ActIC: Automated Characterization Test of Mixed-Signal Integrated Circuits

Tiago H. Moita, Carlos B. Almeida
Dep. of Electrical and Computer Engineering
IST – UTL / INESC-ID
Lisbon, Portugal
tiago.moita@ist.utl.pt, cfb@inesc-id.pt

Marcelino B. dos Santos
Dep. of Electrical and Computer Engineering
IST – UTL / INESC-ID, SILICONGATE LDA
Lisbon, Portugal
marcelino.santos@silicongate.com

***Abstract* — This paper presents a new methodology that establishes a bridge between design and test of mixed-signal integrated circuits. The information present in the schematics, used for design validation (testbenches) during the design phase, is usually not reused in the characterization test of the silicon. Moreover, the feedback that designers can obtain from the lab is presently limited since designers are frequently not familiar with the configuration of the test setup and the visualization options available for the test results. The aim of this work is to allow the characterization test of integrated circuits through the use of interfaces that are familiar to designers: the schematics used for design validation and the visualization tools they use for their simulation results. This approach allows the use of silicon as it was an additional simulator now available for the designer, thus closing the design loop, at the same time that allows a reduction of cost and time devoted to the characterization test.**

Keywords – Mixed-Signal Test; Test Automation; Characterization Test; Virtual Instrumentation; Design and Test Interface.

I. INTRODUCTION

The main problem faced during the test of Analogue and Mixed-Signal (AMS) Integrated Circuits (IC) relies in the circuit complexity since these circuits present distinct design characteristics [1] [2]. Moreover, while for digital ICs the design of testable devices is a topic having nowadays considerable solutions (e.g. [3] or [4]), for AMS ICs, this is an open topic where there are still few overall test methodologies that are not always widely accepted. In fact, while some of the first books addressing digital testing were published in early 1970s (e.g. [5] or [6]) and nowadays design and test of complex digital circuits are automated, for the AMS case, efforts are still oriented to specific techniques and their use on a certain class of circuits [7].

One of the major issues concerning the test of AMS ICs arises from the need to consider continuous signals and circuit parametric deviations, in addition to catastrophic faults [8]. The characterization test of these devices implies the need to consider parametric deviations, which leads to the definition of analogue test metrics that must also take into account the circuit functionality [8]. Moreover, the domain of AMS testing has always tried to cope with a controversy between functional and structural testing, however, functional testing is practically always considered [8] since applying structural testing to analogue circuits requires detailed analysis and modifications on a case-by-case basis [2].

Additionally, the test community still faces another challenge since there has always been a gap between the design and test of circuits. Design and test processes work most of the time as separated areas, not considering the influence that each department's activities have on those of the other, and thus without considering the benefits that the interaction between them can produce [9]. In most of the cases, after the design process is concluded, designers tend to have difficulties in following the test process of the designed circuit since they are not familiar with the test process. The main problem is that the tools used during the test process are different from those used during the design. Thus, design and test activities end up working as separated activities, when in reality they should be managed as concurrent processes [2] since a proper use of the design information can ease up the test process and the interaction with the characterization test is mandatory to close the design loop.

At industrial level there are some solutions (e.g. [10] or [11]) that try to solve this problem. However, these ones present limitations in terms of communication interfaces and have major costs associated which turn them not affordable for research groups or small companies. Moreover, these solutions do not really establish the desirable path between design and test.

Considering the aforementioned facts, it becomes mandatory to develop dedicated testing solutions having an adequate link between design and test but also capable of optimizing and automating the test process. Therefore, the methodology here presented has for main goal the development of a new test solution to automate the characterization test of AMS circuits.

The main goal of the proposed methodology is to make available to designers a tool to characterize the Device Under Test (DUT) using the same type of interface they are familiar: a schematic view that is usually exported as a *SPICE* [12] netlist and the same waveform viewers involved in the design environment, without the need to turn designers into test experts.

In order to achieve these goals, the methodology provides to the designer a library of cells with sources and loads that can

be included in the schematic testbench with two purposes: simulation and configuration of the test setup.

Although the present methodology is general for all kinds of AMS cores, due to the participation of *SILICONGATE* in this research work, particular emphasis is being given to power cores.

