A New Verification Approach for Mixed-Signal Systems

G.Bonfini, M. Chiavacci, R. Mariani, R. Saletti*

YOGITECH SpA, via Lenin 132/p, 56017 San Martino Ulmiano (Pisa), Italia

*Dipartimento Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni
University of Pisa, Via Caruso, I-56122 Pisa, Italia

ABSTRACT

Mixed-signal systems are growing in the semiconductor market and are becoming more and more complex. On the other hand, bugs and malfunctions still appear in these systems mainly because of the lack of interaction between digital and analogue verification. This paper presents an integrated approach providing the designers with a methodology and a set of IPs to interface advanced digital verification environments with mixed-signal simulators based on Analogue-HDL. A theoretical approach based on statistical method is also presented as well as a case study on a CAN-bus transceiver.

1. INTRODUCTION

Due to the complex interaction between analogue and digital verification methodologies [2], the overall verification coverage of a mixed signal system is very often decreased and system bugs appear very late in the design process. If verification of digital sub-systems is based on advanced techniques [1-3] such as constraints capture, randomised or pseudo-randomised stimulus-generation, result collection with coverage analysis, on the other side the verification of analogue sub-systems is based on classical approaches, focused to transient and AC effects. In this case, the requirement of precise behavioural models of the investigated sub-systems is a must [4-8], in order to decrease simulation complexity and achieve an acceptable run-time.
The present digital verification key issues are *functional coverage* and *constraint-driven random test generation*. Functional coverage is the systematic procedure to assess how and how much each "coverage item", or specification requirement, has been covered by the tests. This procedure should become a must of the entire mixed-signal verification flow and should act as a metric for the various verification tasks. Constraint-driven random test generation (i.e. generation of random stimuli with the possibility to constraint such process) is another key feature to avoid that some expected (in the real application) corner cases may not be investigated. However, it is important to highlight that the aim of the approach proposed in this paper is not to replace the designer simulation contribution with a simulator tool but, once the specification document, the application environment and the verification scenario are defined (the main system bottlenecks should be already well known), the proposed approach helps to measure how much input configurations have been used and correlate them with output results.
This paper presents, starting from the basis of a previous work [15] [16] [18], an integrated environment in which both analogue stimuli and output metrics can be generated in an advanced digital verification environment, allowing a top level control of the verification process together with a comprehensive methodology to verify the effect of behavioural models into the whole system. Since automotive is one of the application fields where mixed-signal verification is very important, the proof-of-concept of the proposed approach is applied to the verification of a CAN-bus transceiver.
The next section shortly describes a theoretical method for complex mixed-signal verification, the third one gives an overview of the mixed verification environment, the fourth one describes some basic elements for data transfer from digital to analogue domain and vice-versa, the fifth one shows a real application based on CAN Controller/Transceiver scenario and the sixth one highlights some conclusions.

2. FAULT COVERAGE FOR MIXED SIGNAL VERIFICATION

A complete theoretical description of the proposed methodology is beyond the scope of this paper. However, the main concepts of it are presented in this section, together with a flow chart explaining the verification procedure.
The basic items for the verification of a mixed signal design (Design Under Test) are the following:

- Stimulate "as much as possible" the DUT and collect the useful information to verify its behaviour towards a defined set of specifications (bugs discovering), named ***phase 1*** in the following;
- Evaluate which input parameters have the higher probability (statistical approach) to determine a wrong behaviour (fault) in order to provide useful information for bug treatment, named ***phase 2*** in the following.

