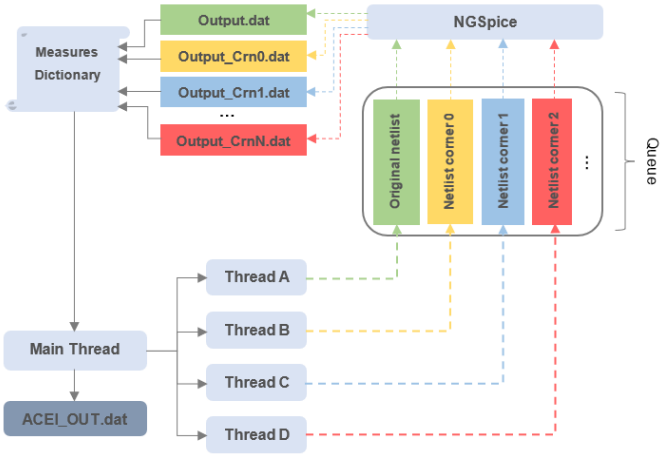

Fig. 16: Developed ACEI workflow

Basically, having a queue filled with netlists (original netlist plus netlists adapted according to the corners), each thread will pick up a netlist from the queue and executes NGSpice. The outputs obtained in each simulation will then be parsed to a dictionary, whose keys are the names of the performance measures and the values are the corresponding values. This dictionary will hold all the measures from all netlists simulated. As soon as all netlists are simulated, all threads will join the main thread, which will go over the measures dictionary and write the output file, according to the structure defined.

F. Generate Output File

The outputs of each simulation are registered in a list of dictionaries. Each dictionary holds the measures of each simulation. The dictionary keys are the names of the measures and the values are the corresponding outputs. The measures of each simulation are recorded in a certain index of the list which, in case of corners simulation, corresponds to the order of the corners in corners.inc file. An example of the output given by NGSpice, with the measures obtained from the AC and OP analyses is shown in Figure 17. When all netlists are executed, the main thread will go through the list and write the output file, printing each measure name and its corresponding value.

```
******
** ngspice-28 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please get your ngspice manual from http://ngspice.sourceforge.net/docs.html
** Please file your bug-reports at http://ngspice.sourceforge.net/bugrep.html
** Creation Date: Mon Sep 17 17:14:21 WEST 2018
******

Circuit: ** NGSpice Output

No. of Data Rows : 1801
gdc                 =  3.191890e+01
gbw                 =  3.374345e+06
gps                 =  3.434080e+00
psrr = 2.848482e+01
sr = 7.176034e+00
outswing = 1.046276e+00
vov_mpm0 = 1.056189e+00
delta_mpm0 = 6.014978e-01
```

Fig. 17: NGSpice Output

The output file, generated by the interface developed, is presented in Figure 18. All measures are indicated in the first line, with its labels separated by the Cornr# tag (highlighted with a blue dashed line) and, in the second line, the values of the respective measures are in the same order as its labels and separated by an ascending number starting at 0 (illustrated with an orange dashed line).

```
Cornr# gdc gbw gps psrr sr outswing vov_mpm0 delta_mpm0 Cornr# gdc gps psrr sr outswing vov_mpm0
delta_mpm0 Cornr# gdc gbw gps psrr sr outswing vov_mpm0 delta_mpm0 Cornr# gdc gbw gps psrr sr
outswing vov_mpm0 delta_mpm0 Cornr# gdc gbw gps psrr sr outswing vov_mpm0 delta_mpm0 Cornr# gdc
gbw gps psrr sr outswing vov_mpm0 delta_mpm0
0 3.191890e+01 3.374345e+06 3.434080e+00 2.848482e+01 7.176034e+00 1.046276e+00 1.056189e+00
6.014978e-01 1 -1.214326e+02 -5.996367e+01 -6.14689e+01 4.146960e-10 1.573023e+00 1.158391e+00
7.325856e-01 2 2.140988e+01 3.198940e+06 1.798278e+00 1.961160e+01 1.445500e+01 8.231050e-01
1.056189e+00 6.014978e-01 3 4.020755e+01 1.673411e+06 9.690863e+00 3.051669e+01 8.767923e-01
1.403326e+00 1.056189e+00 6.014978e-01 4 3.283458e+01 3.173416e+06 3.713671e+00 2.912091e+01
6.417714e+00 1.029244e+00 1.110165e+00 5.821023e-01 5 3.095500e+01 3.561300e+06 3.184819e+00
2.777018e+01 7.972108e+00 1.060133e+00 1.009218e+00 6.170891e-01
```