To achieve the proposed goals, an *Automated Characterization Test Interface for Mixed-Signal Integrated Circuits* (ActIC) was developed. The ActIC implementation considers the development of two main areas: platform hardware and user interface software that includes the necessary control mechanisms for the test equipment.

The hardware platform is based on the lab equipment and two types of dedicated boards: a daughterboard where the DUT connections are hardwired and a main board that contains the test structures (e.g. programmable voltage sources, programmable loads, etc.). Using this approach, it is possible to reuse the main board whenever a new DUT needs to be characterized.

Concerning the software implementation, to guarantee that designers interact with the test setup similarly as they do with the simulation tool, the developed solution consists in configuring the lab equipment based on the netlist exported from the design schematic view. A *Test Scheduler Tool* [13] is also included in order to sequence the variation of different parameters during the simulation and also during the characterization test process. This tool has a Graphical User Interface (GUI) and produces a *SPICE* compatible output, allowing the use of *Alter* and *Parameter Definition* files suitable for simulation or test setup configuration using ActIC. The developed software is based on the Virtual Instrument Software Architecture (VISA) [14] and on the Interchangeable Virtual Instruments (IVI) drivers [15] in order to allow instruments interchangeability without the need to change or rewrite any code to allow the introduction of new measurement instruments into the test platform.

II. THE ACTIC METHODOLOGY

The main concept of the present methodology is the usage, in the design validation schematic (testbench), of a library of sources, loads and probes, which are used for simulation purposes but have a correspondence with the available hardware and thus are used for its automatic configuration. ActIC controls the hardware and converts the acquired values into the format required by the visualization tools that designers use for analysis of the simulation results. Designers are thus encouraged to interact directly with the test process, without the need to have deep knowledge in the test area.

Considering that the interface between the design schematic entry and the simulator is the netlist, this same netlist is also used to interface with ActIC. The format chosen for the netlist is the *SPICE* format because it is widely used in the design environment.

Nevertheless, several challenges arise to accomplish the proposed objectives, namely, hardware limitations. For example, during the translation of the *SPICE* commands to stimuli and physical configurations applied to the test equipment, one must consider the real limitations of the equipment. Additionally, although for the design team the circuits can be observed and tested through simulation having total access to any node of the circuit, the test team only has access to a limited number of nodes. Thus, in order to implement the proposed methodology, one must define that the IC pads (inputs/outputs) will act as the common interface for both test and design testbench simulation.

The process starts with the core design and the definition of the schematic used for design validation with *HSPICE*® [16] (testbench). The simulation setup used for running circuit simulations can now be reused for the configuration of the test platform. To guarantee a full compatibility between the platform and the simulation tools, not only the netlist extracted from the schematic must be analyzed, but also the auxiliary files (normally used for simulation purposes) have to be considered. Thus, to provide a complete interaction between design and test environments, ActIC also reads the files or sections identified as *Header*, *Footer*, *Parameter Definition* and *Alter* files.

The *Test Library* (*TL*) was developed using the designer's schematic entry system (in this case the *Cadence*® software is used). The main goal was to define cells that represent the available equipment in order to allow the designer to use it on the testbench, as a normal source or load. Also, to allow the expansion of the system, the *TL* can be easily updated with new cells in order to include new test instruments.

ActIC also includes a *Test Scheduler Tool* [13] that controls the sequence of parameter variation in the netlist used for simulation and for characterization test purposes. The optimization of this sequence is particularly important for speeding up the test. For example, considering different test temperatures, it should be guaranteed that all other parameters are tested before changing the temperature condition, since usually the process of changing the test chamber temperature is a time consuming task. Moreover, the proposed *Test Scheduler Tool* also provides the means to easily exclude specific parameters combinations that should not be simulated or tested because they correspond to parameter combinations that are not specified or may cause system failure. As an example, a boost DCDC converter may present different maximum load current for the minimum and for the maximum input voltage due to a duty cycle limitation. These restricted combinations of possible values for parameters (Process, Voltage and Temperature: PVT corners) are usually obtained editing manually the corners *Alter* file. The *Test Scheduler Tool* in ActIC uses a GUI to allow easy selection and exclusion of parameter combinations and automatically generates the *Alter* file with all the desired PVT corners.