Let us define $X=(x_1, x_2, \ldots, x_n)$ as the DUT allowed input space, where $x_k(i,t)$ represents the i_{th} time-varying element of the stimulus array $x_k \in R^{mx1}$ where m is the number of the parameters used in the verification process as inputs of the DUT. For example, if three are the possible parameters variations (max, typical and min), the dimension "*n*" of the allowed input space X could be defined as:

$$n \leq 3^m \qquad (1)$$

where the equality holds if there is no correlation among the input configurations.
If we assume that the stimulus array components can be both the classical input signals (used in the real application) and the effects on circuit parameters of process variations, then it is also possible to define the generic i_{th} component of the generic stimulus array x_k as follows:

$$x_k(i,t)=p_i \qquad (2)$$

Where $i = 1..m$ represents the position of the parameter or of the classical input signals inside stimulus array. It is worth noting that p_i could assume minimum, typical and maximum values. For example, in the case of an operational amplifier, the stimulus array could be composed, besides the differential input (for instance $[A*\sin(\omega t)-Vref]$), also by the following circuit parameters: phase margin (***PM***), input offset (***Off***), DC gain (***Gdc***), slew-rate (***SL***) etc. These parameters show variations related to process spread and temperature that are part of the high-level model used during verification. In this case two of the arrays in the input space X could be the following:

$$x_k=([A_{max}*\sin(\omega_{min}t)-Vref_{typ}],PM_{min},Off_{typ},Gdc_{typ}, SL_{max}, \ldots);$$
$$x_h=([A_{typ}*\sin(\omega_{max}t)-Vref_{min}],PM_{max},Off_{min},Gdc_{max}, SL_{typ},..); \qquad (3)$$

As normally done in circuit characterization (corner analysis), only the corner values of each parameter are taken into account and this is sufficient in most of the applications. Once the acceptable stimulus space X is identified and assuming not to have information about the space of the parameters that does not cause faults, the pseudo-random generation of $x \in X$ (for example the one provided by Specman) may be regarded as a good solution in order to allow the definition of a uniform distribution for the probability of X as follows:

$$P(x_k) = 1/n \quad i = 1\ldots n \qquad (4)$$

Where $P(x_k)$ is the probability that x_k is used as DUT stimulus, and n is the X dimension.
The expected outputs are defined as both the outputs of the circuit in its application (e.g. the operational amplifier differential output) and internal probes used to increase the observability of the system (e.g. the current in the output branch of an opamp). From a theoretical point of view, it is possible to define the ideal output space Y as follows: $\forall x \in X \quad \exists y \in Y : y = F(x)$ where F is the ideal transfer function of the DUT and y(t) $\in R^{mx1}$. Instead, Y' is the set of the measured outputs of DUT (evaluated during the verification process). Generally a sub-set only of the output arrays components are used for faults detection, thus it is possible to define:

$$yre_k(h\ldots.q) = yre_k' \quad h<q \text{ and } q \leq m \qquad (5)$$

Where $yre_k \in Y'$ is the real output obtained from stimulus x_k, and $q-h+1$ is the number of sub-set parameters taken into account for faults detection. Similarly, for the ideal output $yid_k \in Y$:

$$yid_k(h\ldots..q) = yid_k' \qquad (6)$$

So, it is possible to define when a "***fault***" can be joined with a generic stimulus array:

"A generic stimulus x_k is joined with a fault, if $yre_k' \neq yid_k'$ "

Therefore it is possible to define a function $\Phi(w)$ that counts the fault events obtained at step w (in other words w is the stimuli number introduced in the DUT at a generic time), and a set X_f with defined dimension j, that contains $\Phi(w)$ stimuli that at step w are joined with a fault, and $j- \Phi(w)$ zeros arrays. It is worth noting that j value is an input of the verification procedure and it has to be defined in order to be able to apply the statistical theory to X_f (i.e. "j has to be big enough"). Based on the described concepts, a set of coverage items is listed below:

1. the value of the Φ function;
2. the coverage of the whole input space X;
3. the coverage of the real/ideal output spaces Y' and Y .