Fig. 18: Output file of ACEI for NGSpice

VI. Results

The AIDA framework optimizes circuits using multi-objective multi-constraint optimization techniques. Nevertheless, uses commercial simulators, as ELDO or HSPICE to evaluate the circuits performance, which prevents its dissemination among the academic community. The NGSpice integration in AIDA appears as the solution to abovementioned limitation, since is an open-source engine. However, this simulator will only be a powerful tool, if the outputs gathered are similar to the ones obtained with the current AIDA's simulators.

In order to compare the performance of both simulators, ELDO and NGSpice, two different circuits will run, in each simulator. The optimized circuits are presented in Figure 19 and Figure 20:

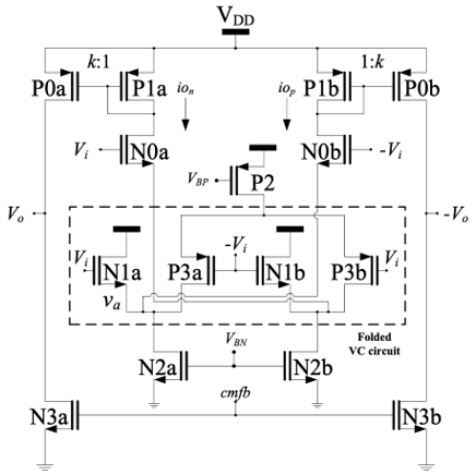

Fig. 19: Folded Voltage-Combiners Biased Amplifier circuit [15]

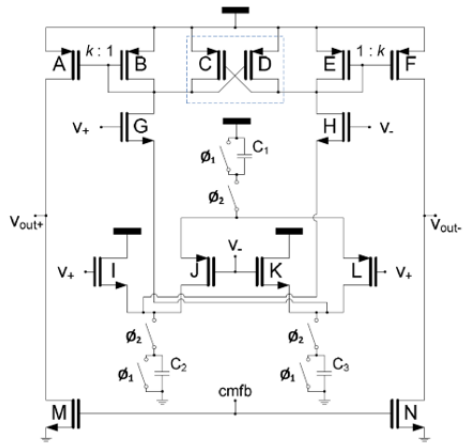

Fig. 20: Dynamic Voltage-Combiners Biased OTA circuit [16]

As AIDA optimizes circuits using a genetic algorithm based on a stochastic process and hashmaps, although each optimization starts with the same seed, the design variables simulated will differ among simulations and simulators. This implies that the measures obtained by each simulator will differ, which does not allow to compare directly the outputs gathered, since circuits with different design variables are being simulated. Taking this into account, the only way to compare both simulators is to simulate the same points optimized in AIDA with ELDO in NGSpice and vice-versa, and compare the measures obtained, not only for the typical library, but also for some corners.

When multiple objectives are considered, the output cannot be one single solution due to not being possible to define a solution better than another, since it is not a matter of one single performance measure being better. Instead, what is obtained is a set of optimal design trade-off solutions, commonly designated as a POF (Pareto Optimal Front). These solutions are non-dominated solutions, which are the feasible solutions which do not have any other solution better than the current one in a certain objective function without worsening another objective function.

A. *Pareto Optimal Front Obtained*

The objectives defined for both circuits were maximizing both the Gain DC and the GBW and minimizing the IDD. The following figures show the POFs obtained simulating both circuits, which are the optimal points considering the objectives defined, according to the Pareto Criterion. The points simulated while optimizing the circuits using ELDO were then simulated in NGSpice and vice-versa, so that the results would be comparable.

