Analog Integrated Circuits and Signal Processing, 39, 177–190, 2004

Behavioral Fault Modeling and Simulation Using VHDL-AMS to Speed-Up Analog Fault Simulation

Y. KILIÇ[1] AND M. ZWOLIŃSKI[2]

[1]*Allegro MicroSystems Europe Ltd., Stuart House, Station Road, Musselburgh, EH21 7PB, UK*
[2]*Department of Electronic and Computer Science, University of Southampton, Highfield, Southampton, SO17 1BJ, UK*
E-mail: yavuz.kilic@ieee.org; mz@ecs.soton.ac.uk

Received ; Revised November 2, 2002; Accepted June 11, 2003

Au: Pls. provide received date.

Abstract. One of the main requirements for generating test patterns for analog and mixed-signal circuits is fast fault simulation. Analog fault simulation is much slower than the digital equivalent. This is due to the fact that digital circuit simulators use less complex algorithms compared with transistor-level simulators. Two of the techniques to speed up analog fault simulation are: fault dropping/collapsing, in which faults that have similar circuit responses compared with the fault-free circuit response and/or with another faulty circuit response are considered equivalent; and behavioral/macro modeling, whereby parts of the circuit are modeled at a more abstract level, therefore reducing the complexity and the simulation time. This paper discusses behavioral fault modeling to speed-up fault simulation for analog circuits.

Key Words: behavioral fault modeling, macro modeling, analog fault simulation, VHDL-AMS

1. Introduction

As transistor sizes shrink, integrated circuits (ICs) have been growing in size and functionality. This growth in IC complexity causes testing to become much more difficult. For digital circuits the problem of testing can be simplified by using standard fault models and fast fault simulation. Faults in digital circuits can be modeled as stuck-at, bridging, delay and open faults. These structural faults can then be used to generate functional test patterns. The objective of a test program for digital circuits translates into determining whether or not a fault exists using the smallest possible number of test patterns [1].

A test pattern is evaluated by looking at its fault coverage. All faults detected with a pattern can be *dropped* from further consideration. Fault simulation is done to assess the fault coverage. There are a number of fault simulation techniques for digital circuits. *Serial fault simulation* is perhaps the simplest method. For each fault, a "faulty" copy of the circuit with that fault inserted is created. Then, all the faulty copies of the circuits along with the fault-free one are simulated with the test pattern. If the output of a faulty circuit differs from the fault-free output, that fault is considered to be detectable.

Another fault simulation technique for digital circuits is *concurrent fault simulation* [2]. The differences between the faulty and fault-free circuit behaviors might be relatively small. Therefore, in concurrent fault simulation the aim is to avoid redundant element evaluation when the faulty and fault-free behaviors are the same hence reducing the computational effort.

Analog and mixed-signal fault simulation has been limited to the serial technique. Faster digital fault simulation methods are not easily applied to analog circuits and/or mixed-signal circuits, because faults do not affect the behavior of circuit nodes in a binary manner.

One way to speed-up fault simulation for analog and mixed-signal circuits is to use behavioral/macro models, where parts of the circuit are modeled at a more abstract level, reducing the complexity and hence the simulation time. Characterizing behavioral fault models requires low-level simulations and is therefore not applicable in every case. There are three situations in which behavioral fault modeling might be of benefit, however. First, if circuit blocks were reused, low-level fault simulations would not have to be repeated.

Second, at the system design stage, information about possible faults and how their effects might be propagated can be used to insert optimal test structures. Third, actual faults can be modeled at a low level (analog or digital), while neighboring circuit blocks can be modeled behaviorally, again reducing the overall simulation time.

In this paper behavioral fault simulation for analog CMOS circuits is investigated. The structure of the rest of the paper is as follows. First, macro modeling for analogue circuits is presented. Then behavioral modeling is discussed with a case study. In Section 4, behavioral modeling using Hardware Description Languages (HDLs) is summarized. In Section 5, a behavioral fault model is developed in VHDL-AMS [3] for an opamp circuit operating in inverting amplifier configuration and the model is simulated using the *hAMSter* VHDL-AMS simulator [4]. Simulation results are given in Section 6. Finally, in Section 7 some conclusions are drawn.

2. Macromodels for Analogue Circuits

Simulation at the transistor level for analog circuits is computationally very expensive. One way to reduce this high simulation cost is to partition a large analog circuit into smaller functional blocks such as opamps (operational amplifiers) and to replace each functional block with its *macromodel* or to describe each block using mathematical equations (a *behavioral model*). This solution is sometimes called *hierarchical fault simulation* [5].

The word *macromodel* usually refers to a compact representation of a circuit that captures those features that are useful for a particular purpose while discarding redundant information [6]. Macromodels developed for SPICE-like simulators are basically electrical networks containing devices such as voltage-controlled voltage sources, instead of the full transistor network, and with fewer nodes than the original circuit.

Many circuits are designed in a modular style, in which functional units are connected to achieve design specifications. The behavior of the whole circuit is determined by how the individual units interact with each other, while what happens inside each is unimportant in terms of capturing the input-output relationship for the entire circuit. The accuracy of a macromodel must, therefore, be defined in terms of how closely its input-output behavior matches that of the original unit [6].

Since the early 1970s, a number of macromodels have been developed mainly for integrated operational amplifier circuits (opamps) [5, 7]. Boyle et al. presented a macromodel for integrated bipolar opamp circuits [8]. This macromodel was six times less complex (in terms of the node count) than the original opamp circuit, and the simulation time was an order of magnitude faster than the device-level model.

The derivation of component values for the Boyle macromodel is not, however, straightforward. Some parameters are modeled using unbalanced input devices and other parameters interact. Therefore, a modular approach was suggested [9], in which a macromodel was derived simply from the published data sheets. Individual parameters were modeled separately and the results combined to provide the output response. Since the parameters were separated they did not interact and only those required were included.

Recent research has focused on how to capture the effect of a fault in an analogue circuit within its macromodel [1, 3, 10]. The fault macromodeling problem was formulated in terms of deriving the macro parameter set, B, based on the performance parameter set, P (gain, the bandwidth, samples on the frequency or time response curves, etc.) of the transistor-level faulty circuit [5]. The accuracy of the macromodel was evaluated by checking the consistency of the performance parameter set, P, between the transistor-level circuit and the macromodel.

Two steps are needed to obtain the macromodel for a functional block within an analog circuit [5]:

1. Perform transistor level fault simulation for each faulty circuit to obtain the value of the performance parameter set P.
2. Map each performance parameter set P to the corresponding macro parameter set, B. This is referred to as *parameter mapping*.

It was assumed that the transistor-level fault list is given and the macromodel structure and the performance parameter set, P, to be matched are predetermined by the circuit designer.

