Analog Integrated Circuits and Signal Processing, 16, 141–155 (1998)
© 1998 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Fault Modeling and Simulation Using VHDL-AMS

A. J. PERKINS, M. ZWOLINSKI, C. D. CHALK AND B. R. WILKINS
Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

Received February 28, 1996; Accepted June 19, 1997

Abstract. Fault simulation is an accepted part of the test generation procedure for digital circuits. With complex analog and mixed-signal integrated circuits, such techniques must now be extended. Analog simulation is slow and fault simulation can be prohibitively expensive because of the large number of potential faults. We describe how the number of faults to be simulated in an analog circuit can be reduced by fault collapsing, and how the simulation time can be reduced by behavioral modeling of fault-free and faulty circuit blocks. These behavioral models can be implemented in SPICE or in VHDL-AMS and we discuss the merits of each approach. VHDL-AMS does potentially offer advantages in tackling this problem, but there are a number of computational difficulties to be overcome.

Key Words: analog simulation, analog test, fault simulation, fault modeling, analog VHDL

1. Introduction

As integrated circuits have grown in complexity, the importance of testing has also grown. It is not sufficient to consign test issues to some post-design phase. Design-for-test, in order to facilitate testing at the manufacturing stage, is now firmly established as an important aspect of digital system design [1]. For analog and mixed-signal circuits, however, testing is still often regarded as a peripheral matter, because design-for-test is perceived to adversely affect performance. Moreover, the testing of small analog circuits may simply be a number of functional tests. Large analog and mixed-signal integrated circuits make functional testing very difficult. Individual parts of the design cannot be tested in isolation, while functional tests may not reveal deeply embedded defects.

While functional testing can be and is used for digital circuits, it is common to assume the existence of stuck-at, bridging and open faults. The testing methodology therefore becomes one of identifying the presence or otherwise of these *structural* faults [2,3]. The object of a testing program for a circuit is to verify whether or not a fault exists using the smallest possible number of test vectors. In general, one test will detect more than one fault and each fault is covered by more than one test. Test pattern generation is thus the process of selecting an optimal set of tests from all possible input patterns. As the generation of the optimal set is unlikely to be feasible for anything other than the smallest circuits, algorithms such as the D-algorithm or PODEM are used to find a test pattern for one fault [2]. Alternatively, random test patterns may be applied. Once a pattern is found, its fault cover can be assessed and all faults detectable with that pattern can be dropped from further consideration. The assessment of fault cover is made using *fault simulation*. For each fault, a copy of the circuit is made containing that fault and no other. The original, fault-free circuit and the faulty copies are simulated with the given test pattern. If the output of the faulty copy differs from that of the original the fault is detectable using that test pattern. Techniques, such as *concurrent fault simulation* [2], exploit the fact that the differences in behavior between the faulty and fault-free circuits are often relatively small, and by avoiding redundant element evaluation, reduce the computational effort required to evaluate all the faulty circuit copies.

Most of these techniques are not immediately applicable to analog circuits. Structural defects, such as open and short circuits, can be identified, but these do not necessarily manifest themselves a simple stuck faults. Hence, test vector generation has to be done in an *ad hoc* manner. The major difficulty, however, is fault simulation. The simulation of analog circuits is at least two orders of magnitude slower than that of similarly sized digital circuits [4]. The number of potential faults to be modeled is far greater in an analog circuit. In a digital circuit, we are only concerned with the interconnection between gates and we assume that the nodes will be stuck-at 1 or stuck-at 0. In an analog circuit we must concern ourselves with every node in the circuit and, in the worst case—in the absence of detailed layout information—we must assume the possibility of every pair of node-to-node shorts and of each branch being open circuit. Thus the number of fault simulations to be performed is likely to be greater than for a similar digital circuit. Finally, a digital node is, after transitions have stabilized and contentions have been resolved, either 1 or 0; an analog node has a value represented by a floating point number. A node in a faulty digital circuit is either exactly the same as in the fault-free circuit or different, making redundancy easy to exploit. In an analog circuit a slight difference between a faulty and a fault-free node may result in a massive difference elsewhere in the circuit, thus making redundancy very difficult to exploit. Moreover, the existence of faults in the analog circuit model may take component models outside their normal, characterized region of behavior and ultimately may render the faulty circuit unsimulatable.

It can thus be seen that analog fault simulation has not been a usable tool for analyzing the effectiveness of test vectors, because too many slow simulations are required. The motivation for the work described in this paper has been to test the effectiveness of various supply current monitoring techniques in detecting analog faults and hence the need to perform fast analog fault simulations.

We first describe how the speed of an analog simulation may be increased by modeling circuits behaviorally. Macromodels for standard analog building blocks such as operational amplifiers have been developed using controlled sources in SPICE. VHDL-AMS offers an alternative approach that is in many ways simpler [5]. We further describe how simpler behavioral models can be used for operational amplifiers in closed-loop configurations. This is important for fault modeling because open-loop characteristics are not generally observable in embedded circuits. The number of simulations can be reduced by observing which faults cause similar observable behavior. An example of fault collapsing is given in Section 3. By modifying the characteristics of the fault-free SPICE and VHDL-AMS behavioral models, faulty behavior can be modeled. We therefore go on to show that such behavioral models can emulate the transistor level models very accurately. One particular difficulty addressed is the propagation of faulty behavior through fault-free circuit blocks; this can take those fault-free blocks outside their characterized region of behavior, thus requiring more complex models.

In order to assess the suitabilty of VHDL-AMS for this work, a commercial implementation of a draft of the 1076.1 working documents was used. Certain difficulties with simulation speed and with robustness were encountered, and these are discussed.

2. Behavioral Modeling

A simple two-stage CMOS operational amplifier is shown in Fig. 1. This is significantly simpler than a ''real'' design; nevertheless it has 8 transistors, 9 nodes and 15 branches, and in comparison to a digital circuit of a similar size, the time required for simulation is relatively high. A number of SPICE macromodels for general operational amplifiers have been developed [6]; the model described in [7] includes the supply current. A comparison of simulation times is given in Table 1. The macromodels are all based around controlled sources. In SPICE, controlled sources may be simply linear functions of one voltage or current, or polynomial functions of several voltages or currents. Such elements are therefore not sufficient to model voltage or current limiting as happens for instance when an op-amp's output voltage saturates. Such effects are instead modeled by the inclusion of, for instance, diodes. Similarly, differential inputs may be modeled by a simplified MOS transistor pair. SPICE macromodels of circuits such as op-amps are therefore a combination of controlled sources and simplified semiconductor elements.

Unsurprisingly, the characterization of such macromodels is not easy. In effect, we are forced to do a multidimensional optimization. In practice, the problem is simplified, to some extent, because each part of the macromodel corresponds to some observable behavior, so the optimization can be done in parts. Analog hardware description languages make such macromodeling a potentially simpler task.