POFs obtained simulating ELDO points in both simulators:

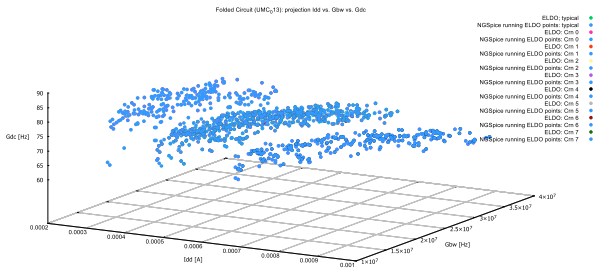

Fig. 21: POFs obtained simulating ELDO points (Folded Voltage-Combiners Circuit)

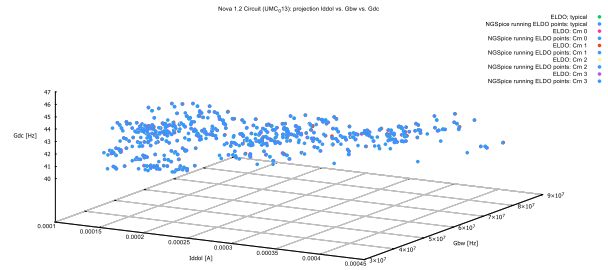

Fig. 22: POFs obtained simulating ELDO points (Dynamic Voltage-Combiners Circuit)

POFs obtained simulating NGSpice points in both simulators:

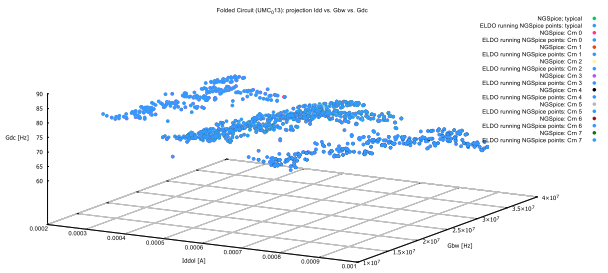

Fig. 23: POFs obtained simulating NGSpice points (Folded Voltage-Combiners Circuit)

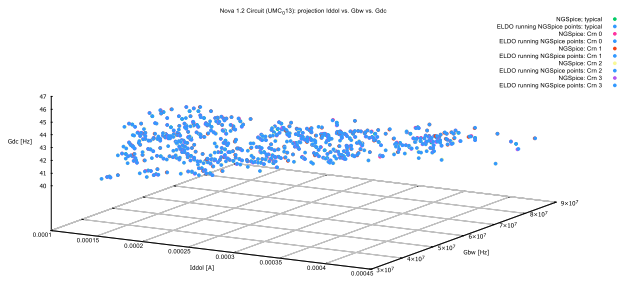

Fig. 24: POFs obtained simulating NGSpice points (Dynamic Voltage-Combiners Circuit)

The graphics in Figure 21 and Figure 22 present the POFs obtained for each circuit, considering that the points simulated in ELDO were also simulated in NGSpice, meaning that the POFs in multiple colours are the ones obtained using ELDO as AIDA's simulation engine and the blue ones are when NGSpice simulated the same points. On the other hand, the graphics in Figure 23 and Figure 24 represent the POFs, for each circuit, which result from simulating NGSpice points in both simulators, which entails that the POFs in multiple colours are the ones using NGSpice as simulator and the blue ones are when ELDO simulates the same points.

All POFs presented above prove that if the optimizations were performed simulating the same points, the outputs obtained from both simulators, ELDO and NGSpice, would be very similar, since when analysing these graphics, most of the blue points are overlapping the other solutions, not only for the typical library, but also for all corners.

B. *Statiscal Analysis*

In order to strengthen these conclusions, a statistical analysis over all the performance measures obtained will be presented. The values of the performance measures gathered from both simulators are paired samples, since these values are obtained for the same point. For each measure, in the first place, the relative error of each measure was determined, i.e., the absolute value of the differences between the outputs of both simulators were obtained. Considering the points generated when optimizing the circuits using ELDO, for each measure, the denominators of the mentioned relative errors are ELDO's outputs, while for the points generated using NGSpice as AIDA's evaluation engine, the denominators of the referred relative errors are NGSpice's outputs.