There are several ways to do parameter mapping. One simple approach is based on analytical design equations that express the macro parameter set, B, as analytical functions of the performance parameter set, P, and the value of B is derived by function evaluation. As analog ICs get more complex, this approach is becoming more difficult. Another simple approach

is to build an empirical mapping function, $B = F(P)$, based on a large number of data pairs (P, B), referred to as the *training set* [5]. Usually the training set is generated by randomly selecting M out of the N performance parameter sets for the faulty circuits obtained by transistor-level simulation and then the value of the macro parameter set B for each selected P is derived. The derivation of each data pair usually requires multiple runs of the macromodel-level simulation [3].

Macromodeling in general and fault macromodeling using SPICE-like languages in particular have, nevertheless, been shown to be very difficult [1, 5–20]. Therefore, another easier and perhaps more efficient way of modeling analog circuits at a higher level is necessary.

3. Behavioral Modeling

A behavioral model describes a circuit block in terms of mathematical equations modeling the functionality of the block, for example, in terms of the input-output relationship. Behavioral modeling has been used for speeding up analog simulation in general [21] and analog fault simulation in particular [1, 10, 20, 22]. In one approach, analog circuits were modeled behaviorally in the C programming language [21]. Broyden's method [23] was used to formulate and solve the model equations in a custom simulator. The main drawback of this work is that since the technique does not require derivatives it cannot be used for small-signal analysis.

Chang et al. [10] presented a behavioral fault model derived from a macromodel of a CMOS operational amplifier from the IEEE Mixed-Signal Benchmark Suite [24] (Fig. 1). The "faulty" macromodel was developed using DC-sweep analysis. The DC behavior of the benchmark opamp operating in inverting, non-inverting and unity gain amplifier configurations was first investigated under different fault conditions, as shown in Fig. 2. Single transistor catastrophic faults, bridging/short and nearly open faults, and parametric faults with W (channel width), L (channel length) and V_{TH} (threshold voltage) varied by $\pm 10\%$ were used for each transistor. Then an attempt was made to group the different faulty behaviors. By comparing the fault-free offset voltage measured at the inputs of the opamp operating in one of the three configurations with the equivalent faulty circuits, four different equivalent fault types were derived [10]: M4 drain-to-gate short (Type I), M5 drain-to-source short (Type II), M7 drain open (Type III), and M5 drain-to-source short (Type IV). The first three fault types were found for the opamp operating in the inverting configuration, where the Type IV fault group was found for the non-inverting configuration.

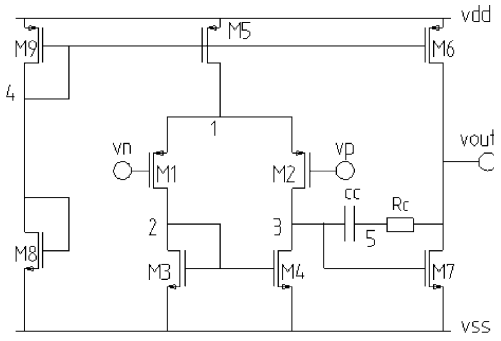

Fig. 1. The 2-stage CMOS Miller opamp used in [10] for behavioral fault modeling.

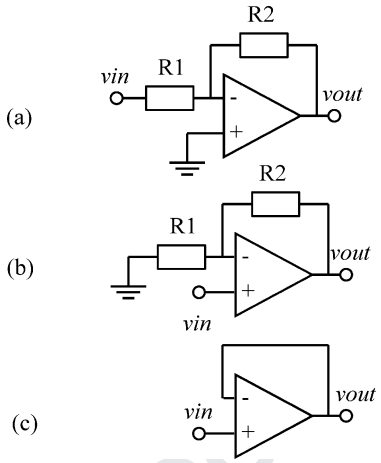

Fig. 2. Three different configurations used in [10] for the benchmark circuit given in [24]: (a) Inverting amplifier, (b) non-inverting amplifier, and (c) unity gain buffer.

The input offset voltage (measured between the positive and negative inputs of the opamp in the closed-loop configurations) and the output voltage versus the input voltage for the fault-free opamp operating in three configurations were determined by simulation. Our HSPICE simulations of these configurations are shown in Figs. 3–5, respectively.

HSPICE simulations of the input offset voltage and the output voltage for each fault group with respect to the input voltage are shown in Figs. 6–9, respectively.

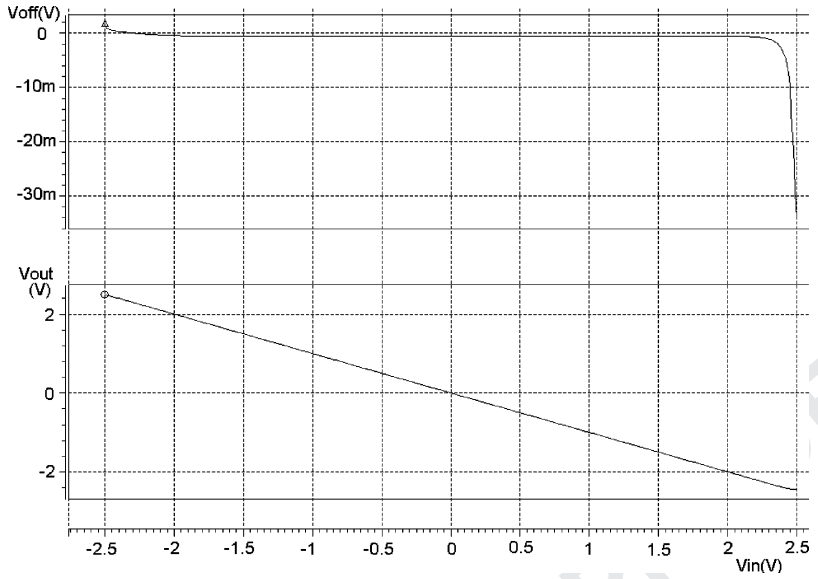

Fig. 3. Input offset voltage and output voltage versus DC-sweep input voltage for the fault-free inverting amplifier.