Thus, item **1** is satisfied if X_f is composed by j arrays not equal to zero, item **2** if all arrays of the allowed input space have been inserted in the DUT and item **3** if the real output space is equal to the ideal one. In the following Table1 some examples are given:

Cases	item 1	item 2	item 3
a	ok	not ok	not ok
b	not ok	ok	not ok
c	not ok	ok	ok

Table1: some coverage cases

Case ***a***: a set of j faults is available for further statistical investigation able to define a set of guidelines for the re-design phase;
Case ***b***: the next step depends on Φ value, then

- if $\Phi > 0$ (but less than j), X_f represents the only DUT stimuli set producing faults. In this case, further statistical elaborations (see ***phase 2*** in the following) do not give additional information and so X_f can directly be used by the designer;

- if $\Phi = 0$, it means that there are not stimuli in input space X producing faults, but at the same time the output space is not completely covered; this means that the observability of the DUT based on the used definition for the discovered faults has to be improved: yre_k' and yid_k', previously defined in (3) and (4), have to contain more than $q-h+1$ components so that an extension of the observable output space for faults detection is achieved.

Case ***c***: if $\Phi(\tau) = 0$ ($\tau \geq n$, with $n=dim(X)$), both input and output spaces are covered and no faults are produced reaching the target of the verification.

The above steps represent ***phase 1*** and its conclusion can be:

- the end of the verification activity itself (***c*** in Table1);
- direct link with design phase (***b*** in Table1 with $\Phi > 0$)
- direct link with a statistical elaboration (***phase 2***) based on a significant set of discovered faults (***a*** in Table1).

Referring to case ***a*** in Table 1 as the result of ***phase 1***, a simplified description of ***phase 2*** is shown in the following.

Thus, assuming that X_f does not contain arrays equal to zero, a set $C = (p_{11}, \ldots p_{ik}, \ldots p_{hm})$ can be generated, which is constituted by all the parameters present in the j arrays composing the set X_f. Starting from set C, it is possible to define a set X_μ composed by arrays arranged in the same way of the stimuli present in X_f and built through all the combinations of the parameters included in set C (obviously each array can contain only one parameter variation).

For instance, if $j = 3$ and $X_f = (x_{f1}, x_{f2}, x_{f3})$:

$$X_f = \left\{ \begin{array}{l} x_{f1} = (p1typ, p2worst, p3worst) \\ x_{f2} = (p1worst, p2worst, p3typ) \\ x_{f3} = (p1worst, p2worst, p3best) \end{array} \right\} \quad (7)$$

Observing (*7*), it is simple to infer that parameter ***p2*** in its ***worst*** variation is responsible of the three faults, but, as previously discussed, the scenario is much more complicated in complex systems. However, using the method above exposed, the following sets can be obtained:

$$C=\{p1typ,p1worst,p2worst,p3worst,p3best,p3typ\} \quad (8)$$

$$X_\mu = \left\{ \begin{array}{l} x_{\mu 1} = (p1typ, p2worst, p3typ) \\ x_{\mu 2} = (p1typ, p2worst, p3worst) \\ x_{\mu 3} = (p1typ, p2worst, p3best) \\ x_{\mu 4} = (p1worst, p2worst, p3typ) \\ x_{\mu 5} = (p1worst, p2worst, p3worst) \\ x_{\mu 6} = (p1worst, p2worst, p3worst) \end{array} \right\} \quad (9)$$

Once set X_μ has been defined, it is possible to generate b sets, each one composed by ε different arrays of X_μ. It is worth noting that in the following discussion the same notations as in ***phase1*** are used, but they are referred to the sets and elements obtained by X_μ.

Thus, Φ_k can be defined as the function that counts the faults generated by the generic set X_k ($k = 1 \ldots b$), and $S_k(p_{il})$ as the function that provides the number of faults generated by the arrays containing parameter p_i with variation l ($l = best, typical, worst$).

Referring again to set X_k, if $\Phi_k \neq 0$, it is possible to define the probability that a parameter p_{il} has to produce a fault:

$$P_k(p_{il}) = \left\{ \begin{array}{l} S_k(p_{il})/(m * \Phi_k) \text{ if } \Phi_k \neq 0 \\ 0 \text{ otherwise} \end{array} \right. \quad (10)$$

where m represents the number of the components of each stimulus arrays.