Following that, 95% confidence intervals for the mean value of the mentioned relative errors were performed, in order to evaluate with high probability, the variations of the errors between the simulators, for each measure. As all the samples associated with each measure and each simulation of the circuit have high dimension, in statistical terms $(n > 30)$, the respective sample mean of the relative errors of each one of them $(\bar{X})$ has an asymptotically normal distribution due to the Central Limit Theorem, so the variable,

$$Z = \frac{\bar{X}-\mu}{\frac{S}{\sqrt{n}}} \dot{\sim} N(0,1), \quad (1)$$

can be used to construct the 95% random confidence interval for the mean value (μ) of the relative errors:

$$\left]\overline{\mathrm{X}} - 1.96\frac{S}{\sqrt{n}}; \overline{\mathrm{X}} + 1.96\frac{S}{\sqrt{n}}\right[, \quad (2)$$

where S is the sample standard deviation of the relative errors. For each observed sample of the relative errors, a concrete 95% confidence interval is obtained:

$$\left]\overline{\mathrm{x}} - 1.96\frac{s}{\sqrt{n}}; \overline{\mathrm{x}} + 1.96\frac{s}{\sqrt{n}}\right[, \quad (3)$$

being $\bar{x}$ and s the point estimates of the sample mean and standard deviation, respectively.

As all the 95% concrete confidence intervals are obtained from random intervals of high confidence level (95%), there is an high probability that they contain the real mean value of the relative errors between the measures.

Generally, all the results obtained prove that the relative errors between both simulators, ELDO and NGSpice, are negligible since, with high probability, the upper limits of all deterministic confidence intervals obtained, for all performance measures were less than 13% and 0.3% for circuits in Figure 19 and Figure 20, respectively, since these intervals are based on 95% random confidence intervals. In fact, if infinite deterministic intervals were construct based on samples of the same size, it is expected that 95% of those intervals real contain the true mean value of the relative errors between both simulators, ELDO and NGSpice.

The maximum upper limit of the intervals obtained for the relative errors of circuit in Figure 19 was higher due to the *slewrate* measure. However, these differences are acceptable since, in ELDO, to calculate the slewrate the user only needs to invoke a command and the simulator handles how this measure is calculated. Thus, although the measure values are similar, there is no way to know exactly how this measure is obtained in ELDO, meaning that this measure's value can be slightly different. Looking at the other performance measures, the upper limits of the relative errors are negligible, less than 0.1%.

Considering the results presented, the integration of NGSpice in AIDA is validated since both graphs and 95% confidence intervals prove that the performance measures obtained are very similar. These results are extremely important to ensure that, from now on, AIDA can be disseminated without any constraints imposed by commercial simulators, namely the limited number of licenses available and the costs associated.

VII. Conclusions

Firstly, the simulator, namely the NGSpice, had to be tested in order to define how each measure can be obtained. These tests included discover the commands to get all measures, according to the analysis performed, and comparing the output values with

ELDO outputs, for the same circuits, in order to understand if the values match with each other.

After that, the main obstacle overcome was running corners, since the NGSpice native solution invoking altermod command is not the best approach, since increases the user complexity when using NGSpice as simulator. The proposed solution was to replicate the original netlist, as much times as the number of corners to run, according to the corners.inc file, which is part of AIDA's project folder. These netlists are run in parallel so that the execution time of following this approach was not too time consuming.

In addition, considering that NGSpice erases from memory all measures got from previous analysis whenever a new analysis invoked, an AIDA feature will be used to fill this gap. Basically, instead of calculating measures that depends on other measures gathered from different analyses in the netlist, the user only has to write down the corresponding formula, in the design.xml file, which is part of the AIDA's project.

All the research and tests performed, using both simulators (NGSpice and ELDO), validate the integration of NGSpice in AIDA. The interface that translates NGSpice outputs to AIDA is working, and multiple optimizations were run. The results obtained were compared both qualitative and quantitatively. The graphics presented confirm that, if the optimizations run over the same points, the POFs obtained would be very similar. In addition, considering the 95% confidence intervals obtained for all performance measures, for both circuits, there is no doubt that the outputs of both simulators are very similar.

Considering the results presented, the main goal was accomplished and, from now on, AIDA will not depend on any commercial simulator to optimize any circuit, since NGSpice is an open source simulation engine, without the constraint of having limited licenses, which allows to disseminate AIDA among all community.

REFERENCES

[1] S. Drimer, "EDA is dead. What comes next is exciting," 3 May 2016. [Online]. Available: S. Drimer, "EDA is dead. What comes next is exciting.," 3 May 2016. [Online]. Available: https://medium.com/@saardrimer/eda-is-dead-what-comes-next-is-exciting-cd5f3301402b.