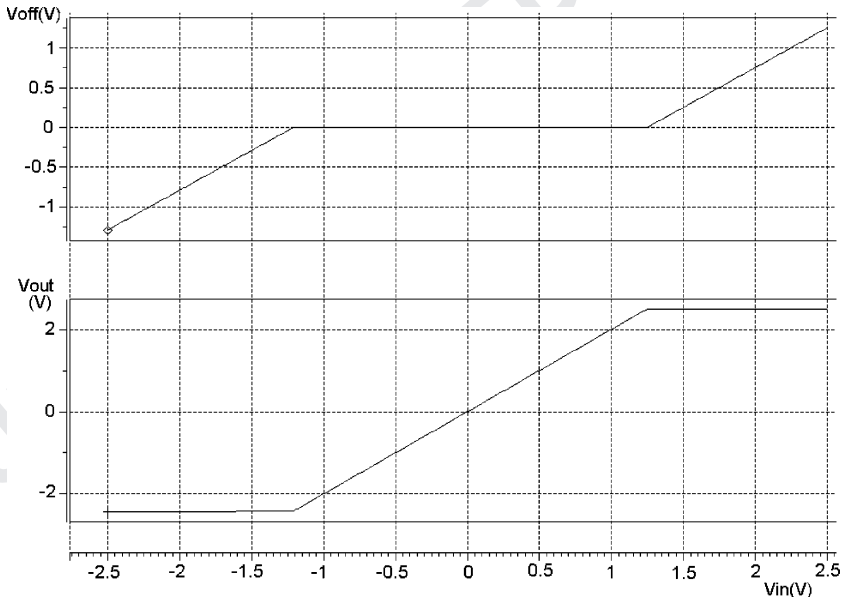

Fig. 4. Input offset voltage and output voltage versus DC-sweep input voltage for the fault-free non-inverting amplifier.

As can be seen from Figs. 7 and 9, the output responses of Type II and Type IV faults are quite similar to the fault-free responses given in Figs. 3 and 4. Type II and Type IV input offset voltages are noticeably different from the fault-free responses. The input offset voltage has a small DC level for Type II faults, but has a non-linear characteristic for Type IV faults.

The remaining two faults have very different characteristics to the fault-free equivalents for both input offset voltages and output voltages. It can be concluded from the figures that a Type I fault causes the inverting amplifier output to be "nearly stuck-at" a negative voltage near to the negative supply voltage level. A Type III fault causes the inverting amplifier output to have a

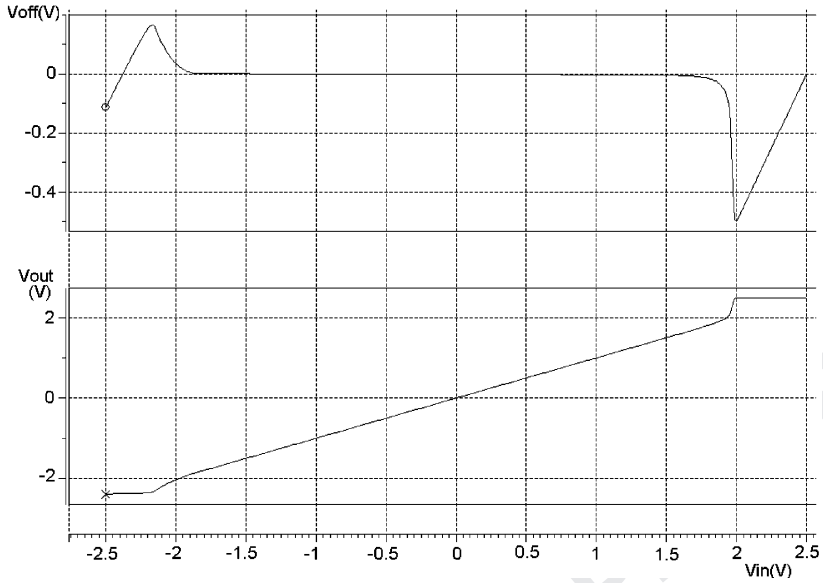

Fig. 5. Input offset voltage and output voltage versus DC-sweep input voltage for the fault-free unity gain buffer.

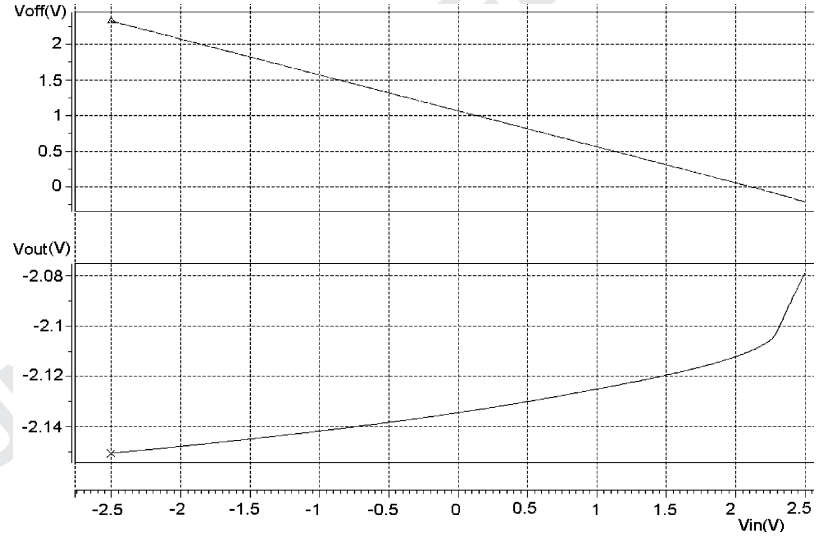

Fig. 6. Input offset voltage and the output voltage for the Type I fault (M4 drain-to-gate short fault for the inverting amplifier configuration).

non-inverting characteristic for the negative values of the DC input signal, and an inverting characteristic for the positive values of the DC input signal. As can be seen from the figures, the input offset voltage at the inputs of the opamp has a linear characteristic for Type I faults, and a piecewise linear characteristic for Type III faults.

The macromodel given in Fig. 10 for the inverting opamp can be used to derive the input output relationship under fault conditions [10]:

$$V_{\text{out}} = A_{\text{CL}}[(1+m)V_{\text{in}} + k] \tag{1}$$

where A_{CL} is the closed-loop gain for the opamp, the parameters m and k are given in [10] as:

$$m = \frac{-R2}{D + R2} \tag{2}$$

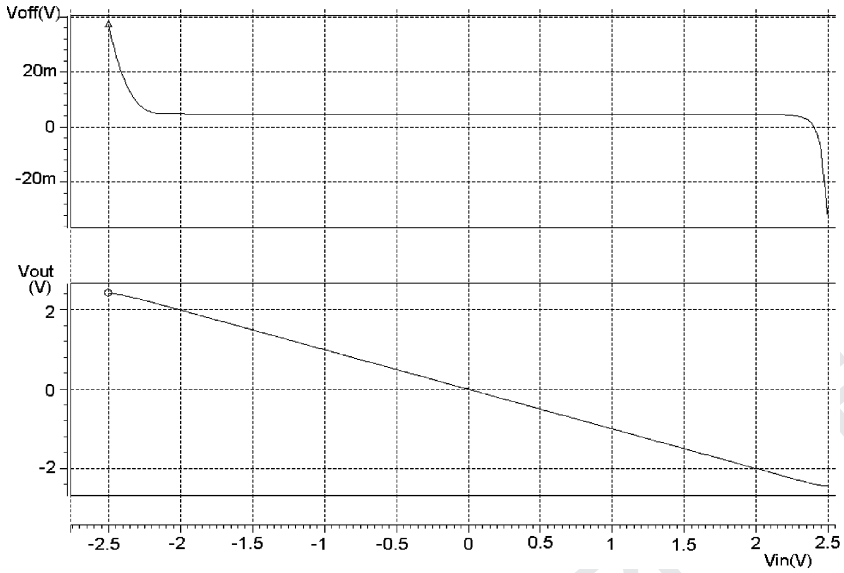

Fig. 7. Input offset voltage and the output voltage for the Type II fault (M5 drain-to-source short fault for the inverting amplifier configuration).