The communication between ActIC and the lab equipment relies on the VISA and IVI drivers. The goal is to allow the expansion of the system and instruments interchangeability developing a system that can interface any type of instrument without the need to rewrite any code when there is a need to change or add new equipment. Furthermore, nowadays the major instrument manufacturers present test equipment compatible with IVI drivers.

III. ACTIC IMPLEMENTATION

The software is responsible for parsing the netlists, identification of test cells and configuration of the test equipment, based on the parsed information. After the application of the test, the program must also convert the acquired results into a format compatible with the simulation tools, like the *CosmosScope*® [17] waveform viewer.

Considering the aforementioned facts, the proposed software implementation has five main functions: 1 – development, in *Cadence*® environment, of the *TL* with cells that correspond to lab equipment, that can be simulated in *HSPICE*® and that include the relevant parameters for the configurations of the sources, loads or lab equipment they represent; 2 – parse the *HSPICE*® netlist, identify the equipment used in the testbench and obtain the corresponding configuration; 3 – implementation of a *Test Scheduler Tool* that controls the equipment according to the test flow obtained from the netlist; 4 – parsing of the measures obtained and formatting them into a proper format for wave visualization in the design flow (e.g. *CosmosScope*®); 5 – development of a GUI to allow the interaction of designers and the test engineers with ActIC.

The software was developed in a modular way to allow parallel development of the software modules, allowing its test and integration in ActIC. It was implemented in the Python programming language with each module being responsible for the implementation of a specific set of functions. Moreover, the software also includes a central main routine that interacts and manages the different functions of each module.

Fig. 1 depicts the project flow resulting from the use of the proposed software.

The next paragraphs will present a description of some of the most relevant elements of the proposed software implementation.

A. TEST LIBRARY (TL)

The *TL* contains cells that represent sources, loads and measurement equipment available in the lab. By creating an instance and setting the parameters of one *TL* cell, the designer has the ability to configure the corresponding equipment.

The *TL* was created, using *Cadence*® native components of *AnalogLib*, for the use in *Cadence*® schematic system using a unique representation for each resource. Additionally, each cell not only represents the test equipment but it also considers some of its main physical specifications and non-ideal characteristics to keep the simulations closer to the lab results. Moreover, to guarantee the expandability of the system, the *TL* is open and allows the easy inclusion of new instruments.

As an example, Fig. 2 depicts the electrical representation and the *TL* symbol of a *HP/Agilent*® *34401A* voltmeter.

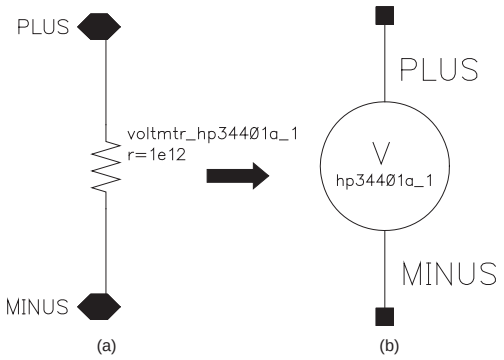

Figure 2. Example of a *TL* element: (a) electrical representation; (b) symbol

B. ActIC

In order to guarantee full compatibility with the simulation tools, ActIC performs the parsing not only of the *HSPICE*® netlist but also of all the auxiliary files normally included for simulation purposes: *Header*, *Footer*, *Alter* and *Parameter Definition* files. *Header* and *Footer* are files that complement the netlist, adding parameter definitions and simulation options (e.g. simulation type commands) that are not obtained by the schematic extraction. These files are always included in the netlist to define parameters and give simulation commands. *Parameter Definition* files includes the definition of the parameters used in all sub-circuits. In order to ensure that the project parameters used by all designers are the same, there is