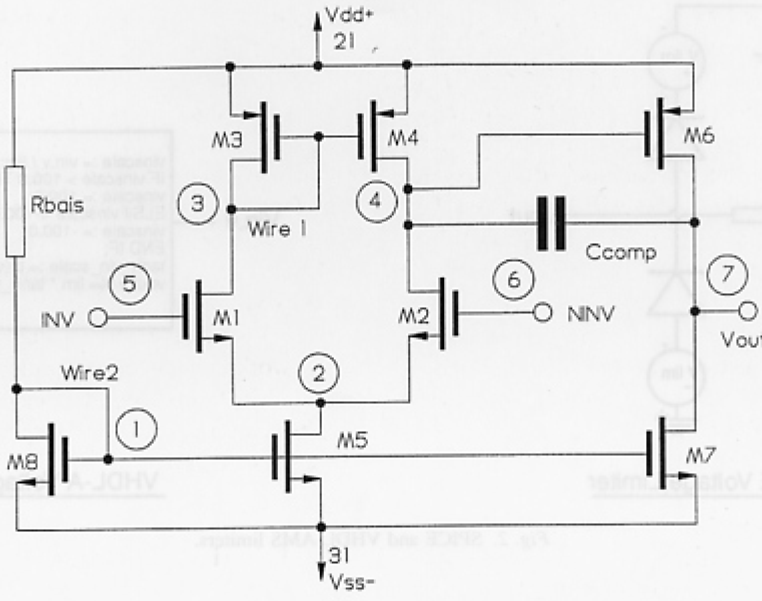


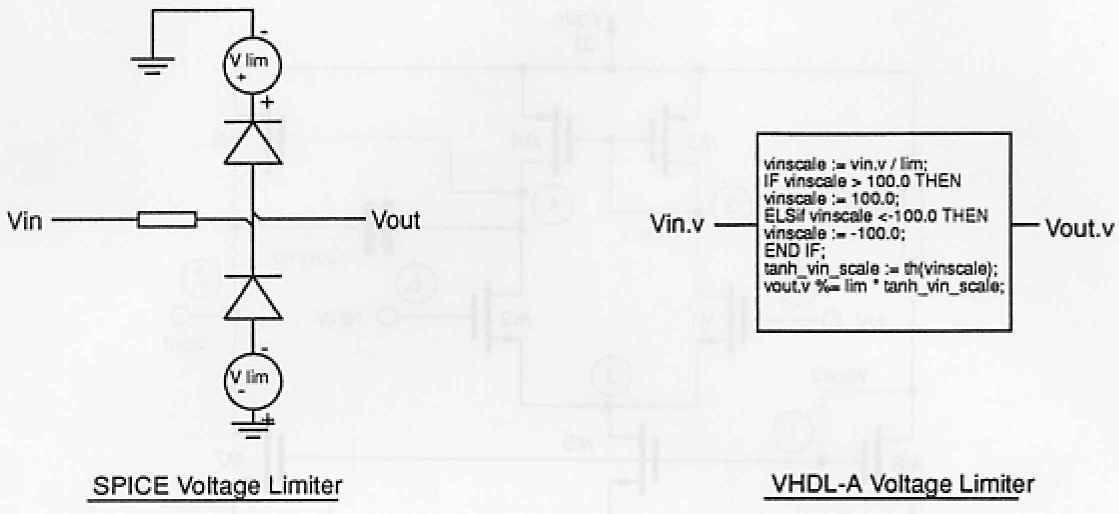
Fig. 1. Two stage CMOS op-amp.

Instead of forcing a mathematical model onto a limited set of defined components, the model can be implemented directly. A number of VHDL-AMS models (or to be precise, models based on the implementation of the draft) have been written which illustrate this, Fig. 2 shows a SPICE and VHDL-AMS implementation of a voltage limiter. For the SPICE model a number of standard components are required to achieve the desired function where as in the VHDL-AMS implementation a simpler mathematical description is used. Another example of a VHDL-AMS model, this time of an open loop op-amp, is shown in Fig. 3, and in Fig. 4 the complete VHDL-AMS code is given. The behavior of the op-amp is defined in terms of its gain, transfer function, output impedance, slew rate, CMRR and power supply current characteristics. Each of these characteristics can be incorporated directly into a mathematical expression. The derivation of the model is described in the Appendix. As with the macromodel of [7], the op-amp model includes the supply current variation. Unlike the SPICE macromodel, the behavior of the model maps directly onto the block diagram.

Again, unlike SPICE, the relation between each parameter and the model is explicitly stated, without the need to map the effects onto controlled source parameters. Because of this clarity, the VHDL-AMS model is, arguably, superior to the SPICE model. Op-amps are not usually used in open-loop configuration, therefore it may not be necessary to model open-loop effects that are not observable in closed-loop configurations. This is especially significant with respect to testing—a common testability metric is whether a node that might have a fault can be controlled from an input and whether the effect of that fault can be observed at an output. Thus it may be argued that a change in the open-loop gain, for

Table 1. Comparison of simulation times for various CMOS op-amps.

Op-amp model	CPU time (s) for 30 μs simulated time
Transistor level, two-stage	15.6
Transistor level, cascode	23.7
Complete macromodel	9.7
Macromodel without V & I limiting	6.6



```
vinscale := vin.v / lim;
IF vinscale > 100.0 THEN
vinscale := 100.0;
ELSif vinscale <-100.0 THEN
vinscale := -100.0;
END IF;
tanh_vin_scale := th(vinscale);
vout.v %= lim * tanh_vin_scale;
```

Fig. 2. SPICE and VHDL-AMS limiters.

instance, may not be directly observable in a closed-loop configuration, and that therefore there is no point in attempting to model it. If an op-amp is modeled in its closed-loop configuration, its macromodel can be significantly simplified. Fig. 5 shows the macromodel for an op-amp configured as an inverting amplifier. It will be observed that 7 parameters are sufficient to completely model the behavior of this circuit. The same macromodel can be used for a non-inverting amplifier configuration, but with different parameter values. A similar model can be constructed for the summing amplifier case. A VHDL-AMS version of the model is shown in Fig. 6. As can be seen, this model is significantly simpler than the open-loop model of Fig. 3. Fig. 7 compares two simulations of the audio mixer circuit of Fig. 8 for both the output voltage and the supply current. In one simulation the circuit is modeled entirely at the transistor level and in the other using VHDL-AMS models of each block. The differences between the models are insignificant.

3. Fault Collapsing and Fault Modeling

As has been noted, defects in digital circuits can often be characterized as single stuck-faults. This fault model represents a form of fault collapsing as a

