



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

Y. KILIÇ¹ AND M. ZWOLIŃSKI²

¹Allegro MicroSystems Europe Ltd., Stuart House, Station Road, Musselburgh, EH21 7PB, UK

²Department 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.

40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

64 Second, at the system design stage, information about
65 possible faults and how their effects might be prop-
66 agated can be used to insert optimal test structures.
67 Third, actual faults can be modeled at a low level (ana-
68 log or digital), while neighboring circuit blocks can be
69 modeled behaviorally, again reducing the overall sim-
70 ulation time.

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

84 2. Macromodels for Analogue Circuits

85 Simulation at the transistor level for analog circuits is
86 computationally very expensive. One way to reduce
87 this high simulation cost is to partition a large analog
88 circuit into smaller functional blocks such as opamps
89 (operational amplifiers) and to replace each functional
90 block with its *macromodel* or to describe each block us-
91 ing mathematical equations (*a behavioral model*). This
92 solution is sometimes called *hierarchical fault simula-*
93 *tion* [5].

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

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

Since the early 1970s, a number of macromodels 111
have been developed mainly for integrated operational 112
amplifier circuits (opamps) [5, 7]. Boyle et al. presented 113
a macromodel for integrated bipolar opamp circuits [8]. 114
This macromodel was six times less complex (in terms 115
of the node count) than the original opamp circuit, and 116
the simulation time was an order of magnitude faster 117
than the device-level model. 118

The derivation of component values for the Boyle 119
macromodel is not, however, straightforward. Some pa- 120
rameters are modeled using unbalanced input devices 121
and other parameters interact. Therefore, a modular ap- 122
proach was suggested [9], in which a macromodel was 123
derived simply from the published data sheets. Individ- 124
ual parameters were modeled separately and the results 125
combined to provide the output response. Since the pa- 126
rameters were separated they did not interact and only 127
those required were included. 128

Recent research has focused on how to capture the 129
effect of a fault in an analogue circuit within its macro- 130
model [1, 3, 10]. The fault macromodeling problem 131
was formulated in terms of deriving the macro param- 132
eter set, B , based on the performance parameter set, 133
 P (gain, the bandwidth, samples on the frequency or 134
time response curves, etc.) of the transistor-level faulty 135
circuit [5]. The accuracy of the macromodel was evalu- 136
ated by checking the consistency of the performance 137
parameter set, P , between the transistor-level circuit 138
and the macromodel. 139

Two steps are needed to obtain the macromodel for 140
a functional block within an analog circuit [5]: 141

1. Perform transistor level fault simulation for each 142
faulty circuit to obtain the value of the performance 143
parameter set P . 144
2. Map each performance parameter set P to the cor- 145
responding macro parameter set, B . This is referred 146
to as *parameter mapping*. 147

It was assumed that the transistor-level fault list is given 148
and the macromodel structure and the performance pa- 149
rameter set, P , to be matched are predetermined by the 150
circuit designer. 151

There are several ways to do parameter mapping. 152
One simple approach is based on analytical design 153
equations that express the macro parameter set, B , as 154
analytical functions of the performance parameter set, 155
 P , and the value of B is derived by function evalua- 156
tion. As analog ICs get more complex, this approach 157
is becoming more difficult. Another simple approach 158

159 is to build an empirical mapping function, $B = F(P)$,
 160 based on a large number of data pairs (P, B) , referred
 161 to as the *training set* [5]. Usually the training set is
 162 generated by randomly selecting M out of the N per-
 163 formance parameter sets for the faulty circuits obtained
 164 by transistor-level simulation and then the value of the
 165 macro parameter set B for each selected P is derived.
 166 The derivation of each data pair usually requires multi-
 167 ple runs of the macromodel-level simulation [3].
 168 Macromodeling in general and fault macromodeling
 169 using SPICE-like languages in particular have, nev-
 170 ertheless, been shown to be very difficult [1, 5–20].
 171 Therefore, another easier and perhaps more efficient
 172 way of modeling analog circuits at a higher level is
 173 necessary.

174 **3. Behavioral Modeling**

175 A behavioral model describes a circuit block in terms
 176 of mathematical equations modeling the functionality
 177 of the block, for example, in terms of the input-output
 178 relationship. Behavioral modeling has been used for
 179 speeding up analog simulation in general [21] and ana-
 180 log fault simulation in particular [1, 10, 20, 22]. In one
 181 approach, analog circuits were modeled behaviorally in
 182 the C programming language [21]. Broyden’s method
 183 [23] was used to formulate and solve the model equa-
 184 tions in a custom simulator. The main drawback of this
 185 work is that since the technique does not require deriva-
 186 tives it cannot be used for small-signal analysis.

187 Chang et al. [10] presented a behavioral fault model
 188 derived from a macromodel of a CMOS operational
 189 amplifier from the IEEE Mixed-Signal Benchmark
 190 Suite [24] (Fig. 1). The “faulty” macromodel was de-

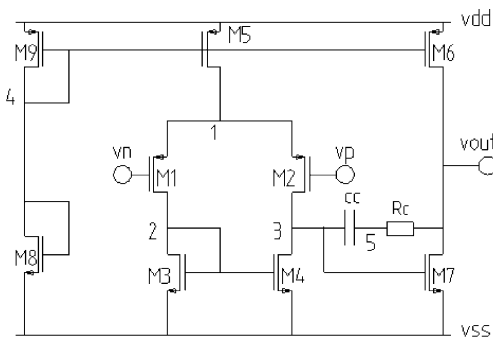


Fig. 1. The 2-stage CMOS Miller opamp used in [10] for behavioral fault modeling.