The formula exposed in (10) can be obtained for all b sets, so that it is possible to calculate the average probability for each parameter variations as follows:

$$P_{av}(p_{il})=(\Sigma_k P_k(p_{il}))/b \text{ with } k=1 \ldots b \quad (11)$$

The values obtained by (11**)** represent the output of ***phase 2*** that gives information to the designer in order to understand the root cause of the faults and consequently fix the design bugs. Once the designer has improved the design, the verification can be re-started from ***phase 1***. If ***phase 1*** gives no faults with input/output spaces completely covered (***c*** in Table1) as a result, the iteration is stopped and the verification process is closed. In Fig.1 a simplified flow chart of the proposed verification approach is shown.

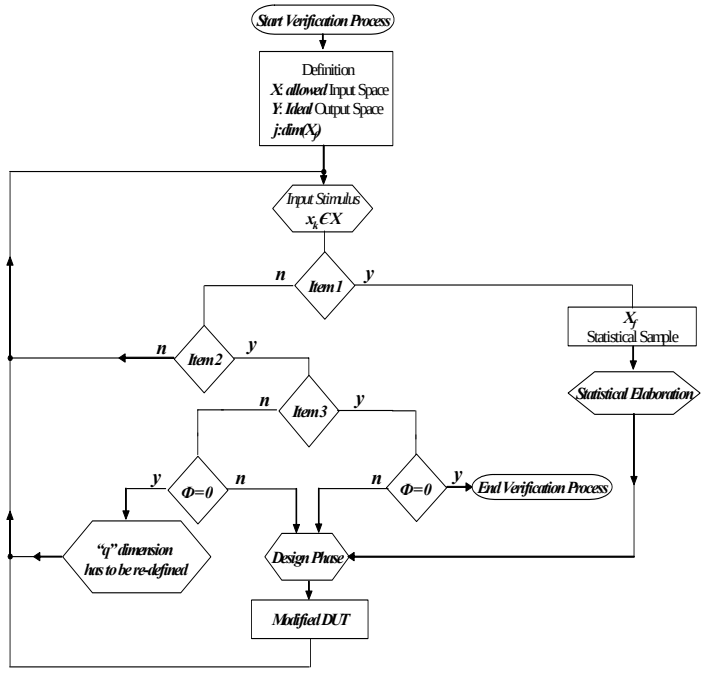

Fig.1: Simplified flow chart of the proposed verification approach

3. THE INTEGRATED MIXED-SIGNAL VERIFICATION ENVIRONMENT

The main verification engine, on which the integrated mixed-signal verification environment is based, is the Verisity's Specman Elite® (by Cadence Design System, www.cadence.com/verisity), a very powerful tool for automating the process of functional verification. Specman offers a comprehensive environment for all aspects of the verification flow: automatic generation of functional tests, data and temporal checking, functional coverage analysis, and HDL (High-Description Language) simulation control. Specman includes a powerful verification language, called "*e*" (under standardization with the IEEE initiative P1647 [9]), that allows the verification engineer to capture the rules from specifications as well as generate tests automatically.
A schematic representation of the integrated mixed-signal verification environment is shown in Fig.2.

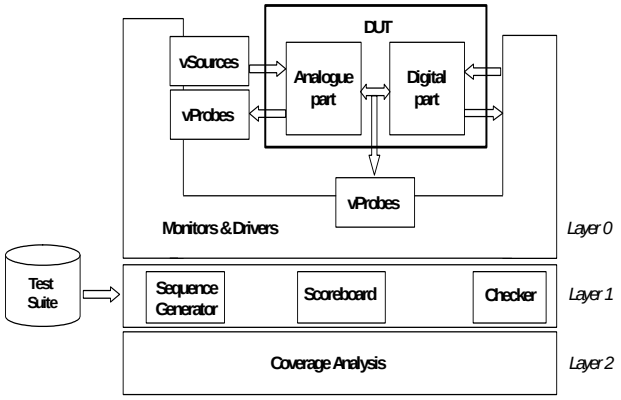

Fig.2: The integrated mixed-signal verification environment

The environment consists of a set of AHDL modules which, driven by Specman via proper *e* procedures, produce a sequence of pseudo-random inputs for the DUT reaching all its sub-systems, and provide to extract, elaborate and digitalize analogue outputs so as to make Specman checking and coverage analisys possible.