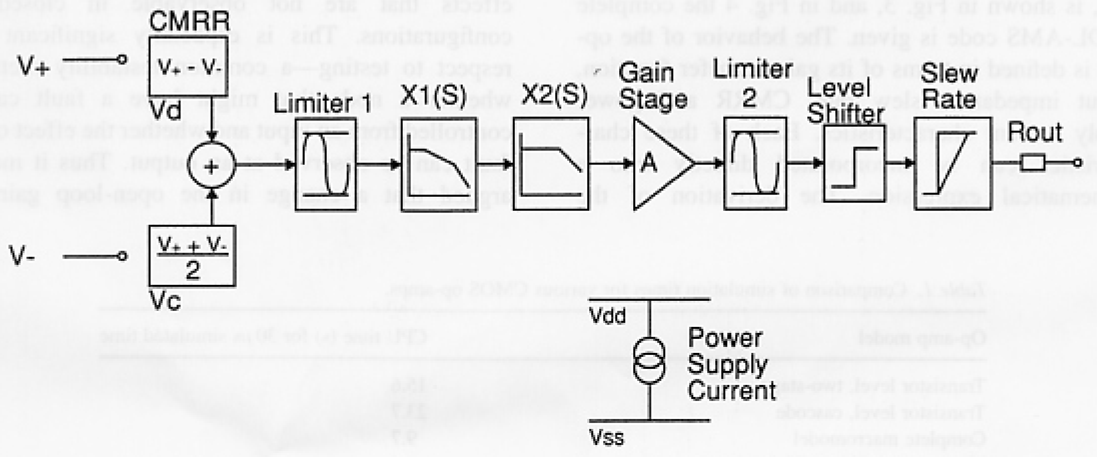


Fig. 3. VHDL-A op-amp block diagram.

```
-- op-18.vhdla Full opamp version 2

ENTITY opamp18 IS
     GENERIC (p1,p2,z1,gain,cmrr,srate,rop,ron,rdc,gmdd,limdiff : real);
     PIN (vinp,vinn,vout,Vdd,Vss : electrical);
END ENTITY opamp18;


ARCHITECTURE a OF opamp18 IS
     STATE     outp1,outp2,soutp1,soutp2,sinp1,
          sinp2,inp1,inp2 : analog;               -- Implicit Variables
     VARIABLE  ip1,ip2,iz1 :analog;             -- temp poles and zero
     VARIABLE  vd,vc,vca,vin,vzeros,comgain :analog;        -- CMRR
     VARIABLE  inl1,outl1,inl2,outl2,lim1,lim2               -- Limiters in/out
          ,vinscale,tanh_vin_scale : ANALOG;                 -- Limiter internal
     VARIABLE  outg :analog;
     VARIABLE  inls,outls : analog;             -- level shift
     VARIABLE  inar,rate,diff,diff2,marate  :ANALOG;         -- Slew rate
     STATE     outar :Analog;                     -- slew rate
     VARIABLE  inro,io,ro :ANALOG;
     VARIABLE  idc,iac :ANALOG;

BEGIN
  RELATION
   PROCEDURAL FOR init =>
   -- default generics
    p1 := 1.0e7;               -- low frequency pole
    p2 := 1.0e2;               -- high frequency pole
    z1 := 1.0e8;               -- high frequency zero
    gain := 1.0e6;             -- open loop  DC gain
    cmrr := 90.0;              -- input CMRR
    srate := 500.0e3;          -- slew rate
    rop := 1.0;                -- output resistance when vout +
    ron := 1.0;                -- output resistance when vout -
    rdc := 5.0e3;              -- Power supply offset current resistance
    gmdd := 100.0e-6;          -- Power supply current vout conductance
    limdiff := 0.5;            -- Output voltage limit from Vdd

    -- seting up varables
    ip1 := 1.0/(twopi*p1);
    ip2 := 1.0/(twopi*p2);
    iz1 := 1.0/(twopi*z1);
    comgain := 1.0 / (10.0** (cmrr/20.0) );
-- ****************************************************************;
-- *********  AC & DC  *********************************************;
PROCEDURAL FOR ac,dc =>
    vzeros := (Vdd.v + Vss.v) * 0.5;     -- virtual 0v of opamp;

    -- ** CMRR **
    vd := [vinp,vinn].v;
    vc := (vinp.v + vinn.v)*0.5 - vzeros;
    vca := vc * comgain;
    vin  := vd + vca;

    --*********** Limiter 1 ******************
     inl1 := vin;
     lim1 := Vdd.v - vzeros;
     vinscale := inl1 / lim1;
     IF vinscale > 100.0 THEN vinscale := 100.0;
     ELSif vinscale <-100.0 THEN vinscale := -100.0;
     END IF;
     tanh_vin_scale := th(vinscale);
     outl1 := lim1 * tanh_vin_scale;

    --********* Pole1 ****************************************
    inp1 := outl1;
    sinp1 := ddt(inp1);
    soutp1 := ddt(outp1);
    -- soutp1/p1 + outp1 == inp1; ** IMPLICIT EQUATION **

    --************ Pole2 ****************************************
    inp2 := outp1;
    sinp2 := ddt(inp2);
    soutp2 := ddt(outp2);
    --  soutp2/p2 + outp2 == inp2 + sinp2/z1;  ** IMPLICIT EQUATION **

    --************ Gain Stage ***************
    outg := outp2*gain;

    --*********** Limiter 2 ******************
    inl2 := outg;
    lim2 := Vdd.v - vzeros - limdiff;
    vinscale := inl2 / lim2;
    IF vinscale > 100.0 THEN vinscale := 100.0;
    ELSif vinscale <-100.0 THEN vinscale := -100.0;
    END IF;
    tanh_vin_scale := th(vinscale);
    outl2 := lim2 * tanh_vin_scale;

    -- *******Shift the output of the gain to be within the power supply **
    inls := outl2;
    outls := inls + vzeros;

    -- ************ Slew rate limiting ***********
    inar := outls; --Remove because of timestep function
    outar := inar;

    -- *********** output resistance **************
    inro := outar;
    io := -vout.i;               -- vo.i is the current into vo !!
    IF (inro > 0.0) THEN ro := rop; -- rop = resistance out positive
     ELSE ro := ron;             -- ron resistance out negative
    END IF;
     vout.v %= inro - (io * ro);          -- ** assign the output **

    -- *********** Power Supply current ***********
    idc := (Vdd.v - vzeros)/ rdc;
    iac := gmdd * inro;
    Vss.i %= idc + iac + io;
    Vdd.i %= idc + iac - io;


-- ****************************************************************;
-- *********  TRANSIENT  *******************************************;
  PROCEDURAL FOR transient =>
    vzeros := (Vdd.v + Vss.v) * 0.5;     -- virtual 0v of opamp;

    -- ** CMRR **
    vd := [vinp,vinn].v;
    vc := (vinp.v + vinn.v)*0.5 - vzeros;
    vca := vc * comgain;
    vin  := vd + vca;

    --*********** Limiter 1 ******************
     inl1 := vin;
     lim1 := 0.1; --Vdd.v - vzeros;
     vinscale := inl1 / lim1;
     IF vinscale > 100.0 THEN vinscale := 100.0;
     ELSif vinscale <-100.0 THEN vinscale := -100.0;
     END IF;
     tanh_vin_scale := th(vinscale);
     outl1 := lim1 * tanh_vin_scale;

    --********* Pole1 ****************************************
    inp1 := outl1;
    sinp1 := ddt(inp1);
    soutp1 := ddt(outp1);
    -- soutp1/p1 + outp1 == inp1; ** IMPLICIT EQUATION **
    --************ Pole2 ****************************************
    inp2 := outp1;
    sinp2 := ddt(inp2);
    soutp2 := ddt(outp2);
    --  soutp2/p2 + outp2 == inp2 + sinp2/z1;  ** IMPLICIT EQUATION **

    --************ Gain Stage ***************
    outg := outp2*gain;

    --*********** Limiter 2 ******************
    inl2 := outg;
    lim2 := Vdd.v - vzeros - limdiff;
    vinscale := inl2 / lim2;
    IF vinscale > 100.0 THEN vinscale := 100.0;
    ELSif vinscale <-100.0 THEN vinscale := -100.0;
    END IF;
    tanh_vin_scale := th(vinscale);
    outl2 := lim2 * tanh_vin_scale;

    -- *******Shift the output of the gain to be within the power supply **
    inls := outl2;
    outls := inls + vzeros;

    -- ************ Slew rate limiting ***********
    inar := outls;
    marate := -1.0 * srate;
    diff := inar - outar;
    rate := diff/time_step;
    IF rate > srate THEN diff2 := srate * time_step;
    ELSIF rate < marate THEN diff2 := marate *time_step;
    ELSE diff2 := diff;
    END IF;
    outar := outar + diff2 ;

    -- *********** output resistance **************
    inro := outar;
    io := -vout.i;               -- vo.i is the current into vo !!
    IF (inro > 0.0) THEN ro := rop; -- rop = resistance out positive
     ELSE ro := ron;             -- ron resistance out negative
    END IF;
     vout.v %= inro - (io * ro);          -- ** assign the output **

    -- *********** Power Supply current ***********
    idc := (Vdd.v - vzeros)/ rdc;
    iac := gmdd * inro;
    Vss.i %= idc + iac + io;
    Vdd.i %= idc + iac - io;

-- ****************************************************************
    EQUATION (outp1,outp2) FOR ac, dc, transient =>

    --** Pole1 **
--    soutp1/p1 + outp1 == inp1;
    soutp1*ip1 + outp1 == inp1;


    --** Pole2 + Zero **
    soutp2*ip2 + outp2 == inp2 + sinp2*iz1;


   END RELATION;
END ARCHITECTURE a;
```