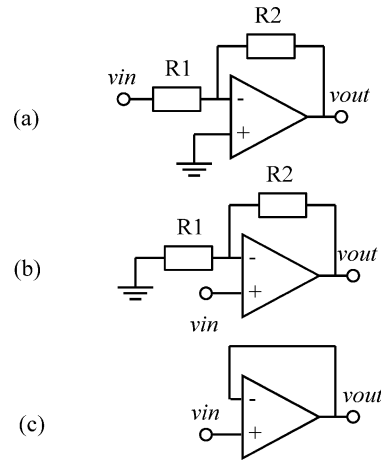


Fig. 2. Three different configurations used in [10] for the benchmark circuit given in [24]: (a) Inverting amplifier, (b) non-inverting amplifier, and (c) unity gain buffer.

veloped 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 paramet-
 ric 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 equiv-
 alent 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.

The input offset voltage (measured between the posi-
 tive 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.

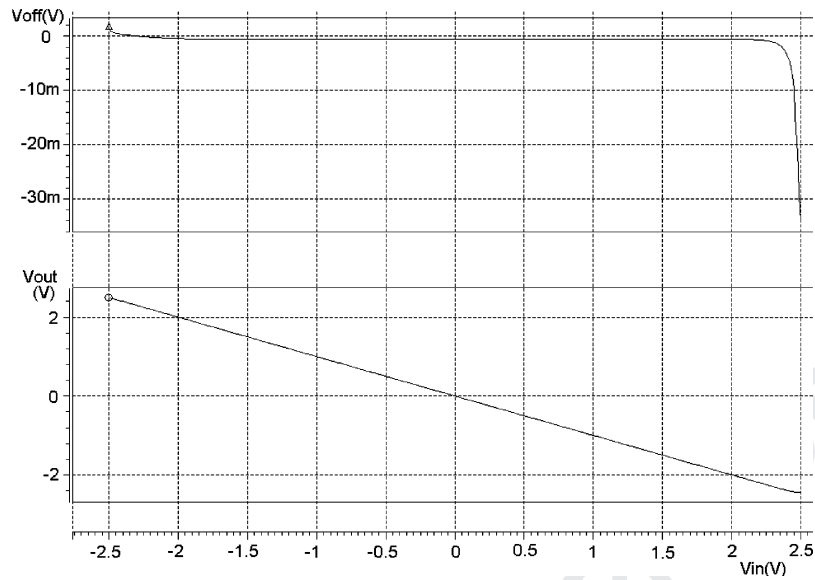


Fig. 3. Input offset voltage and output voltage versus DC-sweep input voltage for the fault-free inverting amplifier.

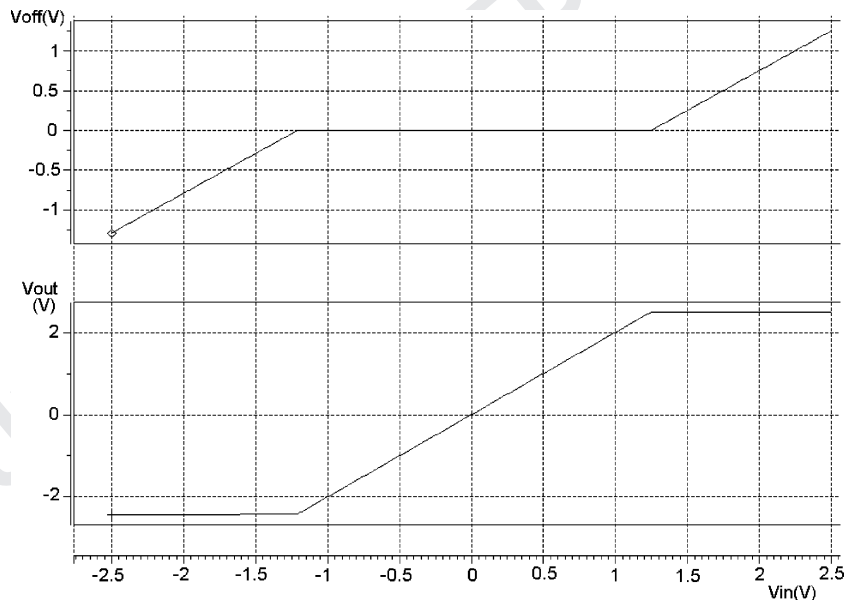


Fig. 4. Input offset voltage and output voltage versus DC-sweep input voltage for the fault-free non-inverting amplifier.

221 As can be seen from Figs. 7 and 9, the output re-
 222 sponses of Type II and Type IV faults are quite sim-
 223 ilar to the fault-free responses given in Figs. 3 and 4.
 224 Type II and Type IV input offset voltages are notice-
 225 ably different from the fault-free responses. The input
 226 offset voltage has a small DC level for Type II faults,
 227 but has a non-linear characteristic for Type IV faults.

The remaining two faults have very different char- 228
 229 acteristics to the fault-free equivalents for both input
 230 offset voltages and output voltages. It can be concluded
 231 from the figures that a Type I fault causes the inverting
 232 amplifier output to be “nearly stuck-at” a negative volt-
 233 age near to the negative supply voltage level. A Type III
 234 fault causes the inverting amplifier output to have a