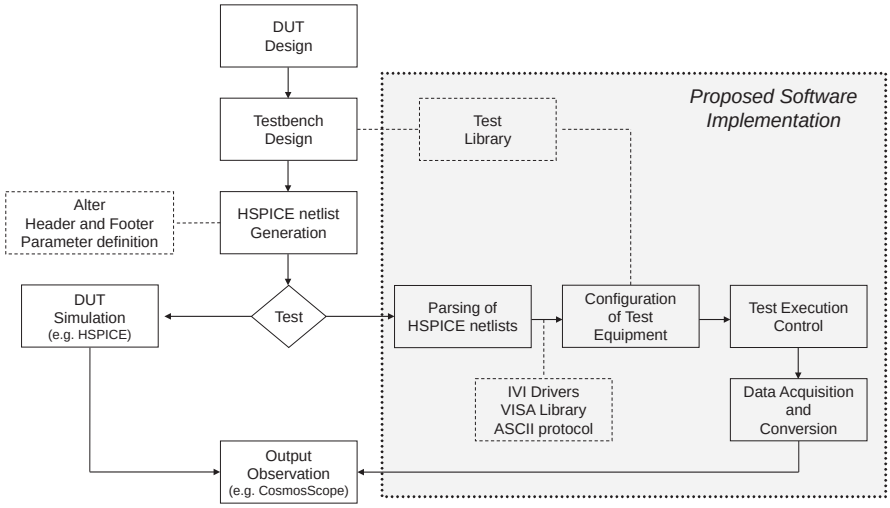

Figure 1. Project flow with ActIC: integrated testbench design and test configuration

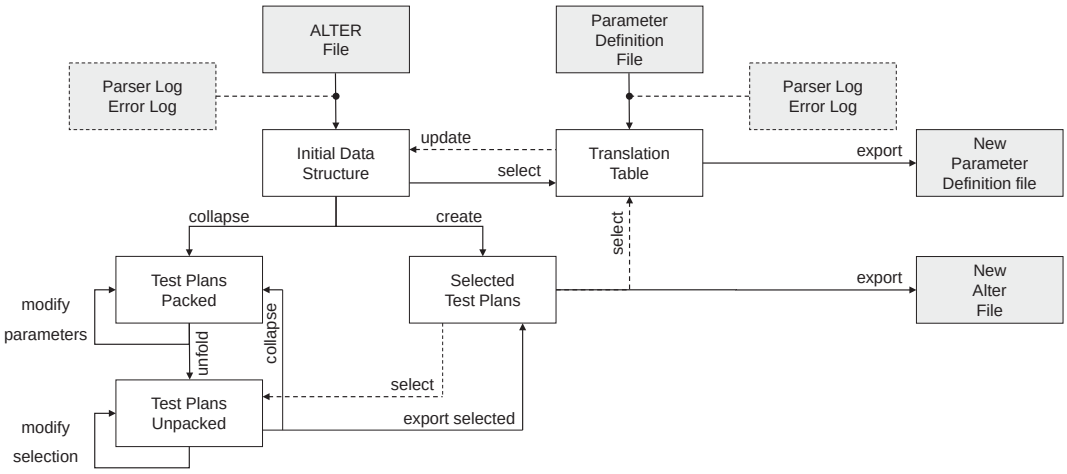

Figure 3. Data flow diagram of the *Test Scheduler Tool*

only one file with the project parameters that is included in all netlists prior to their simulation.

The software interface tool manages six auxiliary modules: *Netlist*, *Alter*, *Parameter Definition*, *Graphical User Interface* (GUI), *Instruments Configuration* and *Test Scheduler Tool*.

The *Netlist* module is responsible for the parsing of the *HSPICE®* netlist, *Header* and *Footer* files. The module parses the netlist and identifies: *.PARAM*, *.TEMP*, *.INCLUDE* and *.SUBCKT* directives. At the same time, the module also analyses other *HSPICE®* files included through the *.INCLUDE* directive and their location is stored in memory structures accessible to the other modules.

The *Alter* module is responsible for parsing any included *Alter* file. This module starts by verifying the file and, afterwards, parses it to collect the useful information. The information contained in an *Alter* file is related to the simulation corners through the enumeration of all the test parameters to be altered through the use of several *.ALTER* directives. Thus, the *Alter* module analyses each directive in order to identify the tests to be executed, knowing that all the data available between each *.ALTER* directive corresponds to the declared *Alter* identified by the previous referred directive.

The module parses the file and builds a list of parameters to be changed for each detected *.ALTER*, creating at the same time a list of the collected *.ALTER* directives themselves, i.e., the sequences of tests to be executed.