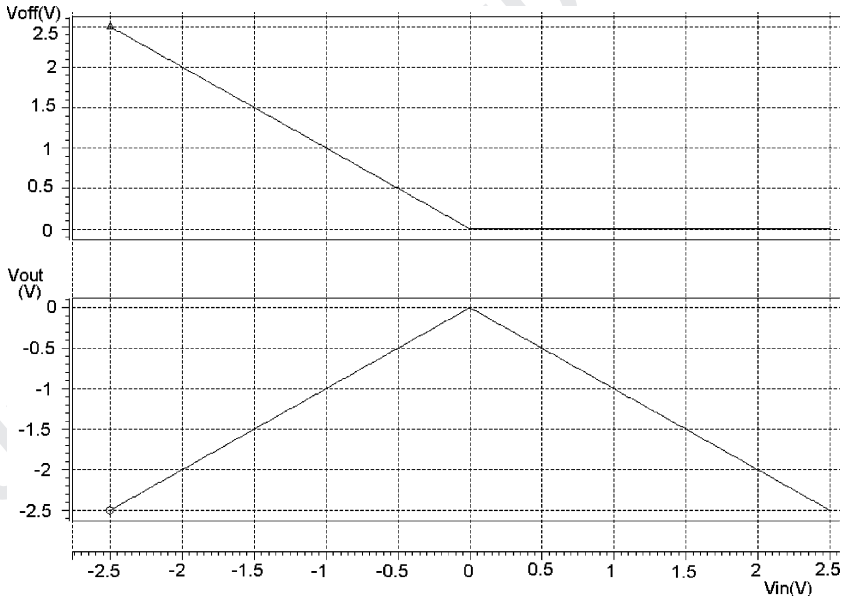

Fig. 8. Input offset voltage and the output voltage for the Type III fault (M6 open drain fault for the inverting amplifier configuration).

and

$$k = aVos + bVdd + cVss \tag{3}$$

where

$$D = B(R2//Ro//Rdd//Rss),$$

$$B = \left(\frac{A}{Ro} - \frac{1}{R2}\right)(Rid//R1//R2//2Ricm),$$

$$a = \frac{R2//D}{A_{CL}(R1//R2//2Ricm//BR2)},$$

$$b = \frac{SF}{A_{CL}Rdd},$$

$$c = -\frac{SF}{A_{CL}Rss},$$

$$A_{CL} = -\frac{R2}{R1},$$

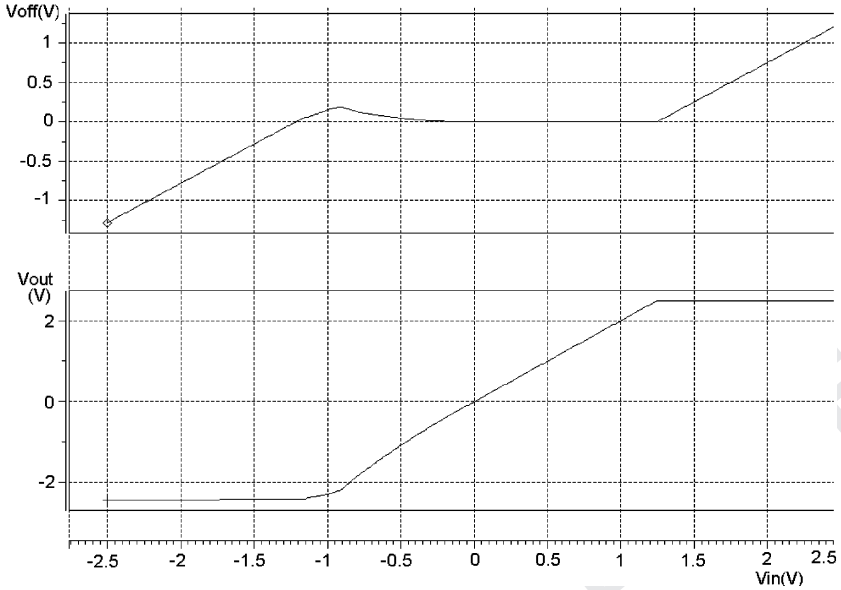

Fig. 9. Input offset voltage and the output voltage for the Type IV fault (M5 drain-to-source short fault for the non-inverting amplifier configuration).

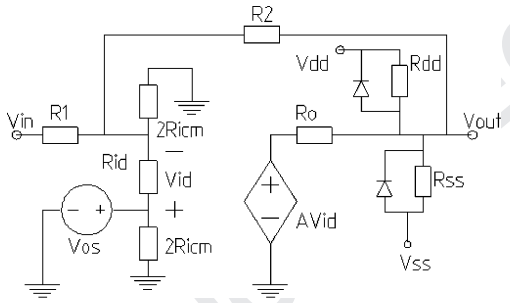

Fig. 10. Macromodel used in [10] to derive the input-output relationship for the closed loop inverting opamp.

$$SF = Rdd//Rss///Ro//(R2//R11)// \times \frac{Ro(R11//R2)}{AR11},$$
$$R11 = R1//2Ricm//Rid,$$

and A represents the open-loop gain.

The non-ideal effects such as the input offset voltage, *Vos*, the finite open-loop gain, A, and the finite input and output resistances, *Rid* (differential mode input resistance), *Ricm* (common mode input resistance), *Ro* (output resistance), and the resistances from the output node to the supply rails (*Rdd* and *Rss*) to model output stuck-at faults were taken into account when deriving Eq. (1). Note that for the fault-free case *Rid*, *Ricm*, *Rdd*, *Rss*, and A would be infinite, *Vos*, and *Ro* would be zero, hence $m \to 0$, and $k \to 0$. When a fault causes the output to be stuck-at some voltage level, $D \to 0$, therefore $m \to -1$, and k is the value of the stuck-at output voltage; the closed-loop gain, A_{CL}, is assumed to be unity. As they are dealt with elsewhere [10], the derivation of the above equations will not be given here.