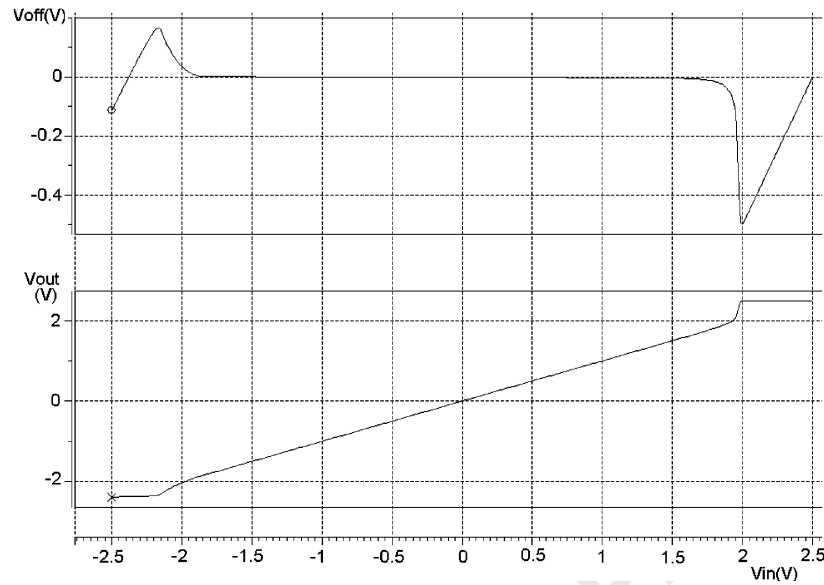


Fig. 5. Input offset voltage and output voltage versus DC-sweep input voltage for the fault-free unity gain buffer.

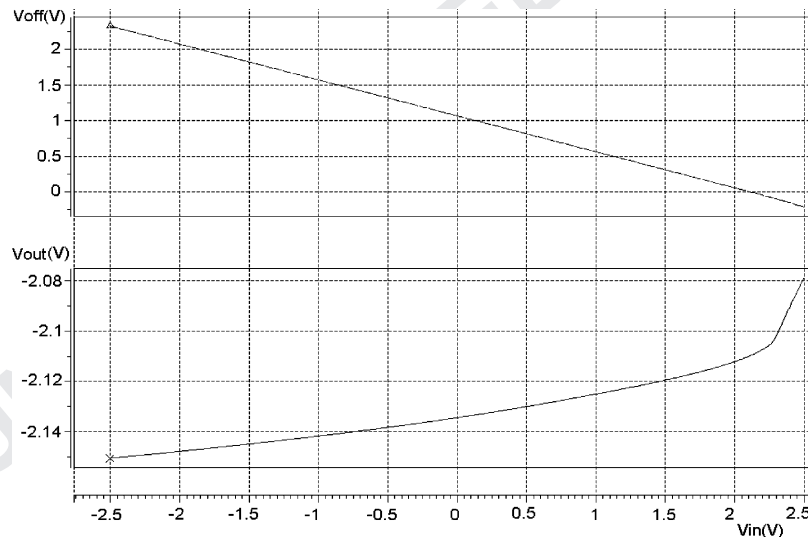


Fig. 6. Input offset voltage and the output voltage for the Type I fault (M4 drain-to-gate short fault for the inverting amplifier configuration).

235 non-inverting characteristic for the negative values of
 236 the DC input signal, and an inverting characteristic for
 237 the positive values of the DC input signal. As can be
 238 seen from the figures, the input offset voltage at the in-
 239 puts of the opamp has a linear characteristic for Type I
 240 faults, and a piecewise linear characteristic for Type III
 241 faults.

242 The macromodel given in Fig. 10 for the inverting
 243 opamp can be used to derive the input output relation-

ship under fault conditions [10]: 244

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

where A_{CL} is the closed-loop gain for the opamp, the 245
 parameters m and k are given in [10] as: 246

$$m = \frac{-R2}{D + R2} \quad (2)$$

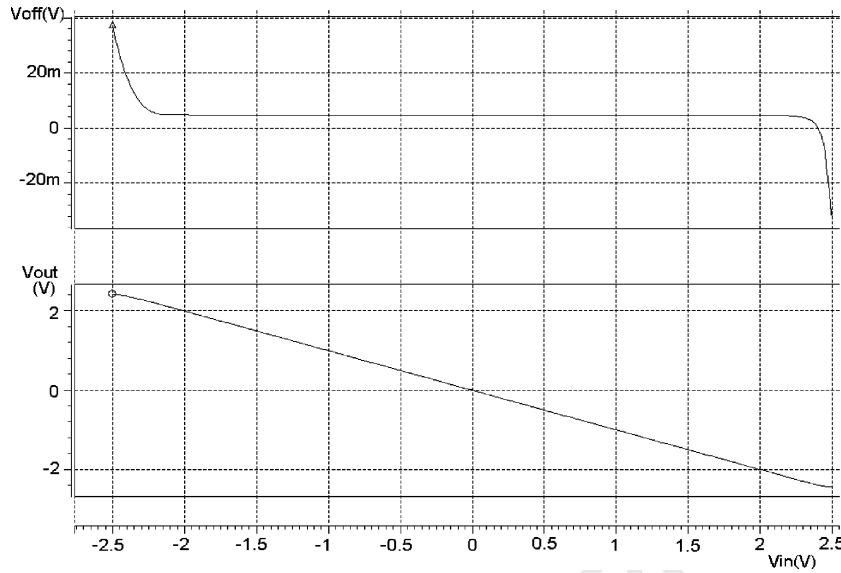


Fig. 7. Input offset voltage and the output voltage for the Type II fault (M5 drain-to-source short fault for the inverting amplifier configuration).

