Fast, Robust DC And Transient Fault Simulation For Nonlinear Analogue Circuits

Z.R. Yang and M.Zwolinski
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ
mz@ecs.soton.ac.uk

Abstract

The evaluation of testing and design for test strategies for analogue and mixed-signal circuits requires efficient analogue fault simulation. By analogy with digital fault simulation, concurrent analogue fault simulation has been proposed to reduce simulation times by avoiding repeated construction of the circuit matrix. Simulation efficiency can be improved by dropping non-convergent faults and by fault collapsing. A robust, fast algorithm for concurrent analogue fault simulation is presented in this paper. Three techniques for automatic fault collapsing and dropping are addressed: a robust closeness measurement technique; a late start rule and an early stop rule. The algorithm has been successfully applied to both DC and transient analyses. A significant increase in the speed of analogue fault simulation has been obtained.

1. Introduction

Fault simulation of analogue integrated circuits has been of considerable interest in recent years. Fault modelling and simulation with SPICE has been necessary for the evaluation of design for test strategies. Normally, all possible faults have to be simulated, which is very time-consuming, particularly when parametric variations are taken into account. Hence efforts have been made to reduce the size of the fault lists and to use macro modelling. Some recent work has concentrated on speeding up the simulation algorithms themselves, and it is with this topic that this paper is concerned.

In [1] it was proposed that analogue fault simulation could be speeded up using techniques analogous to those used in concurrent digital fault simulation. The fault-free and faulty simulations would be performed concurrently in a single simulation run. Suppose that all or part of a faulty version of a circuit were to perform in the same way as the fault-free version. By checking the terminal voltages of semiconductor devices in both the faulty and fault-free versions of the circuit at each Newton-Raphson iteration at each time point in a transient analysis, redundant evaluations of device equations could be avoided. The evaluation of semiconductor device equations may take 60% or more of the simulation time, hence significant speed improvements could, in principle, be obtained.

The work reported in [3] partitioned the circuit matrix into two parts, a constant part and a variable part. The variable part is normally much smaller than the constant. The simulation time can therefore be reduced if the simulation time spent in circuit matrix construction could be significantly reduced.

In [5] the faults were ordered at each simulation iteration so that the initial guess for the fault simulation can be well defined and the total fault simulation iterations can be reduced to a significant level.

The work reported in [4] aimed to reduce fault simulation time by fault collapsing and resulted in a large reduction of fault simulation time.

Concurrent fault simulation for analogue circuits has been considered in [1], [6]. For efficient simulation it is necessary to know where and when to automatically drop faults from the fault list. The key to fault collapsing for analogue circuit simulation is to measure the closeness between the fault-free circuit and the faulty circuits. As will be discussed in this paper, a single-point closeness measurement [5] is not reliable. A robust fault collapsing technique is therefore proposed based on a multi-point closeness measurement. In general, the earlier the fault collapsing, the better the algorithm. However, there is always an unstable stage in simulating analogue circuits, in particular, during DC analysis, because of the iterative methods employed. At a very early stage of the simulation, a circuit's state can be misleading. Hence, the selection and determination of a suitable point for taking the closeness measurement is very important. We call this a *late start* rule. We also know that different fault simulations converge at different rates in DC analysis. Applying the same stop rule to all the fault simulations would inevitably waste resources. It is, therefore, necessary to apply an *early stop* rule to fault simulations.

A robust and fast concurrent fault simulator has been implemented. This paper is organised as follows. The next section has a discussion of the closeness measurement. The third section gives the algorithm for concurrent fault simulation. The fourth section has some examples.

2. Closeness measurement

2.1 Single-point measurement

In order to reduce the computational effort needed for fault simulations, similarities between simulations of faulty circuits and of the fault-free circuit have to be identified. To do this, a measure of closeness is needed. For two vectors of real numbers, the closeness can be measured by an absolute distance measurement [2], such as the Euclidean distance, the Hamming distance or the Manhatten distance.

The Euclidean distance is the most common method for measuring the distance and was the approach adopted in [6], with very limited success. The Hamming distance is useful for measuring the distance between vectors which only contain binary numbers. The Manhatten distance is measured between two pattern swarms.