[2] H. Vogt, M. Hendrix and P. Nenzi, "NGSpice Users Manual," 2 September 2018. [Online]. Available: http://ngspice.sourceforge.net/docs/ngspice-manual.pdf..

[3] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," 1975. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/1975/ERL-520.pdf.

[4] "ACEI – AIDA-C Custom Evaluation Interface User Guide," ICG (Integrated Circuits Group) research team at the associated lab Instituto de Telecomunicações, 2014.

[5] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," vol. 220, no. 4598, 13 May 1983.

[6] A. Torralba, J. Chavez and L. Franquelo, "FASY: a fuzzy-logic based tool for analog synthesis," vol. 15, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1996, pp. 705-715.

[7] K. Meng, P. C. Pan and H. M. Chen, "Integrated hierarchical synthesis of analog/RF circuits with accurate performance mapping," in *12th International Symposium on Quality Electronic Design*, Santa Clara, CA, USA, 2011.

[8] C. W. Lin, P. D. Sue, Y. T. Shyu and S. J. Chang, "A bias-driven approach for automated design of operational amplifiers," in *International Symposium on VLSI Design, Automation and Test, Hsinchu*, Taiwan, 2009.

[9] D. E. Goldberg, "Genetic and evolutionary algorithms come of age," in *Communications of the ACM*, vol. 37, 1994, pp. 113-119.

[10] M. Barros, J. Guilherme and N. Horta, Analog Circuits and Systems Optimization based on Evolutionary Computation Techniques, 1 ed., vol. 294, Springer-Verlag Berlin Heidelberg, 2010.

[11] M. Barros and N. Horta, "GA-SVM feasibility model and optimization kernel applied to analog IC design automation," in *Proceedings of the 17th ACM Great Lakes Symposium on VLSI 2007*, Stresa, Lago Maggiore, Italy, 2007.

[12] R. Santos-Tavares, N. Paulino, J. Higino, J. Goes and J. Oliveira, "Optimization of multi-stage amplifiers in deep-submicron CMOS using a distributed/parallel genetic algorithm," in *IEEE International Symposium on Circuits and Systems, Seattle*, Seattle, WA, USA, 2008.

[13] M. Krasnicki, R. Phelps, R. Rutenbar and L. Carley, "MAELSTROM: efficient simulation-based synthesis for custom analog cells," in *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, New Orleans, LA, USA, USA, 1999.

[14] R. Phelps, M. Krasnicki, R. Rutenbar, L. Carley and J. Hellums, "Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, 2000, pp. 703-717.

[15] J. Koza, F. Bennett, D. Andre, M. Keane and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 109 - 128, July 1997.

[16] T. Sripramong and C. Toumazou, "The invention of CMOS amplifiers using genetic programming and current-flow analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1237 - 1252, November 2002.

[17] Y. Hongying and H. Jingsong, "Evolutionary design of operational amplifier using variable-length differential evolution algorithm," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, Taiyuan, 2010.

[18] S. Chu, H. Huang, J. Roddick and J. Pan, "Overview of Algorithms for Swarm Intelligence," in *Computational Collective Intelligence. Technologies and Applications*, 2011.

[19] K. H. Hingrajiya, R. K. Gupta and G. S. Chandel, "An Ant Colony Optimization Algorithm for Solving Travelling Salesman Problem," *International Journal of Scientific and Research Publications*, vol. 2, no. 8, 2012.

[20] S. Kamisetty, J. Garg, J. Tripathi and J. Mukherjee, "Optimization of Analog RF Circuit parameters using randomness in particle swarm optimization," in *2011 World Congress on Information and Communication Technologies*, Mumbai, India, 2011.

[21] "Gnucap," 2006. [Online]. Available: https://www.gnu.org/software/gnucap/gnucap.html.

[22] "Quite Universal circuit simulator," [Online]. Available: http://qucs.sourceforge.net.

[23] Sandia National Laboratories, "Xyce," [Online]. Available: https://xyce.sandia.gov.

[24] C. Aleixo, "acei-ngspice," 2019. [Online]. Available: https://github.com/catarinaacfa/acei-ngspice/tree/master/acei-ngspice.