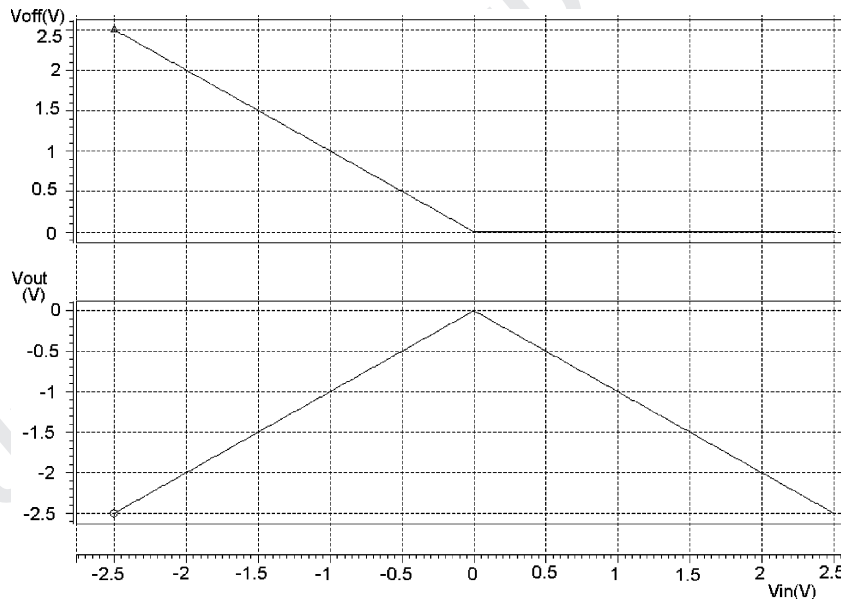


Fig. 8. Input offset voltage and the output voltage for the Type III fault (M6 open drain fault for the inverting amplifier configuration).

247 and

$$k = aV_{os} + bV_{dd} + cV_{ss} \quad (3)$$

248 where

$$D = B(R_2 // R_o // R_{dd} // R_{ss}),$$

$$B = \left(\frac{A}{R_o} - \frac{1}{R_2} \right) (R_{id} // R_1 // R_2 // 2R_{icm}),$$

$$a = \frac{R_2 // D}{A_{CL}(R_1 // R_2 // 2R_{icm} // BR_2)},$$

$$b = \frac{SF}{A_{CL}R_{dd}},$$

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

$$A_{CL} = -\frac{R_2}{R_1},$$

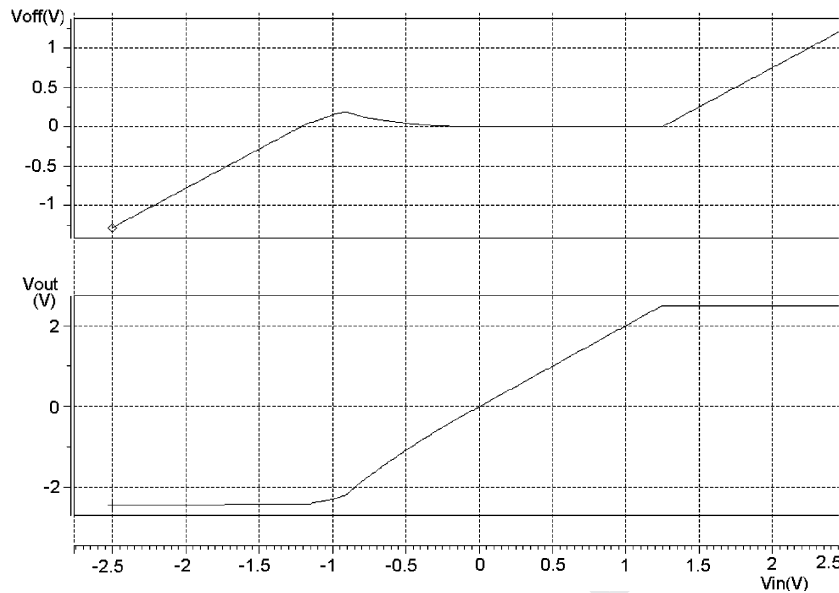


Fig. 9. Input offset voltage and the output voltage for the Type IV fault (M5 drain-to-source short fault for the non-inverting amplifier configuration).

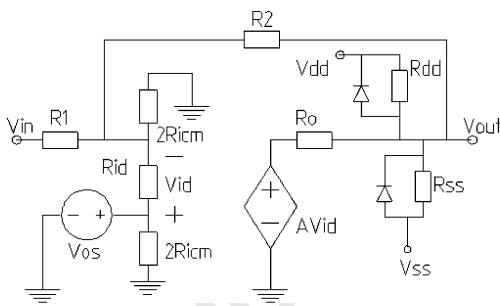


Fig. 10. Macromodel used in [10] to derive the input-output relationship for the closed loop inverting opamp.

$$SF = Rdd // Rss // Ro // (R2 // R11) // \frac{Ro(R11 // R2)}{AR11},$$

$$R11 = R1 // 2R_{icm} // R_{id},$$

249 and A represents the open-loop gain.

250 The non-ideal effects such as the input offset voltage,
 251 V_{os} , the finite open-loop gain, A , and the finite input
 252 and output resistances, R_{id} (differential mode input re-
 253 sistance), R_{icm} (common mode input resistance), R_o
 254 (output resistance), and the resistances from the output
 255 node to the supply rails (R_{dd} and R_{ss}) to model output
 256 stuck-at faults were taken into account when deriving

Eq. (1). Note that for the fault-free case R_{id} , R_{icm} , 257
 R_{dd} , R_{ss} , and A would be infinite, V_{os} , and R_o would 258
 be zero, hence $m \rightarrow 0$, and $k \rightarrow 0$. When a fault causes 259
 the output to be stuck-at some voltage level, $D \rightarrow 0$, 260
 therefore $m \rightarrow -1$, and k is the value of the stuck-at 261
 output voltage; the closed-loop gain, A_{CL} , is assumed 262
 to be unity. As they are dealt with elsewhere [10], the 263
 derivation of the above equations will not be given here. 264

The current limiting effect was also modeled in [10]. 265
 This is due to the finite supply voltage at the output of 266
 the opamp. It is claimed that the model covers all the 267
 parametric faults and 92.5% of the catastrophic faults 268
 that were considered. The model could not model the 269
 M4 drain-to-gate short, M5 drain-to-source short, M1 270
 open-gate faults for the non-inverting amplifier and the 271
 M2 drain-to-gate short, M4 drain-to-gate short, M5 272
 drain-to-source short, M1 open gate, M3 open source 273
 and M5 open gate faults for the unity gain buffer. 274

4. Behavioral Modeling Using HDLs

275

HDLs have been in use for behavioral modeling and 276
 simulation of digital circuits as well as analog elec- 277
 tronic systems, fluid concentrations in chemical pro- 278
 cesses, and even parachute jumps since 1960 [25]. 279
 Currently two of the most widely used standards for 280

281 modeling digital designs are VHDL [26], and Verilog
282 [27]. For analogue circuits, the choice has been between
283 SPICE and proprietary analog HDLs.