The *Parameter Definition* files depict information about all the parameters used in all sub-circuits. Moreover, this type of file normally presents an organization based on a 1:1 relation, i.e., *".PARAM Parameter = Value"*. Thus, since the objective of the *Parameter Definition* module is to parse the information contained on this type of file, the module creates a Python dictionary to save the collected data. Moreover, the *Parameter Definition* file, in its *HSPICE®* regular form, also allows the user to describe parameters as mathematical expressions (*"parameter = expression"*) or to use unit prefixes (e.g. u for micro). Therefore, to guarantee full compatibility with the simulation tools, the present module also supports this type of definition in order to avoid the need to make any changes to the original file. From the designer point of view, the tool behaves exactly as any *HSPICE®* parser would do.

A *Graphical User Interface* (GUI) module was developed to ease up the interaction with the software. The GUI is based on the *Tkinter* library [18] that works on a multiplatform basis. Thus, all operations that require user interaction are interfaced through the GUI menus and information text boxes.

Based on the collected data, the *Instruments Configuration* module issues the necessary commands to control the test setup. This module considers auto-set mode for all the devices, except when the user explicitly defines a specific range or configuration. The *Instruments Configuration* module execution can be described as having two different phases. In a first phase, the module verifies what are the instruments present on the netlist and it checks if the instruments are available at the moment. On a second phase, the instruments are configured and the tests are performed. At this time, two types of test execution can occur, namely, corner simulation (execution of tests to characterize the DUT considering the several corners declared in the *Alter* file) or simple test execution (for example for DC measurements without variations or time dependent tests).

Finally, it is important to present the module responsible for the *Test Schedule Tool* [13]. In order to automate the creation of *Alter* files that define the simulation and test sequence, a software module, whose data flow diagram is depicted in Fig. 3, was developed. It is worth to refer that all included processes in Fig. 3 are internal processes for satisfying the application automation, except "*modify parameters*" and "*modify selection*" which require user actions. Moreover, the grey blocks are accessible to the user or require user input, whereas the rest of the blocks represent application-specific internal structures. All the user interactions are performed by means of an intuitive GUI.

The input is an *Alter* file that identifies the parameters to be changed during simulation or test and the output is a modified *Alter* file with the sequence and combination of parameters defined by the user through a GUI. The proposed solution provides a mechanism to create a hierarchy that corresponds to a sequence of parameters that are changed, giving also the

Figure 4. Parameters modification and test sequences selection interface of the *Test Scheduler Tool*

possibility to exclude test parameters and combinations of parameter values, if needed.

In order to deliver the presented set of functionalities, besides the analysis of traditional *HSPICE®* compliant *Parameter Definition* and *Alter* files, the module also implements a new proprietary extension developed to allow the designation of several mutually exclusive values for a single parameter, whose complete set of combinations has to be considered in the final output simulation model. Thus, this new functionality is accomplish by the introduction of the reserved *OR* operation (the || symbol is used for the *OR* operation) to the normal *HSPICE®* specification to allow the concatenation of parameter values in a single statement (e.g. *.PARAM vin = vinmax || vinmin*). The introduction of this new functionality allows the input *Alter* file to be now specified in the traditional *HSPICE®* format, or by relying onto the *OR* operation to enumerate all possible parameter values, or by combining both approaches. Nevertheless, it is important to refer that, independently of the new implemented functionalities, the module generated *Parameter Definition* and *Alter* files continue to be strict *HSPICE®* compliant.

Fig. 4 shows the interface of the proposed *Test Scheduler Tool*, presenting the user interface where the designer is allowed to change the order of testing parameters by simply altering their relative priority (with + and - buttons), or to include or exclude complete set of parameters, and showing the interface where the user can select a subset of automatically generated test sequences.

IV. Experimental Results

This section presents an example of the application of the proposed methodology to a real case scenario. The example uses the ActIC platform developed tools and corresponds to the characterization test of a LDO (Low-dropout Regulator) core fabricated in *AMS 0.35µm* technology.

The used testbench, depicted on Fig. 5, corresponds to a line transient characterization test that varies the core input voltage between 2.8V and 3.3V with a constant output load current of 30mA. The input voltage of the core is given by a *pwl* voltage source, available on the *TL*. Additionally, to measure the input and output voltages of the LDO, two oscilloscope cells, available on *TL*, representing different channels of a *Tektronix® TDS 540C* oscilloscope, were used.