The current limiting effect was also modeled in [10]. This is due to the finite supply voltage at the output of the opamp. It is claimed that the model covers all the parametric faults and 92.5% of the catastrophic faults that were considered. The model could not model the M4 drain-to-gate short, M5 drain-to-source short, M1 open-gate faults for the non-inverting amplifier and the M2 drain-to-gate short, M4 drain-to-gate short, M5 drain-to-source short, M1 open gate, M3 open source and M5 open gate faults for the unity gain buffer.

4. Behavioral Modeling Using HDLs

HDLs have been in use for behavioral modeling and simulation of digital circuits as well as analog electronic systems, fluid concentrations in chemical processes, and even parachute jumps since 1960 [25]. Currently two of the most widely used standards for

modeling digital designs are VHDL [26], and Verilog [27]. For analogue circuits, the choice has been between SPICE and proprietary analog HDLs.

Analog HDLs support the description of systems of differential and algebraic equations (DAEs). The solution of these systems varies continuously with time. Most analog HDLs support both structural composition and conservation semantics, in addition to behavioral descriptions. Examples of such languages are FAS [28], SpectreHDL [29], and Verilog-A [30].

Mixed-signal design has depended on the use of separate HDLs for the analog and digital parts or, again, on proprietary languages. Mixed-signal languages support both event-driven techniques and DAEs in one simulator. Simulators in this category are MAST/Saber [31], VeriasHDL [31], AdvanceMS [28], hAMSter [4].

Both VHDL and Verilog have been extended to analog mixed-signal design: VHDL-AMS [3], and Verilog-AMS [30]. The analog extensions to VHDL and Verilog should alleviate the multiple-language problem [32].

Since VHDL-AMS was standardized in 1999 there has been some work done on fault modeling using VHDL-AMS. One reason for the limited progress is, perhaps, that there is not yet a robust VHDL-AMS simulator available that has all the VHDL-AMS constructs implemented, such as procedural statements. Perkins et al. attempted to use an analog VHDL for fault modeling and simulation with limited success [1]. The HDL-A modeling language with the ELDO simulator from Anacad (now a part of Mentor) was used. Behavioral model simulation using HDL-A and ELDO was over 4.6 times slower than the macromodel simulation carried out using HSPICE [1]. One of the reasons for this is that the semiconductor device models implemented in ELDO were not as efficient as those were in HSPICE.

5. VHDL-AMS Behavioral Fault Model for the Inverting Opamp

A VHDL-AMS model for the behavioral model given in Eq. (1) has been developed. The values of m and k were derived by carrying out transistor level simulations for four fault types and are given in Table 1.

Considering only the input-output relationship given in Eq. (1), the opamp macromodel given in Fig. 10 can now be simplified to that shown in Fig. 11. All the fault effects and non-ideal effects are approximated to $Fos = mV_{in} + k$, which is applied to the inverting input of the opamp.

Table 1. The values of m and k for different fault groups.

	Parameters	
Fault types	m	k (V)
Type I	−1.02	2.15
Type II	0	0.011
Type III	0 if vin > 0 V −2 if vin < 0	0
Type IV	−1 if vin > ∼1.2 V and vin < ∼−1.2 V 0 if ∼ −1.2 V < vin < ∼ 1.2 V	V_{dd}/2 if vin > ∼1.V V_{ss}/2 if vin < ∼−1.2 V 0 if ∼ −1.2 V < vin < ∼1.2 V

A VHDL-AMS implementation of the behavioral model given in Eq. (1) is shown in Figs. 12 and 13 [33]. rin represents the input resistance of the opamp in Fig. 12 where it is only used for the third equation in Fig. 13. The third equation is needed as there are three quantities declared in the architecture declaration

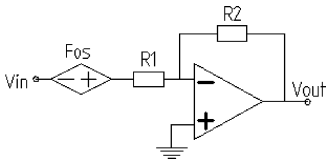

Fig. 11. Behavioral level DC-offset fault model for the inverting opamp.

```
--behavioral opamp
library disciplines;
library ieee;
use disciplines.electromagnetic_system.all;
use ieee.math_real.all;
--entity
entity op_behav is
   generic ( m : real := 0.0; --fault-free value
           k : real := 0.0; --fault-free value
           Acl: real:= -1.0; --closed-loop gain
           rin : real := 100.0e6);
   port (terminal in_node, out_node : electrical);
end;
```

Fig. 12. The VHDL-AMS entity implementation of the behavioral fault model.

shown in Fig. 13. Note that this architecture declaration also covers the supply voltage limiting effect at the output of the opamp.

In order to simulate the VHDL-AMS model shown in Figs. 12 and 13, one also needs VHDL-AMS models for a resistor, a voltage source and a testbench, which are shown in Figs. 14–16, respectively.

```
--architecture
library disciplines;
library ieee;
use disciplines.electromagnetic_system.all;
use ieee.math_real.all;
architecture behav of op_behav is
   quantity vout across iout through out_node;
   quantity vin across iin through in_node;
   quantity Fos : real;
   -- supply voltage limit
   constant v_limit : real := 2.5;
begin
procedural is
  variable vout_calc : real;
  begin
   Fos := m*vin + k;
   vout_calc := Acl * (vin + Fos);
   iin  := (vin - Fos) / rin;
   if (vout_calc > v_limit) then vout := 2.5;
   elsif (vout_calc < -v_limit) then vout := -2.5;
   else vout := vout_calc;
   end if;
end procedural;
end;
```

Fig. 13. The VHDL-AMS architecture implementation of the behavioral fault model.

```
--resistor
library disciplines;
use disciplines.electromagnetic_system.all;
entity resistor is
  generic (rnom : real := 0.0);
  port(terminal p,m  : electrical); --interface ports
end resistor;
architecture behav of resistor is
    quantity r_e across r_i through p to m;
begin
    r_i == r_e/rnom;
end behav;
```

Fig. 14. A VHDL-AMS model of a resistor.

```
-- voltage source
library disciplines;
use disciplines.electromagnetic_system.all;
--entity declaration.
entity v_source is
   generic (dc_value : real :=-2.50);
   port(terminal p,m: electrical); --interface ports
end v_source;
--architecture declaration.
architecture behav of v_source is
    quantity v_in across i_out through p to m;
begin
  v_in==dc_value*now; -- slow transient
end architecture behav;
```

Fig. 15. A VHDL-AMS model of a voltage source.

```
--testbench
library disciplines;
library ieee;
use disciplines.electromagnetic_system.all;
use ieee.math_real.all;
entity ex_op_behav is end;
architecture testbench of ex_op_behav is
  terminal in_node, out_node, vsrc : electrical;
begin
  op_behav_uut : entity op_behav (behav)
    generic map ( m => 0.0,k => 0.0, acl => -1.0)
    port map ( in_node  => in_node,
               out_node => out_node);
  vin_dc : entity v_source (behav)
  generic map ( dc_value => 5.0)
  port map ( p => vsrc, m => electrical_ground);
  rsrc : entity resistor (behav)
    generic map (rnom => 100.0)
    port map (p => vsrc, m => in_node);
end;
```

Fig. 16. A VHDL-AMS testbench used with the hAMSter simulator to simulate the behavioral model shown in Figs. 12 and 13.