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

291 Mixed-signal design has depended on the use of sep-
292 arate HDLs for the analog and digital parts or, again, on
293 proprietary languages. Mixed-signal languages support
294 both event-driven techniques and DAEs in one simula-
295 tor. Simulators in this category are MAST/Saber [31],
296 VeriasHDL [31], AdvanceMS [28], hAMSter [4].

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

302 Since VHDL-AMS was standardized in 1999 there
303 has been some work done on fault modeling using
304 VHDL-AMS. One reason for the limited progress is,
305 perhaps, that there is not yet a robust VHDL-AMS sim-
306 ulator available that has all the VHDL-AMS constructs
307 implemented, such as procedural statements. Perkins
308 et al. attempted to use an analog VHDL for fault model-
309 ing and simulation with limited success [1]. The HDL-
310 A modeling language with the ELDO simulator from
311 Anacad (now a part of Mentor) was used. Behavioral
312 model simulation using HDL-A and ELDO was over
313 4.6 times slower than the macromodel simulation car-
314 ried out using HSPICE [1]. One of the reasons for
315 this is that the semiconductor device models imple-
316 mented in ELDO were not as efficient as those were in
317 HSPICE.

318 5. VHDL-AMS Behavioral Fault Model 319 for the Inverting Opamp

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

324 Considering only the input-output relationship given
325 in Eq. (1), the opamp macromodel given in Fig. 10 can
326 now be simplified to that shown in Fig. 11. All the fault
327 effects and non-ideal effects are approximated to $Fos =$

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

Fault types	Parameters	
	m	k (V)
Type I	-1.02	2.15
Type II	0	0.011
Type III	0 if $v_{in} > 0$ V -2 if $v_{in} < 0$	0
Type IV	-1 if $v_{in} > \sim 1.2$ V and $v_{in} < \sim -1.2$ V 0 if ~ -1.2 V < $v_{in} < \sim 1.2$ V	$V_{dd}/2$ if $v_{in} > \sim 1$.V $V_{ss}/2$ if $v_{in} < \sim -1.2$ V 0 if ~ -1.2 V < $v_{in} < \sim 1.2$ V

$mV_{in} + k$, which is applied to the inverting input of the
opamp.

A VHDL-AMS implementation of the behavioral
model given in Eq. (1) is shown in Figs. 12 and 13
[33]. r_{in} 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