2.2 Multi-point measurement

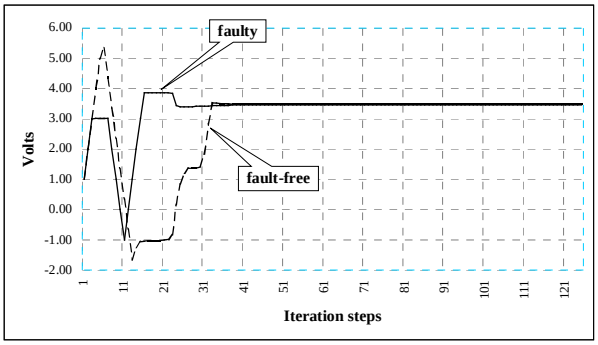

Figure 1. An illustration of non-constant distance between a faulty circuit and the fault-free circuit.

All these distance calculation methods use two single points (Euclidean and Hamming distance) or pattern swarms (Manhatten distance). It is doubtful that they can accurately highlight the closeness between faulty circuits and the fault-free circuit because the distance will not be always the same during a simulation. Figure 1 illustrates this. The vertical axis indicates the response (volts) and the horizontal axis denotes the iteration steps. The solid line shows the response of a faulty circuit and the dotted line that of the fault-free circuit. It can be seen that the distance between the faulty circuit and the fault-free circuit is not the same between the 1st and the 40th iterations, although at several iterations the two curves coincide. Moreover, the faulty circuit and the fault-free circuit behave identically for a few iterations. Therefore we replace the single-point closeness measurement by a multi-point closeness measurement, which is conducted during M iterations:

$$\hat{d}_f^E = \frac{1}{M}\sum_{m=1}^{M}\left\|\mathbf{x}_0^m - \mathbf{x}_f^m\right\|,$$

where $\mathbf{x}_0^m$ and $\mathbf{x}_f^m$ are the responses at the m^{th} Newton-Raphson iteration for the fault-free circuit and the f^{th} faulty circuit respectively. M is the user-defined continuous steps for closeness measurement and $\hat{d}_f^E$ is the closeness measurement.

2.3 The late start rule

In order to obtain convergence, particularly in DC analysis, the Newton-Raphson algorithm is commonly damped. This damping may be linear or non-linear. For example, if x^m is the value of a node voltage at the m^{th} Newton-Raphson iteration, Δx^m is the calculated increment and α is the largest change voltage allowed, e.g. 1 volt, the new value of the node voltage, x^{m+1}, is incremented by the smaller of $\left|\Delta x^m\right|, \alpha$.

Using damped values for the closeness measurement is misleading. Figure 2 shows a typical response curve. It can be seen that after 13 iterations, the response starts to change more gradually. We define this point (13th iteration) as a *stable point* because the response at the node will not subsequently dramatically change.

Unfortunately, not all the responses at the nodes of a circuit will simultaneously arrive at their stable points. Because the change before a stable point is not a real change, using these damped results will result in misleading measurements. We therefore start the closeness measurement after most nodes in a circuit arrive at their stable points. This is the *late start rule*.

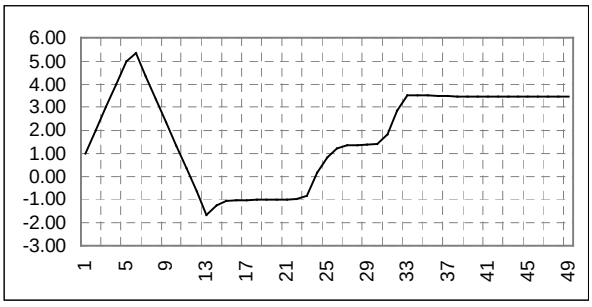

Figure 2. Illustration of a stable point. At the early stages of simulation, the change of node voltage has been damped to 1 volt.