After the testbench was complete, a *HSPICE®* netlist was generated. The results were collected after the automatic configuration and execution of the test and simulation of the testbench and plotted with *CosmosScope®*. The simulation results and those obtained through the test under real conditions

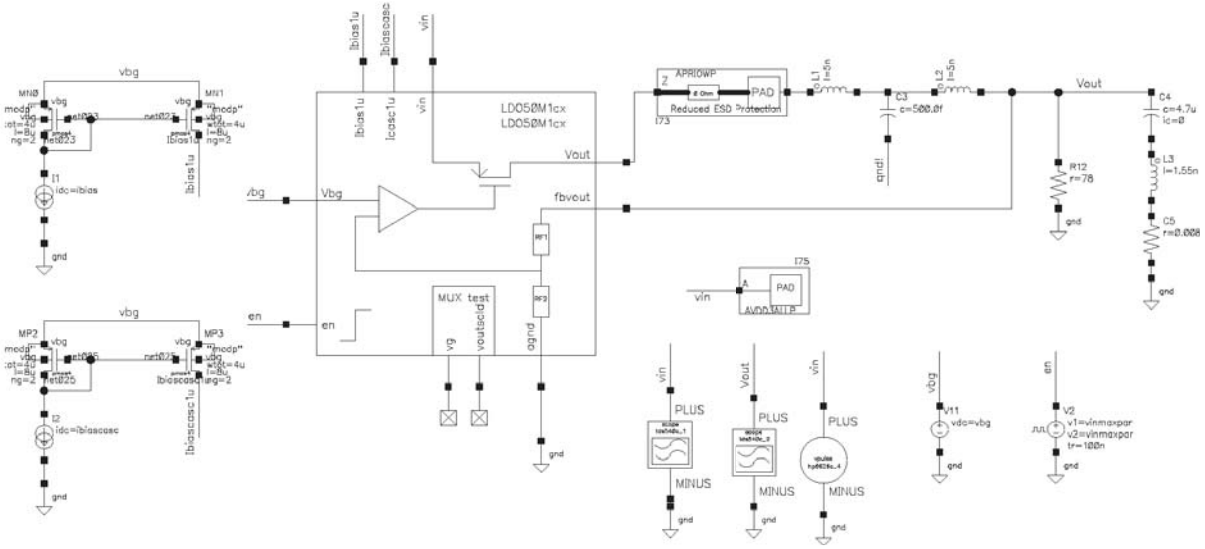

Figure 5. Testbench of a LDO core line transient characterization test

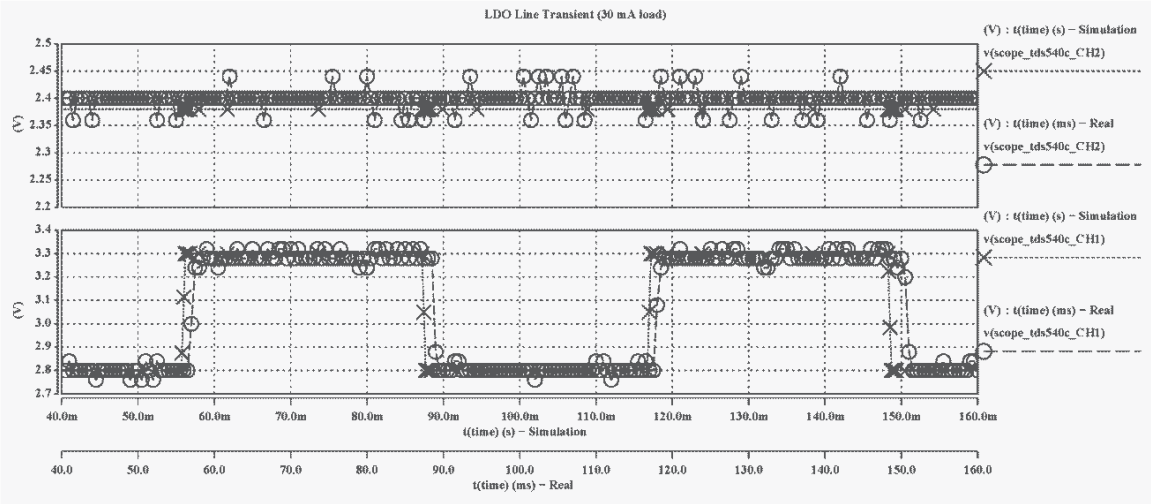

Figure 6. Laboratory versus simulation results for the LDO line transient test

can be observed on Fig. 6.