Fig. 4. VHDL-AMS code for open-loop op-amp.

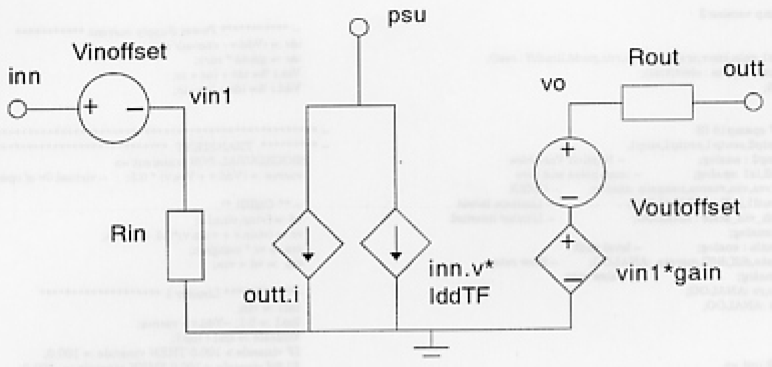


Fig. 5. Op-amp macromodel.

potentially large number of structural defects are assumed to manifest themselves as simple electrical faults. Even if the fault model is imprecise, test patterns designed to detect such faults will sometimes uncover other defects [8]. Analog circuits do not lend themselves to such simple fault models, not least because by definition, the range of possible behaviours is continuous rather than discrete. Nevertheless, by intuition it is likely that some defects will cause similar faulty behavior to others. If this can be shown to be true, the number of faults to be simulated would be reduced and hence the complexity

```
ENTITY opamp IS
  GENERIC(Rin, Vinoffset, Rout, gain,
          Voutoffset, IddTF : analog);
  PIN(inn, outt, psu : electrical);
END ENTITY opamp;

ARCHITECTURE c_loop OF opamp IS
  VARIABLE vin1, vo :analog;

BEGIN
  RELATION
    PROCEDURAL FOR init =>
      Rin := 400.0e3;
      Vinoffset := 40.0e-6;
      Rout := 1.0e3;
      gain := -50.0;
      Voutoffset :=0.0;
      IddTF := 70.0e-6;

    PROCEDURAL FOR ac,dc,transient =>
      vin1 := inn.v - Vinoffset;
      inn.i %= vin1 / Rin;
      vo := vin1 * gain + Voutoffset;
      outt.v %= vo - (-outt.i * Rout);
      psu.i %= -( (inn.v*IddTF) + outt.i);

  END RELATION;
END ARCHITECTURE c_loop;
```

Fig. 6. VHDL-AMS closed-loop op-amp model, including fault modeling.

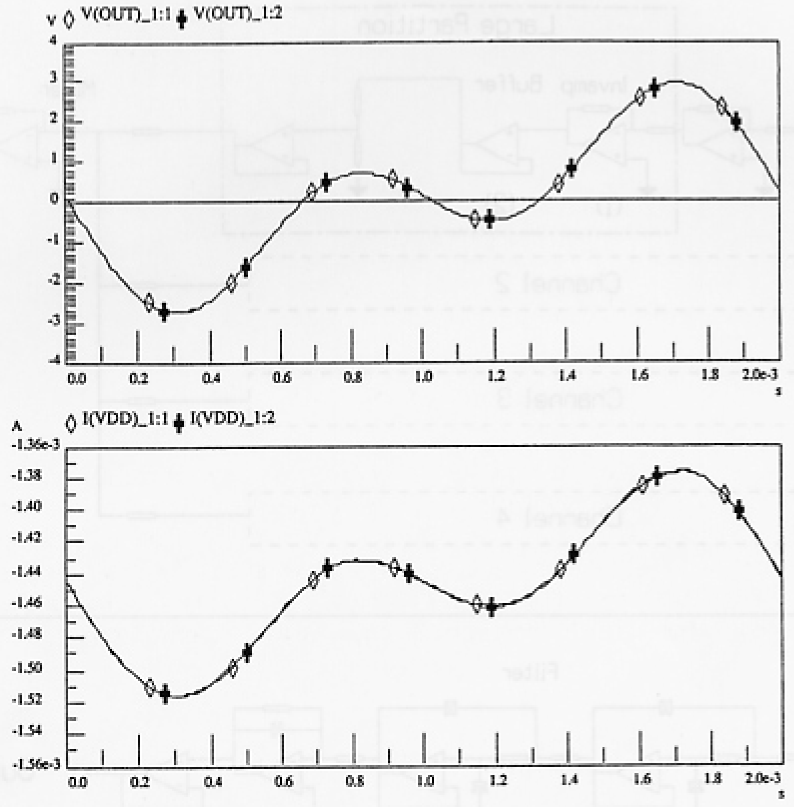


Fig. 7. A comparison between the VHDL-AMS and SPICE macromodel.

of fault simulation for analog circuits would be reduced.

A deeper problem exists concerning what constitutes a fault in an analog integrated circuit. Defects in the form of missing or extra metal or other material will cause bridging faults, open circuits and parametric changes to devices. In theory, the number of such potential defects over an integrated circuit tends towards infinity. Techniques and simulators [9] exist to predict the most likely defects, but the existence or otherwise of defects is dependent on the layout of the circuit. It is not therefore possible to derive a definitive fault list simply from the circuit schematic. As an alternative, it is possible to assume, for a small circuit, that a possible bridging fault exists between each network node and every other node. We discount, for the moment, bridging faults involving three or more nodes. Similarly, it is possible to assume that each branch of a circuit may become an open circuit. Finally, we can change the parameters of each network component. Of these possible faults, short circuits are probably the most likely, and can be modeled in SPICE as small resistances of e.g. 1 Ω between nodes [10]. Open circuits involving MOSFETs cause the drain source current to be zero at all times and can be modeled by setting the threshold voltage, V_T, to, say, 100 V [11]. Thus one fault model covers three faults. Both the fault models can be introduced without topological changes to the SPICE netlist, but introducing such changes manually is nevertheless a tedious task. Automatic generation of netlists containing faults and their automatic simulation is possible [12], but the time-consuming nature of analog fault simulation remains. The two-stage CMOS op-amp of Fig. 7 has 9 nodes, including the supply rails and 15 branches. There are, therefore 36 possible bridging faults involving two nodes (although we would normally choose to omit the case where the supplies are bridged) and 15 possible open circuit faults, giving a total of 51 possible faults and hence 52 simulations of such a circuit would be required. It will be appreciated that any technique to