The *vSources* blocks represent "verification Sources" which are the models of a signal source to transfer digital signal from the digital verification environment to the mixed-domain simulator or to control mixed signal generators. Examples of vSources are: signal generators, noise injectors, parameters spread emulator, etc. *vProbes*, "verification Probes", are models of a signal probe to transfer analogue signals from the mixed-domain simulator to the digital Verification environment. Examples of vProbes are: voltage/current/time detectors, min/max functions, FFT and THD functions, etc. vSources and vProbes, as shown in the next paragraph, are implemented partially in "e" and partially in AHDL Analogue HDL, such the VerilogAMS [10]. Moreover, an integrated mixed-signal verification environment interface written in "e" helps the verification engineer to handle the vProbes and vSources in order to implement the desired verification scenario.

Drivers are procedures written in "e" that generate the proper set of stimuli, either directly driving the digital part or driving the analogue part through the vSources. *Monitors* receive the outputs directly or through the vProbes. In addition, the detection of the failures occurring in the analogue or the digital DUT is done through the use of a *scoreboard* that contains all the data needed to compare the expected responses with the real ones. The *Checker* controls the data coming from the monitors, to check for rules compliance for instance. A *sequence generator* is provided to handle and schedule the whole verification environment. Therefore, using all the information coming from monitors and scoreboard related to the covered input space, it is possible to elaborate a coverage function representing the compliance of the mixed system with its specifications.

4. *v*SOURCE & *v*PROBE CONCEPTS

The schematic diagram of a generic *v*Source is shown in Fig.3.

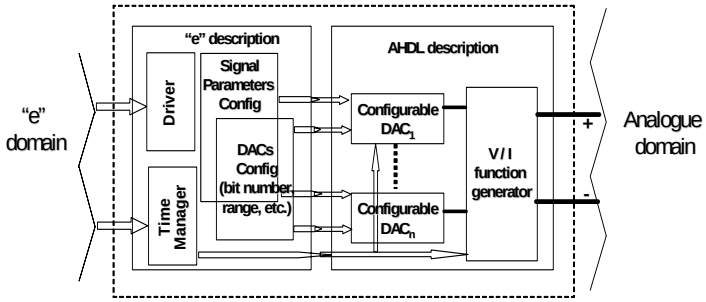

Fig.3: Schematic diagram of vSource

As shown in Fig.3, a vSource block is composed by an "e" description part and an AHDL one. The "e" part transfers three kinds of information into the AHDL domain:

- Signal parameters: in the integrated mixed-signal verification environment, the stimuli coming from "e" domain are transformed in the parameters of a function generator. For example, in order to produce a sine-wave, the following parameters are needed: frequency, amplitude, phase, delay etc. It is worth noting that the vSource output signals represent also technology parameters spread for a corner-cases analysis of an analogue DUT sub-system (as discussed in the previous section).

- Configuration signals: a configurable DACs layer converts the digital stimuli in analogue ones in a specific range and with a certain resolution (see Configurable DAC blocks in Fig.3);

- Time Manager signals: it is necessary to optimise verification/simulation time, e.g. by decoupling the digital simulator events with mixed-signal ones, or providing special time function

impossible to reproduce in the digital verification environment.

The schematic diagram of a generic vProbe is shown in Fig.4. A generic vProbe acts in the opposite way than a generic vSource: the selected analogue output signals coming from analogue DUT are transferred into the "e" domain in order to collect them in a proper way (see Monitor in Fig.4) and elaborate dynamic coverage analysis. In addition, since it is not possible to manage complex mathematical function with "e" language, some vProbes include a signal processing unit in the AHDL domain (see V/I Signal Processing block in Fig.4).