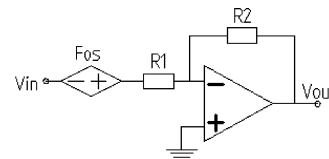


Fig. 11. Behavioral level DC-offset fault model for the inverting opamp.

```
--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.

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

339 In order to simulate the VHDL-AMS model shown
 340 in Figs. 12 and 13, one also needs VHDL-AMS models
 341 for a resistor, a voltage source and a testbench, which
 342 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 343
 declaration shown in Fig. 15 is realized using a pre- 344
 defined VHDL-AMS function, *now*, which returns the 345
 value of the current time at each step as simulation pro- 346
 ceeds. This is done in order to simulate the DC-sweep 347
 analysis, which is not defined in VHDL-AMS (unlike 348
 many SPICE-like simulators). This technique is called 349
slow transient simulation. 350

351 **6. Simulation Results**

352 The slow transient simulation results using the hAM-
 353 Ster simulator and the behavioral closed-loop VHDL-
 354 AMS model of the inverting opamp (the fault free case)
 355 with the necessary component and voltage source mod-
 356 els and the testbench given in the previous sections are
 357 shown in Figs. 17 and 18.

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

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

Note that the output response of the opamp, vout, 367
 found for the positive values of vin and the negative 368
 values of vin for Type I faults (Figs. 19 and 20) are the 369
 same (“nearly stuck-at” -2.14 V), as expected. 370

For Type III faults Fos is determined using the fol- 371
 lowing if-then construct in the VHDL-AMS model. 372

For Type IV faults Fos is determined using the fol- 373
 lowing if-then construct in the VHDL-AMS model. 374

DC-sweep analysis cannot be performed for VHDL- 375
 AMS. Therefore, the transient simulation results for 376
 different fault types using the VHDL-AMS behav- 377
 ior models and the hAMSter simulator were compar- 378
 ed with the transient simulation results obtained us- 379
 ing transistor level models with HSPICE simulator. To 380
 do that a sine wave with 2 V peak-to-peak magnitude 381
 and 1 KHz frequency was applied to both behavioral 382
 and transistor level circuits. The simulators were run

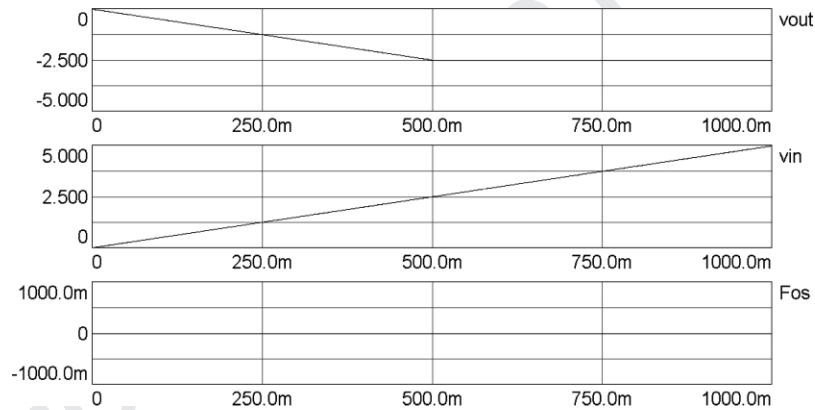


Fig. 17. Slow-transient simulation results using the VHDL-AMS model with hAMSter for the positive values of the input voltage source.

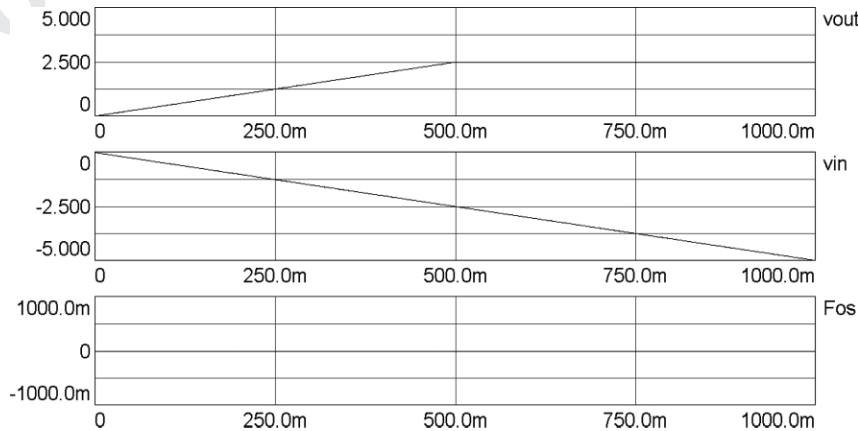


Fig. 18. Slow-transient simulation results using the VHDL-AMS model with hAMSter for the negative values of the input voltage source.

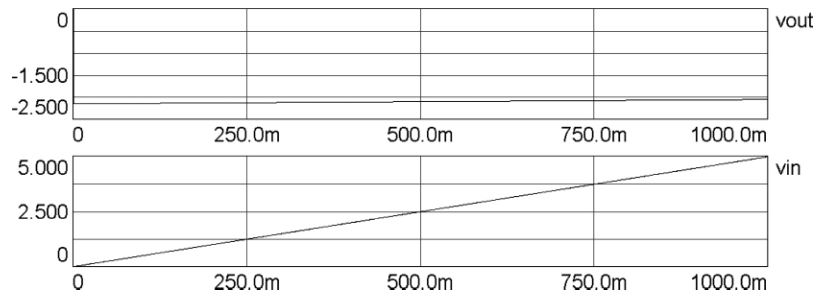


Fig. 19. Slow-transient simulation using hAMSter for Type I faults for the positive values of v_{in} .

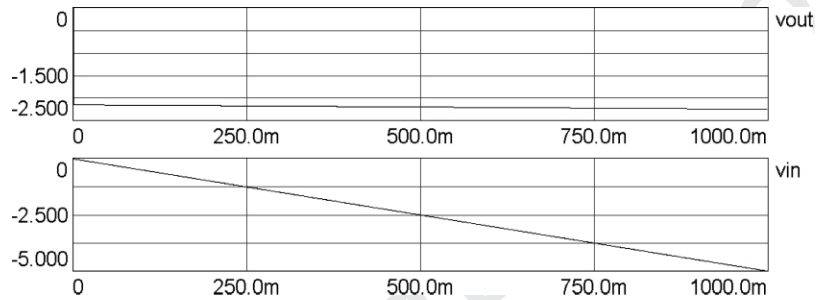


Fig. 20. Slow-transient simulation using hAMSter for Type I faults for the negative values of v_{in} .

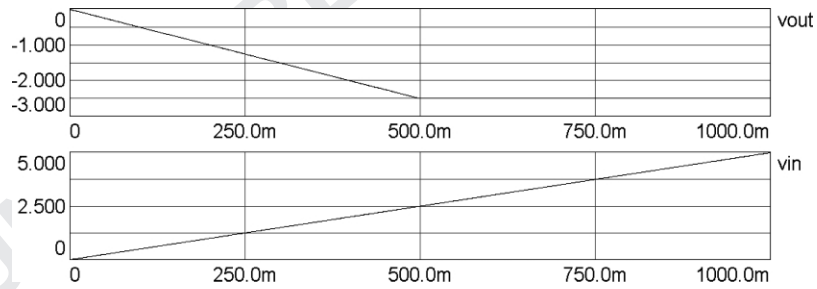


Fig. 21. Slow-transient simulation using hAMSter for Type II faults for the positive values of v_{in} .

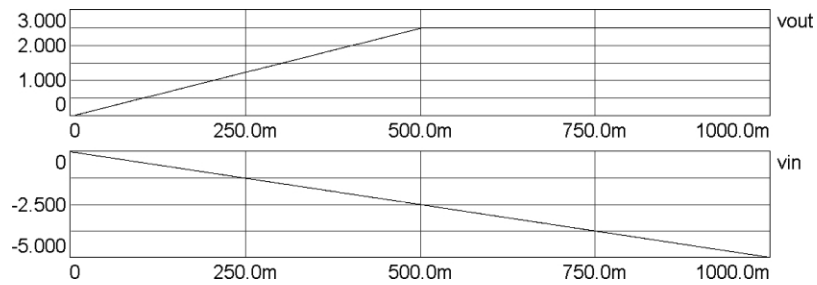


Fig. 22. Slow-transient simulation using hAMSter for Type II faults for the negative values of v_{in} .

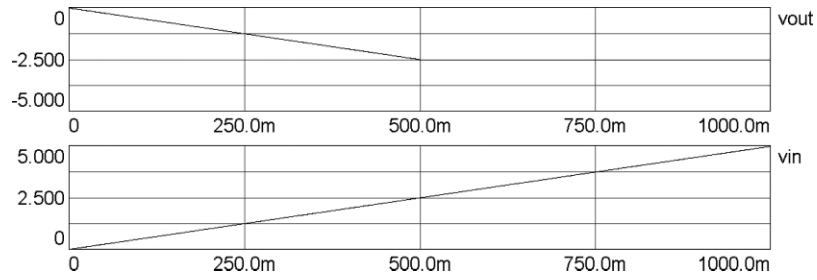


Fig. 23. Slow-transient simulation using hAMSter for Type III faults for the positive values of v_{in} .

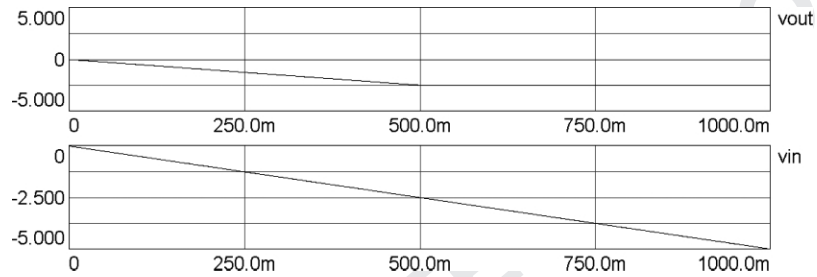


Fig. 24. Slow-transient simulation using hAMSter for Type III faults for the negative values of v_{in} .

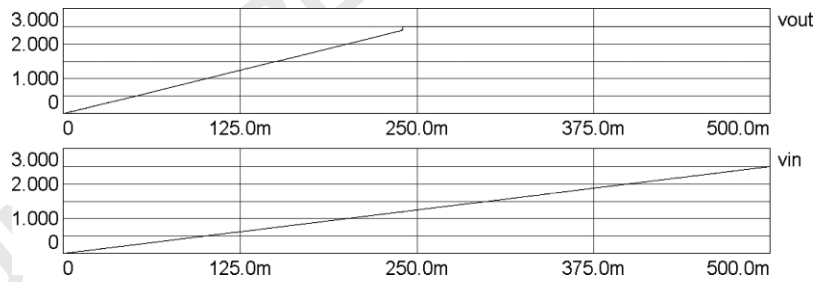