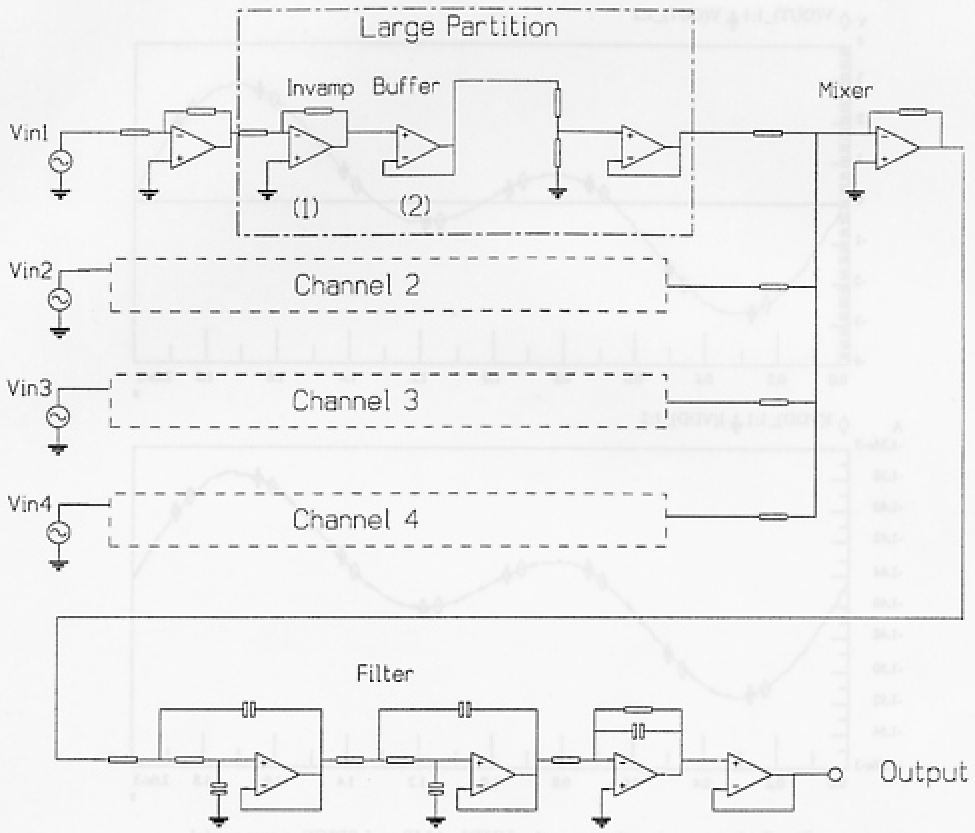


Fig. 8. Audio mixer circuit.

reduce the number of simulations is therefore of interest. Parametric changes have been omitted from consideration, not least because the distinction between a "fault" and a performance change due to parameter variation is difficult to define.

As the existence, or otherwise, of a fault is only observable at the outputs of a circuit, we have not considered the equivalence of faults in op-amps in open-loop configuration. Instead, we have again examined op-amps configured as inverting, non-inverting and summing amplifiers. The behavior of each circuit configuration was assessed using transient, AC and pole-zero analyses. The following measurements were taken:

- the output voltage offset (transient analysis with 100 Hz sine wave input)
- the RMS supply current (transient analysis with 100 Hz sine wave input)
- the voltage gain (v(out)/v(in)) (AC analysis)
- the supply current to input voltage admittance (i(vdd)/v(in)) (AC analysis)
- the input resistance (AC analysis)
- the output resistance (AC analysis)
- the voltage gain pole (AC & pole-zero analysis)

The two transient analyses can effectively be considered as DC sweeps. Table 2 shows how these measurements compare for a number of faults. If the all the measured values for a group of faults are the same or very similar, as shown, those faults can be collapsed, i.e. only one of the group needs to be simulated.

Certain faults cause difficulties in the SPICE analyses. For instance, a fault may cause the network matrix to tend towards singularity. At the very least, the simulation time increases; at worst the simulation fails completely. Other faults move the circuit away from the normal operating point. These latter faults will cause the output voltage to behave in a very non-

Table 2. Selected results of fault simulations of the inverting amplifier.

Fault	vout	psu rms	vout tf	idd tf	Rout (out)	Rin	Pole (out)	Fault Group
fault free	0v	4e-6	−49.8	70u	6.40	400k	20k	
2 to 3	−5v	0.0	73u	0	1.5k	20M	391k	A
2 to 4	4.9	0	−13m	0	580	20.2M	2.6M	B
2 to 5	4.9	0	−10m	0	480	8M	85k	B
2 to 6	−5	0	74u	0	1.5k	20.4M	155k	A
2 to 7	4.9	0	29u	0	604	20.4M	100k	B
2 to 21	−5	0	74u	0	1.5k	20.4M	126k	A
3 to 6	−5	7.6e − 7	74u	13u	1.5k	20.4M	1M	A
3 to 7	3.1	0	2	2.8u	35	19M	220k	D
3 to 21	4.9	0	−12m	0	570	20.1M	619k	B
4 to 21	−5	0	74u	0	1.5k	20.4M	4.2M	A
m1 o/c	4.94	0	18u	50n	375	20.4M	1.4M	C
m2 o/c	−5	0	74u	65n	1.5k	20.4M	5M	A
m3 o/c	−5	0	74u	0	1.5k	20.4M	1.5M	A
m4 o/c	4.91	0	−12m	0	577	20.4M	679k	C
m5 o/c	−4.84	0	117u	0	2.4k	20.4M	1M	A
m6 o/c	−5	0	74u	65n	1.5k	20.4M	772k	A

Table 3. Summary of fault collapsing.

	Bridging faults	No. open faults	Fault groups	Ungrouped	Unsimulatable	Nonlinear	Indistinguishable
Inverting	38	10	16	5			
Summing	38	10	12	15	1		
Non-inverting	32	10	6		1	14	1

linear way with respect to the input voltage. It is not generally possible to group such faults. Table 3 summarizes the fault collapsing for the three op-amp configurations. The number of fault simulations can be reduced in each case by up to 56%.

4. Behavioral Fault Modeling

For those fault groups that change the characteristics of the op-amp, while maintaining linear behavior, the faulty behavior can be mapped onto the SPICE behavioural model of the configured op-amp, by changing model parameters as specified in Table 2. Those faults that produce non-linear behavior require a more complex behavioral model. The linear controlled sources must be replaced by piecewise linear controlled sources. These models are not, however, available in all versions of SPICE. Exactly the same parametric changes may be made to the VHDL-AMS models to give behavioral models of faulty circuit blocks. The non-linear effects of certain faults can be modeled explicitly in VHDL-AMS.

In principle, therefore, simulations of complex analog (and mixed-signal) circuits can be performed in which fault-free and faulty circuit blocks are modeled behaviorally. Because of fault-collapsing, the number of fault simulations is reduced, and because of behavioral modeling, the complexity of the model and hence the simulation time is, in theory, reduced.