3. Concurrent fault simulation using the robust fault collapsing method

3.1 The early stop rule

During concurrent analogue fault simulation, in particular, during DC analysis, the fault simulations do not all converge at the same iteration step. Employing the same stop rule for all the fault simulations during concurrent simulation will hide the advantages of concurrent fault simulation. The *early stop rule* proposed here is very simple in that whenever one fault simulation converges, that fault simulation stops while other fault simulations carry on. Convergence of one fault simulation is measured in the usual way [7] by

$$\tilde{\varepsilon} = \frac{\left| x^{m+1} - x^{m} \right|}{\alpha \cdot \max(x^{m+1}, x^{m}) + \eta}$$

where, α and η are two user-defined coefficients, x^m is a solution of a nonlinear equation at the m^{th} iteration and x^{m+1} is a solution of a nonlinear equation at the $(m+1)^{\text{th}}$ iteration. When $\tilde{\varepsilon} < 1$, the simulation has converged.

3.2 Concurrent fault simulation algorithm

From the above, we have built a robust fault collapsing method for fast and efficient concurrent analogue fault simulation. Let $\mathbf{G}_f$ represent the circuit description for the f^{th} fault, and Ω^m, the fault list at the m^{th} iteration. A fault, $\mathbf{G}_f \in \Omega^m$, can be dropped from the fault list

$$\Omega^{m+1} = \Omega^m - \mathbf{G}_f$$

if $\mathbf{G}_f$ behaves "similarly" to the fault-free circuit,

$$\hat{d}_f^E = \frac{1}{M} \sum_{p=m-M-1}^{m} \left\| \mathbf{x}_0^p - \mathbf{x}_f^p \right\| < T,$$

where T is the threshold for the minimum distance.

The algorithm for concurrent analogue fault simulation has been embedded into our own analogue circuit simulator. The sequence of actions for DC analysis or at one time point in transient analysis is as follows:

Step 1. Constitute the original fault list, $\Omega^0 = \{\mathbf{G}^1, \mathbf{G}^2, \cdots, \mathbf{G}^N\}$, by inserting all possible (N) faults into copies of the circuit. We define $\mathbf{x}_0^m, m \in [0, \infty)$ as the response vector for the fault-free circuit and $\mathbf{x}_f^m, m \in [0, \infty)$ as the response vector of the f^{th} faulty circuit at the m^{th} iteration during the fault simulation.

Step 2. Pre-simulate for M Newton-Raphson iterations until most nodes arrive at their stable points.

Step 3. Conduct the m^{th} iteration of the fault simulation for all the faulty circuits, $\mathbf{G}_f^m \in \Omega^{m-1}$, as well as for the fault-free circuit ($m>M$). After the simulation, $\mathbf{x}_0^m$ and $\mathbf{x}_f^m \left| \mathbf{G}_f^m \in \Omega^{m-1} \right.$ are available.

Step 4. Carry out a multi-point closeness measurement for all the faulty circuits, which are still in the fault list, Ω^{m-1}. A new fault list is generated by

$$\Omega^m = \Omega^{m-1} - \tilde{\Omega}^m,$$

where

$$\tilde{\Omega}^m = \left\{ \mathbf{G}_f \left| \hat{d}_f^E = \frac{1}{M} \sum_{p=m-M-1}^{m} \left\| \mathbf{x}_0^p - \mathbf{x}_f^p \right\| < T \right. \right\}$$

is the set of all the "close" faults, that have to be dropped from the fault list.

Step 5. Apply the early stop rule to $\mathbf{G}_f^m \in \Omega^m$. If $\tilde{\varepsilon}^f < 1$, the f^{th} fault simulation has converged. Hence $\Omega^m = \Omega^m - \mathbf{G}_f^m$.

Step 6. If one or more of the fault simulations has not converged, go back to step 3, otherwise, stop.

4. Examples

Three example circuits were simulated using DC and transient to evaluate:

- whether concurrent fault simulation is faster than separate fault simulations;
- whether fault collapsing can further save fault simulation time; and
- whether the algorithm is reliable.

The first two points can be assessed by comparing the CPU time and the last can be assessed by whether the early dropping of faults retains the accuracy.

The example circuits used were a two-stage bipolar amplifier, composed of six bipolar transistors and ten resistors; a differential amplifier, composed of four bipolar transistors and five resistors; and a CMOS two-stage amplifier with nine transistors.