Note that input voltage source in the architecture declaration shown in Fig. 15 is realized using a predefined VHDL-AMS function, *now*, which returns the value of the current time at each step as simulation proceeds. This is done in order to simulate the DC-sweep analysis, which is not defined in VHDL-AMS (unlike many SPICE-like simulators). This technique is called *slow transient* simulation.

6. Simulation Results

The slow transient simulation results using the hAMSter simulator and the behavioral closed-loop VHDL-AMS model of the inverting opamp (the fault free case) with the necessary component and voltage source models and the testbench given in the previous sections are shown in Figs. 17 and 18.

Note that the X-axis in Figs. 17 and 18 represents the time in seconds, where Y-axis represents vout, vin, and Fos in Volts. (Unless otherwise stated, for the rest of the paper it will be assumed that X-axis will represent time in seconds for the simulation results obtained using hAMSter).

Using the values for the parameters m and k from Table 1 yields the simulation results shown in Figs. 19–26.

Note that the output response of the opamp, vout, found for the positive values of vin and the negative values of vin for Type I faults (Figs. 19 and 20) are the same ("nearly stuck-at" −2.14 V), as expected.

For Type III faults *Fos* is determined using the following if-then construct in the VHDL-AMS model.

For Type IV faults *Fos* is determined using the following if-then construct in the VHDL-AMS model.

DC-sweep analysis cannot be performed for VHDL-AMS. Therefore, the transient simulation results for different fault types using the VHDL-AMS behavioral models and the hAMSter simulator were compared with the transient simulation results obtained using transistor level models with HSPICE simulator. To do that a sine wave with 2 V peak-to-peak magnitude and 1 KHz frequency was applied to both behavioral and transistor level circuits. The simulators were run

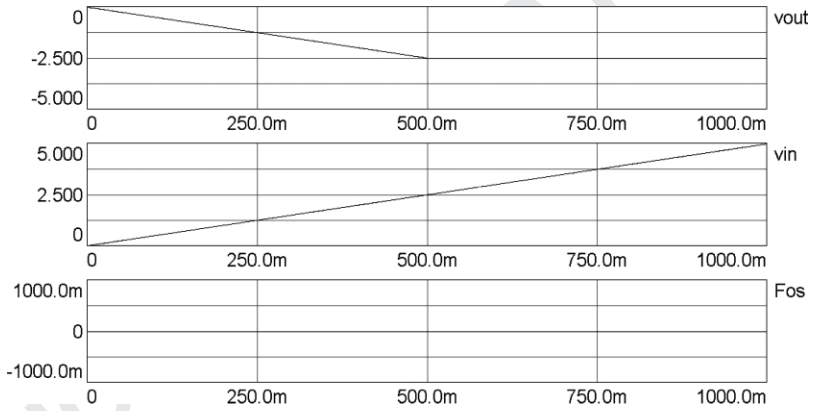

Fig. 17. Slow-transient simulation results using the VHDL-AMS model with hAMSter for the positive values of the input voltage source.

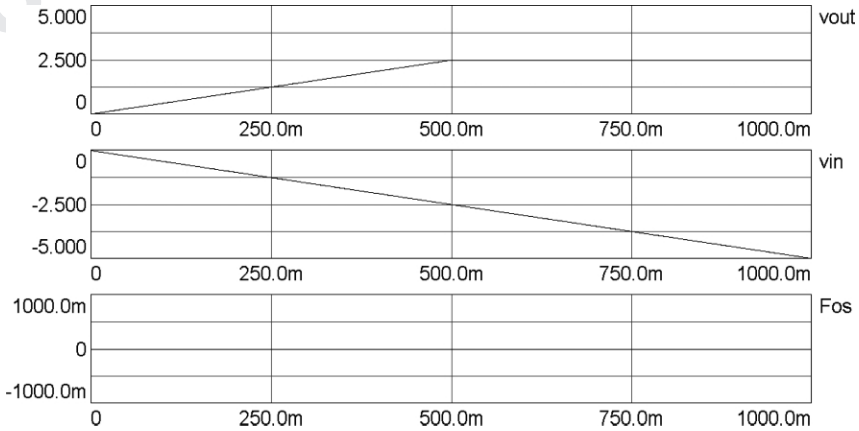

Fig. 18. Slow-transient simulation results using the VHDL-AMS model with hAMSter for the negative values of the input voltage source.

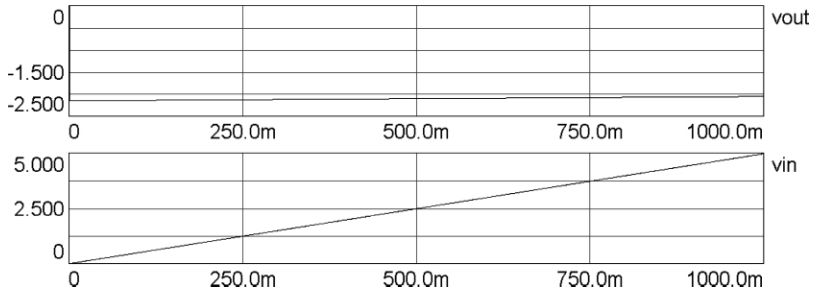

Fig. 19. Slow-transient simulation using hAMSter for Type I faults for the positive values of vin.

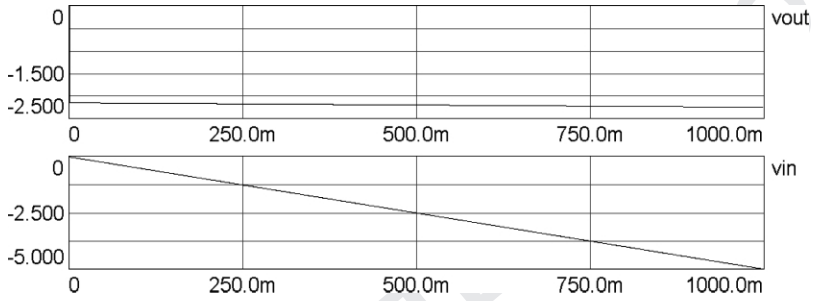

Fig. 20. Slow-transient simulation using hAMSter for Type I faults for the negative values of vin.