We tested this assumption with two circuits with multiple op-amps. The first circuit, an audio mixer shown in Fig. 8 , has four identical input channels with four op-amps in each, followed by a mixing and filtering section with five op-amps. The circuit can be thought of as simply a chain of standard op-amp configurations. The second circuit is a leapfrog filter, Fig. 9, that has six op-amps and both local and global feedback. In both cases, we introduced faults into an op-amp and observed, by simulation at transistor and behavioral level, the effect on both the output and the supply current for the entire circuit. The detailed

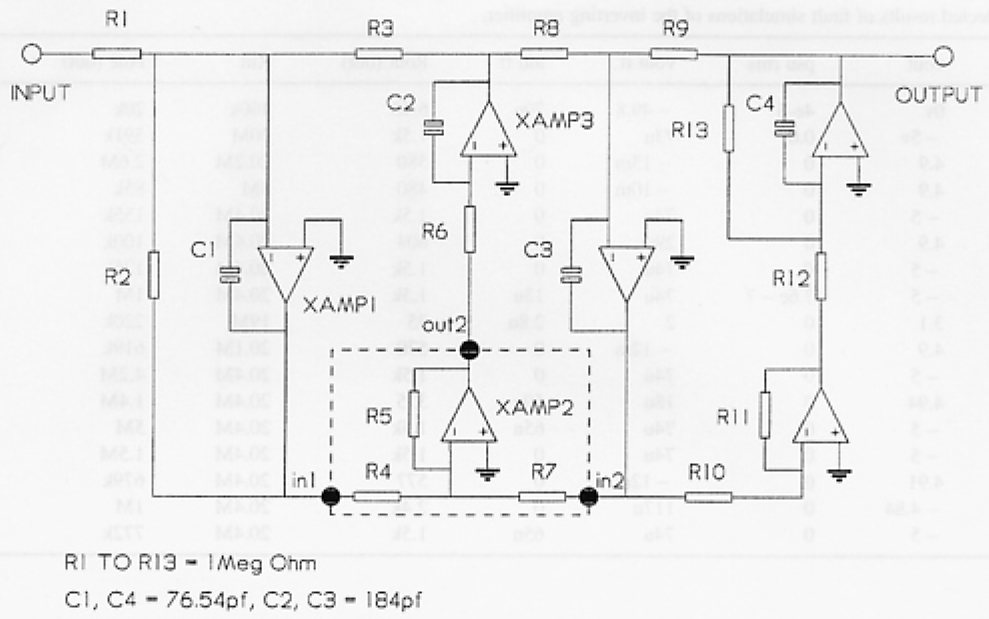


Fig. 9. Leapfrog filter.

results of these investigations are described elsewhere, but to illustrate the potential accuracy of behavioral modeling, Figs. 10 and 11 show that the transistor models, the SPICE macromodels and the VHDL-AMS models give almost exactly the same voltage output and supply current behavior when faults are inserted into op-amp (1) in the mixer circuit. Fig. 10 shows the response to a fault in fault class A, while Fig. 11 shows the response to a fault from fault class B.

Certain faults, however, cause particular difficulties. In general, we consider a fault is detectable if it causes the supply current to vary by more than, say, 10%. Many of the behavioral models give an RMS supply current value with an accuracy to within 2% of that of the transistor models. As we are measuring the supply current of the entire circuit, in an attempt to emulate realistic test conditions, it should be noted that the introduction of a fault model may cause the output of that circuit block to take an abnormal value, and hence the supply current of succeeding stages may also differ from the fault-free value. The fault masking effect that has been observed with quiescent supply current monitoring in digital circuits [13] does not necessarily apply to analog circuits. Faults that cause the supply current to differ significantly from the fault-free case may cause the absolute value of the supply current simulated with behavioral models to be somewhat inaccurate when compared with the transistor model. It may be argued that these differences do not matter—provided that the absolute value is outside the normal circuit tolerances, its precise value is unimportant.

Further difficulties arise when the effect of a fault is to change either the input or output current or voltage of a stage to such an extent that the neighboring stage is operated outside its specified range. This is a particular, philosophical difficulty with all fault modeling. Models are in general designed to capture a particular set of characteristics. The introduction of a fault causes, by definition, abnormal effects. Therefore, if a fault causes a neighboring stage to operate in an unforeseen manner, the simulation results are likely to be inaccurate. This has been observed in the mixer circuit, where a bridging fault at the input of the buffer op-amp, (2) in Fig. 8, causes the previous stage to be heavily loaded. Similarly, in the leapfrog filter, the global feedback tends to reinforce fault effects in the highlighted op-amp, driving the output of that stage to one of the supply rails. This, in turn, takes the succeeding stage outside its normal operating range and causes it to saturate too. As we are concerned with modelling the supply current, it is important that these saturation effects should be

Fault Class A

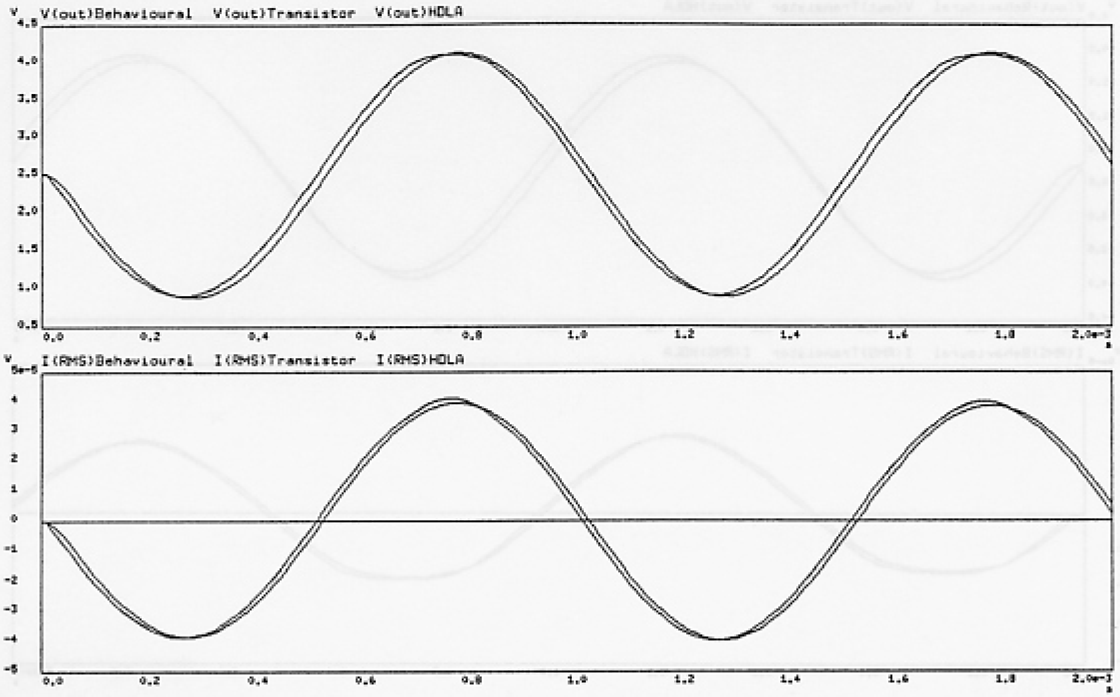


Fig. 10. Fault class A comparison between the transistor model, SPICE macromodel and the VHDL-AMS model.

properly modeled. In the leapfrog filter, the integrating stages are modeled using the open-loop op-amp model with a capacitor in the feedback loop.