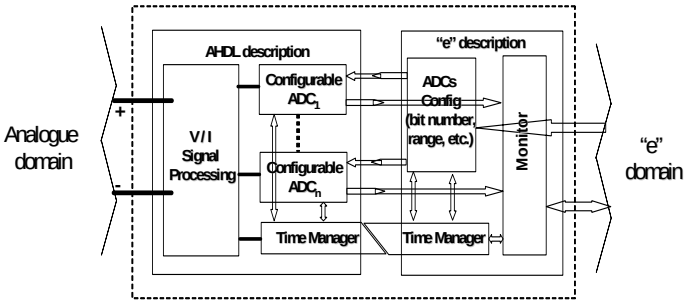

Fig.4: Schematic diagram of a vProbe.

5. CAN CONTROLLER/TRANSCEIVER CASE STUDY

A CAN Controller/Transceiver system will be used as real application scenario of the integrated mixed-signal verification environment. The analogue part of the CAN transceiver was modelled in VerilogAMS language and connected with the digital transceiver part acting as a failure detector. Thanks to the eVC ("e" Verification Component) of the CAN Controller available in YOGITECH ([12-14]), it is possible to generate a real traffic for the CAN Transceiver. This scenario is completed by a "Fault Injector" AHDL component [18] that introduces fault on the CANH/CANL bus line using a set of standard fault (ISO 11898-1). The simplified schematic diagram of CAN-based mixed-signal verification environment is shown in Fig.5.

The analogue part of the CAN transceiver has been modelled in VerilogAMS language and connected with the digital transceiver part acting as a failure detector. Fault injection is particular important since most of the electronics system require al least a noise analysis (i.e. the injection of disturb in the control lines ad the measurements of the consequent degradation of the system mission). A typical example of this is the verification of the robustness of CAN protocol units against line noise, e.g. the injection of bit errors in the packets. In this case CAN devices should be able to deal with these kinds of error and re-transmit until they are successful. At the same time they should not clog the bus with re-transmissions due to faulty or marginal designs and these must be identified before silicon goes into production. Thus, using a set of the standard faults (ISO 11898-1) that occur on the bus line is possible to recognize if the whole system is able to detect and recognize them in the proper way, and, in the case an error occurs, it is possible to recognize its nature (digital or analogue) and position.

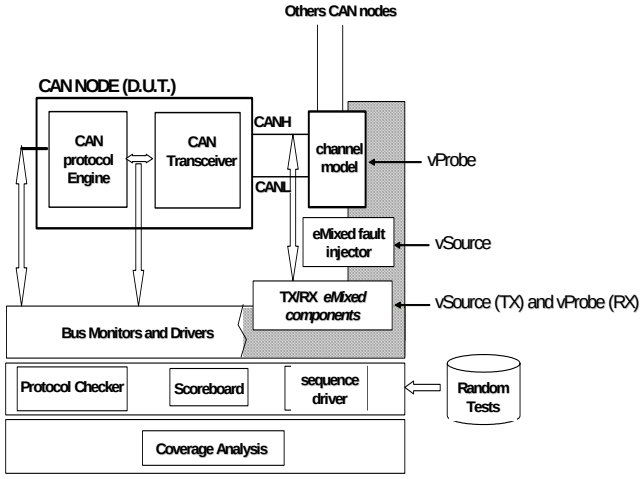

Fig.5: CAN Controller/Transceiver "integrated mixed signal verification environment" scenario

6. CONCLUSION

A standard and consolidated methodology and a common environment for the verification of mixed-signal system are not available yet: analogue/digital design and simulation activities are only linked at top-level and for smaller sub-systems. This is due to the fact that analogue simulators are orders of magnitude slower than digital simulators [5]. The end effect is that even trivial design mistakes are enough to cause serious failures of the whole system, because the users can forget to verify a particular case by using traditional test-benches. Moreover, many hypotheses have to be assumed on the analogue/digital interface.

It is proven [8] that by using AHDL models the simulations are very fast and can be very accurate if the non ideal effects are modelled in a proper way [7]. Based on that, the integrated mixed-signal verification environment proposed in this paper is an answer to the previous mentioned open issues: for instance, it can both randomise the analogue sub-system non ideal effects as in a classical fully analogue transistor level simulation approach, and also produce a coverage analysis based on covered input/output spaces and analogue/digital coverage items.