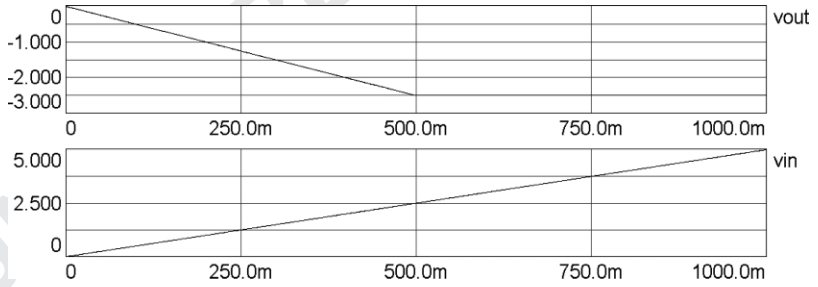

Fig. 21. Slow-transient simulation using hAMSter for Type II faults for the positive values of vin.

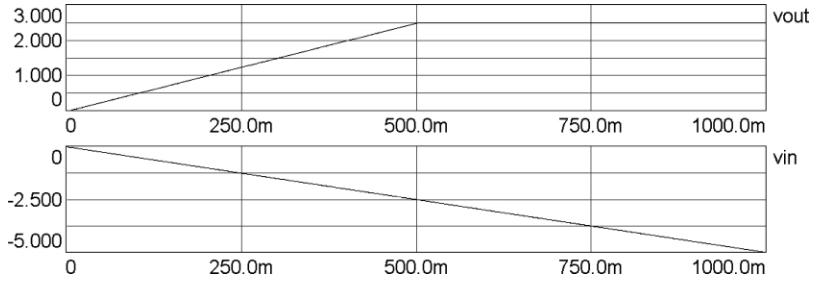

Fig. 22. Slow-transient simulation using hAMSter for Type II faults for the negative values of vin.

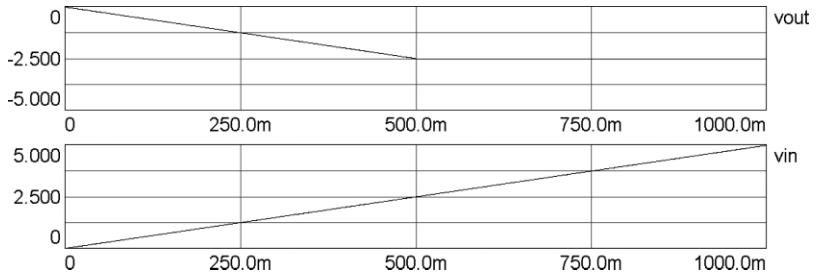

Fig. 23. Slow-transient simulation using hAMSter for Type III faults for the positive values of vin.

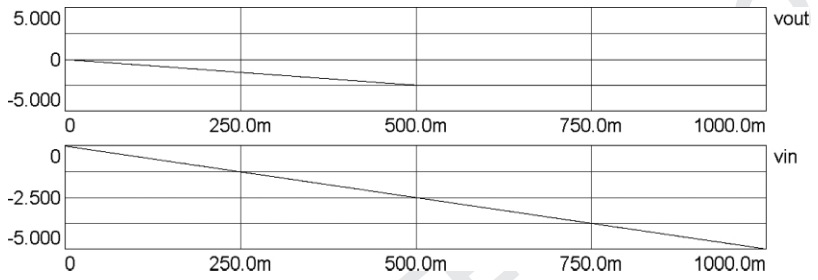

Fig. 24. Slow-transient simulation using hAMSter for Type III faults for the negative values of vin.

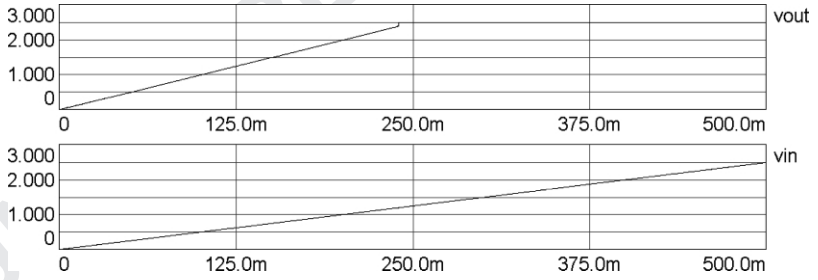

Fig. 25. Slow-transient simulation using hAMSter for Type IV faults for the positive values of vin.

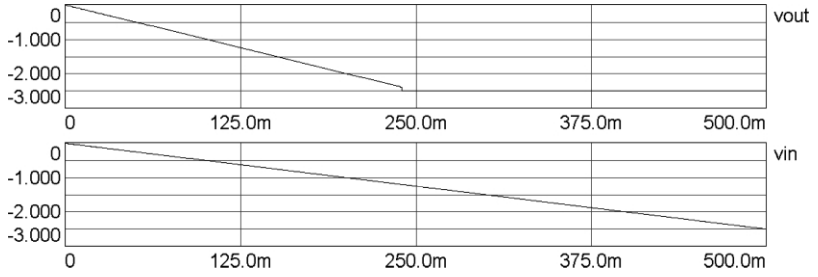

Fig. 26. Slow-transient simulation using hAMSter for Type IV faults for the negative values of vin.

Table 2. Comparison of CPU times for transistor level transient HSPICE simulations against VHDL-AMS behavioral level hAMSter simulations.

	The CPU time (s) Simulator	
Fault type	hAMSter	HSPICE
Fault I	90 m	400 m
Fault II	90 m	360 m
Fault III	100 m	37.37
Fault IV	140 m	350 m

```
if (vin < 0.0) then
    Fos := -2.0*vin;
else
    Fos := 0.0;
end if;
```

Au: Pls. cite Fig. 27 in the text.

Fig. 27. if-then construct implemented in the VHDL-AMS model for Type III faults.

```
if (vin > 1.2) then
   Fos := -vin + 1.25;
elsif (vin < -1.2) then
   Fos := -vin - 1.25;
else
   Fos := 0.0;
end if;
```

Fig. 28. If-then construct implemented in the VHDL-AMS model for Type IV faults.

for 5 ms with 10 μs iteration steps. Table 2 shows the CPU time spent for each case with the different approaches.

As can be seen from the table there is an average 4.4 times speed-up for Fault I, Fault II cases. The speed-up for the Type III faults between the behavioral and the transistor level simulations is 373.7 times.

The reason why the behavioral model is so much faster than the transistor level for Type III faults is that Type III faults are open drain faults and HSPICE struggles with an incompletely defined circuit. Finally the speed-up for Type IV faults is around 2.5 times. The behavioral model for Type IV faults is relatively slow compared to other behavioral models due to evaluation of the if-then construct required in the procedural statement (Fig. 28) to model the Type IV faults.