These fault conditions can be modeled, behaviorally, in a number of ways. In the case of the mixer circuit, the scope of each behavioral block can be extended to embrace not one op-amp, but three. In effect therefore, we perform a higher level of fault collapsing and modeling. The effects of a fault are buffered from neighboring stages by implicit fault-free models of the adjacent op-amps, operating under faulty conditions. The fault thus occurs in the middle of a sliding window, which moves around the circuit according to the fault to be modeled.

The out-of-specification effects in the leapfrog filter are harder to model. The basic problem is that the behavioral model of an op-amp does not deliver enough current when the output is saturated. Thus the output stage of the macromodel can be enhanced, either by using more complex controlled sources, or even with a simple MOS transistor representation of the output stage.

The CPU times for simulations of two channels of the audio mixer of Fig. 8 for 2 ms with a stimulus of 1 kHz are shown in Table 4. Three modeling approaches are compared: a full transistor model; a SPICE behavioral model and a VHDL-AMS behavioral model. Three fault simulation times are shown, together with that of the fault-free simulation. It will be seen that the SPICE behavioral simulation is about 3 times faster than the transistor-level simulation, but that the VHDL-AMS simulation is significantly slower than both. Some reasons for this are discussed below.

5. Discussion

It has been noted that the use of an analog hardware description language for modeling both fault-free and faulty behavior of analog blocks is easier than the alternative: SPICE macromodeling. This is because of the greater flexibility of an HDL compared with the fixed structure of controlled sources and other circuit elements. This strength is, however, also a weakness. The incorporation of a semiconductor model into a

Fault Class B

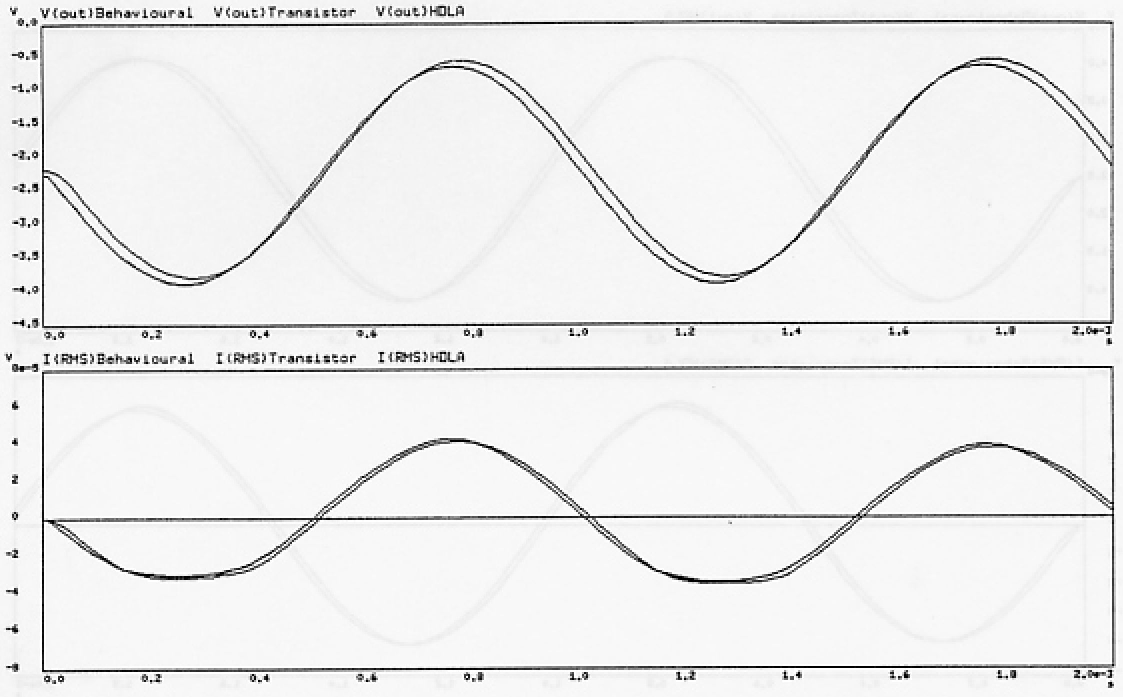


Fig. 11. Fault class B comparison between the transistor model, SPICE macromodel and the VHDL-AMS model.

circuit simulator, such as SPICE, is not a trivial task. Device equations must be formulated. These must be continuous throughout the expected range of behavior of the device. Moreover, it may be possible, during a Newton–Raphson iteration, for a circuit variable to, momentarily, take an unrealistic value. The model must be robust enough to handle this situation. Furthermore, because the Newton–Raphson algorithm uses the partial derivatives of a function with respect to all the controlling variables, these partial derivatives must be implemented and must also be continuous [14]. While VHDL-AMS does not require the user to specify partial derivatives, the other comments regarding model implementation apply. Many of these difficulties become particularly apparent when models are made to operate in ways that were not originally envisaged.

Our experiences with VHDL-AMS have been gained through use of an early VHDL-AMS implementation—the HDL-A modeling language in the ELDO simulator from Anacad. We do not believe, however, that the difficulties we have encountered are necessarily a consequence of one particular simulator, but are likely to be endemic to all analog hardware description languages. The difficulties noted above with device models have also become apparent with VHDL-AMS descriptions. For instance, although it is possible to describe the transfer characteristic of an

Table 4. Comparison of CPU times for different modeling approaches.

	Fault-free	Fault Class A	Fault Class B	Fault Class L
Transistor level	39.0	34.0	35.9	37.3
SPICE macromodel	12.0	12.0	11.9	11.9
VHDL-AMS	55.0	53.0	55.0	57.0

op-amp as a piecewise linear function, it has been found that simply clamping a voltage causes timestep control difficulties.

This was evident in the simulation of the open loop op-amp of Fig. 3. Initially a piecewise linear voltage clamp was used, but this caused numerical oscillation when the input to the clamp changed rapidly. This often occurred when the op-amp, in a closed loop configuration, started to limit the output. As this occurred, the voltage difference between the input terminals would rise rapidly as the feedback no longer kept the negative terminal within a few microvolts of the positive terminal. The limiter was then changed to a hyperbolic cotangent function to smooth the transition from one region to another, which improved the stability, although the problem was not solved completely.

The use of VHDL-AMS was not found to deliver fully all of the desired benefits, namely quicker simulation speed and simpler representations of analog functional blocks. When comparisons between the SPICE and VHDL-AMS macro models of the fault model op-amp were done the VHDL-AMS actually simulated slower. This is almost certainly because the implementation of the simulator used was not fully optimized for VHDL-AMS as it included a full SPICE-compatible simulator as well. The VHDL-AMS parts are simulated using a secant method, which while easier to implement, converges more slowly to a solution at each time step [15]. Similarly, transitions between piecewise linear segments are not limited, which should prevent oscillations. To achieve a major decrease in simulation times the VHDL-AMS model would have to represent a lager circuit and hence a higher level of abstraction that was used here.