The test setup was totally configured using exactly the same testbench used by the core designers to simulate the core. Moreover, using the *TL* cells, not only the designer is now able to configure the entire test setup using a tool that he understands, but also, since the cells of the *TL* include models for instrumentation's non-ideal behavior, the simulations conducted are closer to the real laboratory results.

V. CONCLUSIONS

This paper presents a solution to integrate the design and characterization test of AMS cores that is supported by a *Test Library* (*TL*) and ActIC. The *TL* includes cells with a direct correspondence to available lab equipment. These cells can be used in the schematic testbench for simulation at the same time that they are recognized in the netlist parser of ActIC that extracts the relevant information for equipment configuration. ActIC also includes a *Test Scheduler Tool* that allows the easy selection of the corners the designer wants to simulate and writes them in an *HSPICE*® compatible *Alter* file that can be used in simulation or used by ActIC to schedule the lab test.

The main advantages for designers are the use of the *Test Scheduler Tool* (of ActIC) and the possibility to close the design loop by interacting directly with the silicon using the same interfaces they use for simulations: *Cadence*® schematic editor and a visualization tools like *CosmosScope*®.

The main advantage for test engineers is the automatic configuration of the equipment for long sequences of measurements with different parameters (e.g. temperatures, input and output voltages and load currents).

ACKNOWLEDGMENT

T. H. Moita acknowledges the contribution of FCT – Fundação para a Ciência e a Tecnologia, by supporting him with a Fellowship (Ref: SFRH/BD/61505/2009). This work was partially funded by FCT through the project PTDC/EEA-ELC/113902/2009 and through the INESC-ID multiannual funding (PEst-OE/EEI/LA0021/2011).

REFERENCES

[1] B. Vrignon and S. Dhia, "On-chip sampling sensors for high frequency signals measurement: evolution and improvements," 5th IEEE International Caracas Conference on Devices, Circuits and Systems, pp. 270-275, 2004.

[2] N. C. Lee, "Practical considerations for mixed-signal test bus," International Test Conference, pp. 591-592, 1993.

[3] M. Abramovici, M. Breuer and A. Friedman, Digital Systems Testing and Testable Design. IEEE Press, 1990.

[4] J. Semião et al., "Exploiting parametric power supply and/or temperature variations to improve fault tolerance in digital circuits," 14th IEEE International On-Line Testing Symposium, pp. 227-232, 2008.

[5] H. Chang, E. Manning and G. Metze, Fault Diagnosis of Digital Systems. New York: Wiley – Interscience, 1970.

[6] A. Friedman and P. Menon, Fault detection in digital circuits. Prentice-Hall, 1971.

[7] J. Augusto and C. Almeida, "A tool for single-fault diagnosis in linear analog circuits with tolerance using the T-Vector approach," VLSI Design, 2008.

[8] A. Bounceur, S. Mir, L. Rolíndez and E. Simeu, "CAT platform for analogue and mixed-signal test evaluation and optimization," IFIP International Conference on Very Large Scale Integration, pp. 320-325, 2006.

[9] D. Bello, R. Tangelder and H. Kerkhoff, "Modeling a verification test system for mixed-signal circuits," IEEE Design & Test of Computers, vol. 18, no. 1, pp. 63-71, 2001.

[10] National Instruments Corporation, "Integrate LabVIEW with SPICE/PSpice Simulators". Available: http://www.ni.com/white-paper/3514/en

[11] National Instruments Corporation, "NI PXI Power Management IC (PMIC) Test System". Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/207104

[12] A. Vladimirescu, The SPICE Book. John Wiley & Sons Inc, 1994.

[13] A. Ilic, T. Moita, C. Almeida and M. Santos, "Test Scheduler: A design based tool for test automation," XXVI Conference on Design of Circuits and Integrated Systems, pp. 195-200, 2011.

[14] IVI Foundation, VPP-4.3: The VISA Library (Revision 5.0), 2010.

[15] IVI Foundation, Getting Started Guide, 2009.

[16] Synopsys, "HSPICE® Simulator". Available: http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx

[17] Synopsys, "CosmosScope® Graphical Waveform Analyzer". Available: http://www.synopsys.com/systems/saber/pages/cosmos_scope_ds.aspx

[18] F. Lundh, An introduction to Tkinter, 1999.