In summary, this approach links analogue and digital worlds in a common environment to help the verification engineer to produce more test cases than traditional approaches, in a faster way, minimizing overlap simulation and in a virtually independent way from the designer point of view.

In the near future, the proposed mixed signal verification scenario could include the devices and

packages failure models, so that the failure rate and qualification time will be both reduced.

REFERENCES

[1] M.Edwards, W.Rosenstiel, M.Witerholer, T.Oppold, C.Schilz-Key, Y.Kashai, T.Kuhn, "Object Oriented Hardware Synthesis and Verification", *Proc. of the 14th International Symposium on System Synthesis, 2001, Montreal, Canada, pp. 189-194*

[2] Bill Luo, Jim Lear, "A Unified Functional Verification Approach for Mixed Analog-Digital ASIC Designs", *Legerty INC, DesignCon 2003*

[3] Koji Ara, Kei Suzuki, Hitachi, Ltd, "A Proposal for Transaction-Level Verification with Component Wrapper Language", *Proc. of Design Automation and Test in Europe (DATE), March 03-07 2003, Munich, Germany*

[4] "IEEE Std VHDL 1076.1-99: The Analog and Mixed Signal Extension for VHDL", 1999

[5] C.Brown, "Mastering of Mixed-Signal Verification" EETIMES, 17 March, 2004

[6] B.Li, L.Jia, H.Tenhunen, "Optimization of analog modelling and Simulation", *Proc. of the 5th International Conference on Solid-State and Integrated Circuit Technology, 1999, Beijing, China, pp.385-388*

[7]A.J. Ginés, E. Peralías, A. Rueda, N. Martínez Madrid and R. Seepold, "A Mixed-Signal Design Reuse Methodology Based on Parametric Behavioural Models with Non-Ideal Effects", *Proc. of Design Automation and Test in Europe (DATE), March 2002, Paris, France*

[8] E. Peralías, A. J. Acosta, A. Rueda, J. L. Huertas, "A VHDL- based Methodology for the Design and Verification of Pipeline A/D Converters", *Proc. of Design Automation and Test in Europe* (DATE), March 2000, Paris, France, pp. 534-538

[9] IEEE 1647 web site: http://www.ieee1647.org/index.html

[10] Verilog-AMS Language Reference Manual: "Analog & Mixed-Signal Exstension to Verilog HDL", version 2.1, January 20, 2003

[11] "CAN Specification, version 2.0", Robert Bosch, 1991

[12] A.Di Blasi, F.Colucci, R.Mariani, "Y-CAN Platform: A Re-usable Platform for Design, Verification and Validation of CAN-Based Systems On a Chip", ETS-2003 Symposium, May2003

[13] C.Turner, C.Mueller, "Re-usable CAN IP block", CAN newsletter 3/2002 CIA

[14] eVC CAN datasheet, YOGITECH SpA, 2002

[15] R.Mariani, M.Chiavacci, G.Bonfini "Foundamentals of a novel approach for mixed analogue-digital verification", 9th IEEE European Test Symposium, Informal Session, Ajaccio (Corsica), 23-26 May 2004

[16]G.Bonfini, M.Chiavacci, F.Colucci, F.Gronchi, R.Mariani. E.Pescari, A.Sterpin, "An Integrated Mixed-Signal Verification Environment", International Test Conference (ITC) 2005, submitted

[17] Zoubir, A.M. and Boashash, B. (1998), "The bootstrap and its application in signal processing", IEEE Signal Processing Magazine, Vol.15, pp.56-76

[18] G.Bonfini, M.Chiavacci, F.Colucci, F.Gronchi, R.Mariani, E.Pescari, A.Sterpin, "Fault Coverage in A New Mixed-Signal Verification Approach", International Mixed-Signals Testing Workshop 2005 (IMSTW'05), 27-29 June, Cannes, France.