Conclusions

Fault simulation of analog circuits is important for confidence in analog and mixed-signal integrated circuits. At present, such fault simulation is of limited use, because of the speed of analog simulation and the large number of faults to be simulated. Simulation can be speeded-up by using simpler, behavioral models. The number of simulations can be reduced by fault collapsing. We have demonstrated that both these techniques yield valuable gains. Nevertheless, SPICE macromodeling is difficult. Analog hardware description languages offer a means to simplify behavioral modeling. The standard for VHDL-AMS is likely to be finalized in 1997, but it is apparent, from our investigations that behavioral modeling in general, and fault modeling in particular can cause difficulties for the underlying simulators. It is not unreasonable to suppose that the robustness and efficiency of VHDL-AMS simulators will improve, but it is still apparent that analog fault simulation *algorithms* need to be developed to support such modeling.

Acknowledgment

This work has been supported by EPSRC grants GR/K32661 and GR/J84120.

Appendix. Derivation of VHDL-AMS Op-amp model

The basic transfer function of the op-amp in Fig. 4 is given by:

$$\frac{V_{out}(S)}{V_{in}(S)} = \frac{(1 - s/z_1)}{(1 + s/p_1)(1 + s/p_2)}$$

Where p_1 and p_2 are the poles and z_1 is the high frequency zero. It is not possible to write the function in s notation directly, as VHDL-AMS only supports differential expressions. Rearranging the above expression, we get:

$$V_{out}(s)\left(1 + \frac{s}{p_1}\right) = V_{in}(s)\left(\frac{1 - s/z_1}{1 + s/p_2}\right) = V_{p1}(s)$$

which can be rewritten as:

$$V_{out}(s) + \frac{1}{p_1}\cdot s \cdot V_{out}(s) = V_{p1}(s)$$

$$V_{p1}(s) + \frac{1}{p_2}\cdot s \cdot V_{p1}(s) = V_{in}(s) - \frac{1}{z_1}\cdot s \cdot V_{in}(s)$$

The inverse Laplace transform of which is:

$$V_{out}(t) + \frac{1}{p_1}\cdot\frac{dV_{out}(t)}{dt} = V_{p1}(t)$$

$$V_{p1}(t) + \frac{1}{p_2} \cdot \frac{dV_{p1}(t)}{dt} = V_{in}(t) - \frac{1}{z_1} \cdot \frac{dV_{in}(t)}{dt}$$

The `ddt()` ''function'' is used to calculate the time derivatives of state variables (quantities), which are saved as new state variables. (Strictly, this is an implicit quantity, according to the proposed 1076.1 standard, and should be written as `v'dot`, not as a function.) Each of these expressions is now stated as an implicit equation. The remainder of the model is concerned with modeling saturation and other non-linear effects.

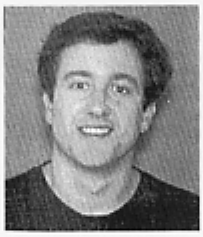
References

1. N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design, A Systems Approach, 2nd Ed.* Addison-Wesley, Reading, MA, 1993, Chapter 7.
2. M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital systems testing and testable design.* Computer Science Press, New York, NY, 1990.
3. C. Di and J. Jess. ''On accurate modeling and efficient simulation of CMOS opens.'' *Proc. IEEE Int. Test Conf.* pp. 875–882, 1993.
4. K. G. Nichols, T. J. Kazmierski, M. Zwolinski, and A. D. Brown. ''Overview of SPICE-like circuit simulation algorithms.'', *IEE Proc.-Circ. Dev. Syst.* 141, pp. 242–250, 1994.
5. D. Rodriguez. ''Analog-VHDL—as an application, a real example.'' *IFIP Trans. A-Comp. Sci. and Tech.* 32, pp. 587–604, 1993.
6. 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* SC-9, pp. 353–363, 1974.
7. C. Chalk and M. Zwolinski. ''Macromodel of CMOS operational amplifier including supply current variation.'' *Electronics Letters* 31, pp. 1398–1400, 1995.
8. T. M. Storey and W. Maly. ''CMOS Bridging Fault Detection.'' *Proc. Int Test Conf.* pp. 842–851, 1990.
9. R. J. A. Harvey, A. M. D. Richardson, E. M. J. G. Bruls, and K. Baker. ''Analogue Fault Simulation Based on Layout Dependent Fault models.'' *Proc. Int. Test Conf.* pp. 641–649, 1994.
10. A. Meixner and W. Maly. ''Fault Modelling for the Testing of Mixed Signal Circuits.'' *Proc. Int. Test Conf.* pp. 564–572, 1991.
11. P. Caunegre and C. Abraham. ''Achieving Simulation-Based Test Program Verification and Fault Simulation Capabilities for Mixed-Signal Systems.'' *IEEE ED&TC*, pp. 469–477, 1995.
12. S. J. Spinks and I. M. Bell. ''Analogue Fault Simulation.'' *IEE Colloquium, Mixed Mode Modelling and Simulation*, No. 1994/205, 1994.
13. Y. K. Malaiya and A. P. Jayasumana. ''Enhancement of resoltion in supply current based testing for large ICs.'' *Proc. IEEE VLSI Test Symp.* pp. 291–296, 1991.
14. T. L. Quarles. ''Analysis of Performance and Convergence Issues for Circuit Simulation.'' Electronics Research Laboratory, University of California, Berkeley, CA, Memo. UCB/ERL M89/4, Chapter 3.
15. L. W. Nagel. ''SPICE2: A computer program to simulate semiconductor circuits.'' Electronics Research Laboratory, University of California, Berkeley, CA, Memo. UCB/ERL M520, Chapter 5.

Brian Wilkins qualified in Electrical Engineering from University College, London. He joined Southampton University in 1965, and is now Senior Lecturer in charge of the Test Engineering Laboratory. For the past 15 years he has been working on various problems associated with testing and DFT, most recent on implementation of boundary scan, testability guidelines, and analog and mixed-signal testing. He is a member of the IEEE Computer Society and a Fellow of the IEE.

Christopher Chalk received his B.Sc. (physics with electronics) from the University of East Anglia (UK) in 1986. In 1994 he received an M.Sc. in electronics from the University of Southampton (UK). He is currently working as a research assistant at the University of Southampton in collaboration with the Universities of Hull and Huddersfield evaluating novel mixed signal test methodologies in addition to studying for a Ph.D. on AC RMS power supply current monitoring. Prior to this, he has held positions at the

British Broadcasting Corporation as a Radio Test Engineer. His current research interests include power supply current monitoring, automatic test pattern generation, behavioral modeling with high level description languages (VHDL-AMS and spectreHDL), built in current sensors and design for testability.



Andrew Perkins is a research assistant in the Electronics and Computer Science Department at the University of Southampton. His research interests include the design and test of analog and mixed signal integrated circuits. He received a B.Eng.(Hons) degree from Manchester Polytechnic and an M.Sc. from the University of Southampton.

Mark Zwolinski gained his B.Sc. and Ph.D. from the University of Southampton. He has been a lecturer in the Department of Electronics and Computer Science at the University of Southampton since 1990. His research interests include mixed-signal simulation, mixed-signal test and synthesis. He has published over 30 papers in the fields of design automation and test and is co-author of a book on circuit simulation. He is a member of the IEE, the IEEE, ACM and is a Chartered Engineer. He is also a member of the IEEE Computer Society Test Technology Committee and the ACM Special Interest Group on Design Automation.