7. Conclusions

Capturing circuit behavior under faulty conditions at a higher level using mathematical equations (behavioral modeling) is somewhat simpler than the macromodel approach.

Analog fault simulation is a key factor in analog/mixed-signal test generation. Currently such fault simulation is of limited use due to the speed of analog simulation and the large number of faults to be simulated. Simulation can be speeded up by using number of techniques. Behavioral modelling is one of those techniques. We have shown in this paper how one can increase analog fault simulation speed by using behavioral models. We have used VHDL-AMS for the behavioral modelling. It is clear that as VHDL-AMS simulators become more powerful it will be easier to model analog/mixed-signal circuits at a higher level so as to speed-up simulation in general and analog fault simulation in particular.

References

1. A.J. Perkins, M. Zwolinski, C.D. Chalk, and B.R. Wilkins, "Fault modeling and simulation using VHDL-AMS." *Analog Integrated Circuits and Signal Processing*, vol. 16, pp. 141–155, pp. 53–67, 1998.
2. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.
3. www.eda.org/analog
4. http://www.hamster-ams.com/
5. C.-Y. Pan and K.-T. Cheng, "Fault macro modeling for analog/mixed-signal circuits," in *IEEE International Test Conference, ITC'97*, 1997, pp. 913–922.
6. G. Casinovi and A. Sangiovanni-Vincentelli, "A macromodeling algorithm for analog circuits." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 2, pp. 150–160, 1991.
7. B. Perez et al., "A new nonlinear time-domain op-amp macromodel using threshold functions and digitally controlled network elements." *IEEE Journal of Solid-State Circuits*, vol. 23, no. 4, pp. 959–971, 1988.
8. G.A. Boyle, D.O. Pederson, B.M. Cohn, and J.E. Solomon, "Macromodeling of integrated circuit operational amplifiers." *IEEE J. of Solid State Circuits*, vol. SC-9, pp. 353–363, 1974.
9. M.E. Brinson and D.J. Faulkner, "Modular SPICE macromodel for operational amplifiers." *IEE Proc.- Circuits Devices Syst.*, vol. 141, no. 5, pp. 417–420, 1994.
10. Y.-J. Chang et al., "A behavior-level fault model for the closed-loop operational amplifier." *Journal of Information Science and Engineering*, vol. 16, no. 5, pp. 751–766, 2000.

Au: Pls. verify page range.

11. C. Chalk and M. Zwolinski, "Macromodel of CMOS operational amplifier including supply current variation." *Electronics Letters*, vol. 31, pp. 1398–1400, 1995.
12. P. Mandal and V. Visvanathan, "Macromodeling of the AC characteristics of CMOS op-amps," in *IEEE 1993 Conference on Computer Aided Design, Digest of Technical Papers*, 1993, pp. 334–339.
13. M.E. Brinson and D.J. Faulkner, "A SPICE noise macromodel for operational amplifiers." *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 42, no. 3, pp. 166–168, 1995.
14. G. Krajewska and F.E. Holmes, "Macromodeling of FET/bipolar operational amplifiers." *IEEE Journal of Solid-State Circuits*, vol. SC-14, no. 6, pp. 1083–1087, 1979.
15. C. Turchetti and G. Masetti, "A macromodel for integrated all-MOS operational amplifiers." *IEEE Journal of Solid-State Circuits*, vol. SC-18, pp. 389–394, 1983.
16. M.E. Brinson and D.J. Faulkner, "SPICE macromodel for operational amplifier power supply current sensing." *Electronics Letters*, vol. 30, no. 23, pp. 166–168, 1994.
17. R.V. Peic, "Simple and accurate nonlinear macromodel for operational amplifiers." *IEEE Journal of Solid-State Circuits*, vol. 26, no. 6, pp. 896–899, 1991.
18. B. Al-Hashimi, "Behavioural simulation of filters." *IEE Colloquium on Analogue Simulation: The Dream & The Nightmare*, pp. 51–55, 1995.
19. A.I. Kayssi and K.A. Sakallah, "Macromodel simplification using dimensional analysis," in *1994 Int. Symp. on Circuits and Systems*, 1994, pp. 335–338.
20. M. Zwolinski, Z.R. Yang, and T.J. Kazmierski, "Using robust adaptive mixing for statistical fault macromodelling." *IEE Proceedings: Circuits, Devices and Systems*, vol. 147, no. 5, pp. 265–270, 2000.
21. G. Casinovi, "Multi-level simulation of large analog systems containing behavioral models." *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 13, no. 11, pp. 1391–1399, 1994.
22. E. Bruls et al., "Analogue fault simulation in standard VHDL." *IEE Proc. Circuits Devices Syst.*, vol. 143, no. 6, pp. 380–385, 1996.
23. C.G. Broyden et al., "A class of methods for solving nonlinear simultaneous equations," *Mathematics of Computation*, vol. 19, no. 92, pp. 577–593, 1965.
24. http://www.ee.washington.edu/mad/benchmarks/benchmarks.html
25. E. Christen and K. Bakalar, "VHDL-AMS, A hardware description language for analog applications." *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 46. no. 10, pp. 1263–1272, 1999.
26. VHDL Language Reference Manual, IEEE Standard 1076–1993.
27. Standard Description Language Based on the Verilog™ Hardware Description Language, IEEE Standard 1364–1995.
28. www.mentor.com
29. www.cadence.com
30. www.verilog.com
31. www.analogy.com
32. www.ednmag.com
33. Y. Kilic, "Testing techniques and fault simulation for analogue CMOS integrated circuits." University of Southampton, PhD Thesis, Chapter 6, 2001.

Au: Pls. verify Journal Name

Yavuz Kılıç gained his B.Sc. (Hons) (electronics and telecommunications), M.Sc. (microelectronics) from Yıldız Technical University of Istanbul (Turkey), and Ph.D. from the University of Southampton (UK) in 1994, 1996, and 2001, respectively. He is currently with Allegro MicroSystems Europe Ltd., Scotland (UK) as a senior design engineer. Prior to this, he has held positions with Cyan Technology, Cambridgeshire (UK) and Philips Semiconductors, Southampton (UK) as a design engineer, and at the University of Southampton (UK) and Yıldız Technical University (Turkey) as a Research Assistant. His current interests include mixed-signal CMOS/BiCMOS integrated circuit design, simulation, test, and power management related IC design. He is a member of the IEEE Circuits and Systems Society.

Mark Zwolinski received the B.Sc. and Ph.D. degrees in electronics from the University of Southampton, UK, in 1982 and 1986, respectively. He is a Senior Lecturer in the Department of Electronics and Computer Science, University of Southampton, UK. His research interests include behavioral synthesis, design for test, fault simulation and VHDL. He is chair of the IEEE DATC subcommittee on fault simulation. He has co-authored about 100 research papers in technical journals and conferences.