Fig. 25. Slow-transient simulation using hAMSter for Type IV faults for the positive values of v_{in} .

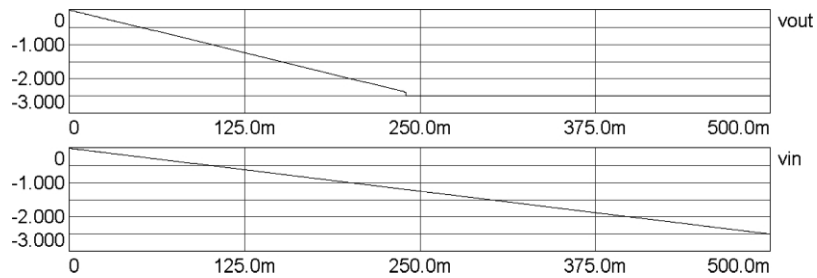


Fig. 26. Slow-transient simulation using hAMSter for Type IV faults for the negative values of v_{in} .

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

Fault type	The CPU time (s)	
	hAMStor	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.

383 for 5 ms with 10 μ s iteration steps. Table 2 shows
 384 the CPU time spent for each case with the different
 385 approaches.

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

390 The reason why the behavioral model is so much
 391 faster than the transistor level for Type III faults is that
 392 Type III faults are open drain faults and HSPICE strug-
 393 gles with an incompletely defined circuit. Finally the
 394 speed-up for Type IV faults is around 2.5 times. The
 395 behavioral model for Type IV faults is relatively slow
 396 compared to other behavioral models due to evalua-

tion of the if-then construct required in the procedural 397
 statement (Fig. 28) to model the Type IV faults. 398

7. Conclusions 399

Capturing circuit behavior under faulty conditions at a 400
 higher level using mathematical equations (behavioral 401
 modeling) is somewhat simpler than the macromodel 402
 approach. 403

Analog fault simulation is a key factor in 404
 analog/mixed-signal test generation. Currently such 405
 fault simulation is of limited use due to the speed of 406
 analog simulation and the large number of faults to 407
 be simulated. Simulation can be speeded up by using 408
 number of techniques. Behavioral modelling is one of 409
 those techniques. We have shown in this paper how one 410
 can increase analog fault simulation speed by using 411
 behavioral models. We have used VHDL-AMS for the 412
 behavioral modelling. It is clear that as VHDL-AMS 413
 simulators become more powerful it will be easier to 414
 model analog/mixed-signal circuits at a higher level so 415
 as to speed-up simulation in general and analog fault 416
 simulation in particular. 417

References 418

1. A.J. Perkins, M. Zwolinski, C.D. Chalk, and B.R. Wilkins, 419
 "Fault modeling and simulation using VHDL-AMS." *Analog In-* 420
tegrated Circuits and Signal Processing, vol. 16, pp. 141–155, 421
 pp. 53–67, 1998. Au: Pls. verify 422
2. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Sys-* 423
tems Testing and Testable Design, IEEE Press, 1990. range. 424
3. www.eda.org/analog 425
4. http://www.hamster-ams.com/ 426
5. C.-Y. Pan and K.-T. Cheng, "Fault macro modeling for 427
 analog/mixed-signal circuits," in *IEEE International Test Con-* 428
ference, ITC'97, 1997, pp. 913–922. 429
6. G. Casinovi and A. Sangiovanni-Vincentelli, "A macromodeling 430
 algorithm for analog circuits." *IEEE Transactions on Computer-* 431
Aided Design of Integrated Circuits and Systems, vol. 10, no. 2, 432
 pp. 150–160, 1991. 433
7. B. Perez et al., "A new nonlinear time-domain op-amp macro- 434
 model using threshold functions and digitally controlled network 435
 elements." *IEEE Journal of Solid-State Circuits*, vol. 23, no. 4, 436
 pp. 959–971, 1988. 437
8. G.A. Boyle, D.O. Pederson, B.M. Cohn, and J.E. Solomon, 438
 "Macromodeling of integrated circuit operational amplifiers." 439
IEEE J. of Solid State Circuits, vol. SC-9, pp. 353–363, 1974. 440
9. M.E. Brinson and D.J. Faulkner, "Modular SPICE macromodel 441
 for operational amplifiers." *IEE Proc.- Circuits Devices Syst.*, 442
 vol. 141, no. 5, pp. 417–420, 1994. 443
10. Y.-J. Chang et al., "A behavior-level fault model for the closed- 444
 loop operational amplifier." *Journal of Information Science and* 445
Engineering, vol. 16, no. 5, pp. 751–766, 2000. 446

- 447 11. C. Chalk and M. Zwolinski, "Macromodel of CMOS opera-
448 tional amplifier including supply current variation." *Electronics*
449 *Letters*, vol. 31, pp. 1398–1400, 1995.
- 450 12. P. Mandal and V. Visvanathan, "Macromodeling of the AC char-
451 acteristics of CMOS op-amps," in *IEEE 1993 Conference on*
452 *Computer Aided Design, Digest of Technical Papers*, 1993, pp.
453 334–339.
- 454 13. M.E. Brinson and D.J. Faulkner, "A SPICE noise macromodel
455 for operational amplifiers." *IEEE Transactions on Circuits and*
456 *Systems-I: Fundamental Theory and Applications*, vol. 42, no.
457 3, pp. 166–168, 1995.
- 458 14. G. Krajewska and F.E. Holmes, "Macromodeling of FET/bipolar
459 operational amplifiers." *IEEE Journal of Solid-State Circuits*,
460 vol. SC-14, no. 6, pp. 1083–1087, 1979.
- 461 15. C. Turchetti and G. Masetti, "A macromodel for integrated all-
462 MOS operational amplifiers." *IEEE Journal of Solid-State Cir-*
463 *cuits*, vol. SC-18, pp. 389–394, 1983.
- 464 16. M.E. Brinson and D.J. Faulkner, "SPICE macromodel for op-
465 erational amplifier power supply current sensing." *Electronics*
466 *Letters*, vol. 30, no. 23, pp. 166–168, 1994.
- 467 17. R.V. Peic, "Simple and accurate nonlinear macromodel for op-
468 erational amplifiers." *IEEE Journal of Solid-State Circuits*, vol.
469 26, no. 6, pp. 896–899, 1991.
- 470 18. B. Al-Hashimi, "Behavioural simulation of filters." *IEE Collo-*
471 *quium on Analogue Simulation: The Dream & The Nightmare*,
472 pp. 51–55, 1995.
- 473 19. A.I. Kayssi and K.A. Sakallah, "Macromodel simplification us-
474 ing dimensional analysis," in *1994 Int. Symp. on Circuits and*
475 *Systems*, 1994, pp. 335–338.
- 476 20. M. Zwolinski, Z.R. Yang, and T.J. Kazmierski, "Using robust
477 adaptive mixing for statistical fault macromodelling." *IEE Pro-*
478 *ceedings: Circuits, Devices and Systems*, vol. 147, no. 5, pp.
479 265–270, 2000.
- 480 21. G. Casinovi, "Multi-level simulation of large analog systems
481 containing behavioral models." *IEEE Transactions on Computer*
482 *Aided Design of Integrated Circuits and Systems*, vol. 13, no. 11,
483 pp. 1391–1399, 1994.
- 484 22. E. Bruls et al., "Analogue fault simulation in standard VHDL." *IEE Proc. Circuits Devices Syst.*, vol. 143, no. 6, pp. 380–385, 1996.
- 487 23. C.G. Broyden et al., "A class of methods for solving nonlinear
488 simultaneous equations," *Mathematics of Computation*, vol. 19,
489 no. 92, pp. 577–593, 1965.
- 490 24. <http://www.ee.washington.edu/mad/benchmarks/benchmarks.html>
- 491
- 492 25. E. Christen and K. Bakalar, "VHDL-AMS, A hardware descrip-
493 tion language for analog applications." *IEEE Transactions on*
494 *Circuits and Systems-II: Analog and Digital Signal Processing*,
495 vol. 46, no. 10, pp. 1263–1272, 1999.
- 496 26. VHDL Language Reference Manual, IEEE Standard 1076–
497 1993.
- 498 27. Standard Description Language Based on the Verilog™
499 Hardware Description Language, IEEE Standard 1364–
500 1995.
- 501 28. www.mentor.com
- 502 29. www.cadence.com
- 503 30. www.verilog.com
- 504 31. www.analog.com
- 505 32. www.ednmag.com
33. Y. Kiliç, "Testing techniques and fault simulation for analogue CMOS integrated circuits." University of Southampton, PhD Thesis, Chapter 6, 2001. 506 507 508



Yavuz Kiliç 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. 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524



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. 525 526 527 528 529 530 531 532 533 534