4.1 DC analysis

We inserted 36 faults into the two-stage BJT amplifier. The DC fault simulations for three faults did not converge. Of the remaining 33 faults, there were 15 short faults, which are all the possible short-circuits across the terminals of the six transistors, nine short-circuits across the resistors and nine parametric faults in the resistors (reducing the resistance values by 50%). We conducted separate fault simulations and a concurrent

fault simulation. In the separate fault simulations, the faulty and the fault-free circuits simulated one by one. The times of the separate fault simulations are listed in Table 1.

Table 1. The time behaviour of separate fault simulations for BJT two-stage amplifier

no.	type	iterations	cpu(ms)
0	Fault-free	43	70
1	short	26	30
2	short	44	70
3	short	31	60
4	short	27	40
5	short	26	40
6	short	24	40
7	short	36	40
8	short	31	50
9	short	51	80
10	short	44	70
11	short	41	70
12	short	30	50
13	short	-	-
14	short	55	90
15	short	23	40
16	short	24	30
17	short	33	50
18	short	32	60
19	short	43	70
20	short	38	40
21	short	116	150
22	short	55	90
23	short	-	-
24	short	31	60
25	short	49	80
26	short	46	70
27	parameter	47	70
28	parameter	44	70
29	parameter	43	60
30	parameter	34	50
31	parameter	125	170
32	parameter	43	60
33	parameter	50	80
34	parameter	39	60
35	parameter	-	-
36	parameter	43	70
Total			2230

It can be seen that the separate fault simulations need 2230 ms in total. In contrast only 1810 ms are required by the concurrent fault simulation giving a 19% reduction in the CPU time.

If the early stop rule is not used, the concurrent simulation needs 3350 ms (50% increase in CPU time). This is because a few fault simulations require more than 100 Newton-Raphson iterations, as shown in Table 1. Most other fault simulations converge in fewer than 50 iterations, but without the early stop rule have to be iterated to the limit.

If fault collapsing is applied with the threshold value set to 0.1 volt and using a 3-point measurement can further reduce the CPU time to 1590 ms or a further 12% giving a total of 29% saving in CPU time compared with separate fault simulation.

Similarly, we inserted 19 faults into the differential BJT amplifer and 24 faults into the CMOS opamp. One fault in the CMOS opamp simulations failed to converge. Again we performed separate fault simulations and a concurrent fault simulation. The CPU times required are summarised in Table 2.

Table 2 CPU times in ms for DC fault simulation of example circuits.

Circuit	Separate	Concurr-ent	Early Stop	Collaps-ing	Saving
2-stage BJT	2230	3350	1810	1590	29%
Diff BJT	160	160	80	70	56%
CMOS	3270	3830	3200	2750	16%

Table 3. Fault Simulation results for DC analysis of BJT two-stage amplifier

no.	ε^f	collapse iteration	no.	ε^f	collapse iteration
1	0.377	37	19	0.000	35
2	0.211	35	20	0.829	
3	1.217		21	0.490	
4	1.322		22	0.000	22
5	1.278		23	-	
6	0.028	35	24	1.017	
7	0.540		25	1.984	
8	0.751		26	2.031	
9	0.363	35	27	0.518	43
10	0.007	35	28	0.126	38
11	0.144	22	29	0.000	35
12	0.883		30	0.302	36
13	-		31	0.024	122
14	2.783		32	0.000	22
15	1.008		33	0.664	22
16	0.793		34	0.470	
17	1.288		35	-	
18	0.744		36	0.411	22

This shows that concurrent fault simulation is faster than separate fault simulation. provided that the early stop rule and fault collapsing are applied.

For DC analysis, the accuracy of fault collapsing is measured by the difference between the fault-free circuit and faulty circuits on the final results

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_i^f = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{x}^0 - \mathbf{x}_i^f\right\|.$$

where, N is the number of the faults that have been collapsed. Table 3 lists the results for the BJT two-stage amplifier. The second column is the difference between the fault-free circuit and faulty circuits. The final column gives the iteration, at which the fault is collapsed. The final error is 0.198, which is larger than the threshold (0.1), but is still small.

4.2 Transient analysis

The same circuits were fault simulated in transient analyses. Convergence could not be reached for certain faults. Table 4 summarises the CPU times required for convergent faults.

Table 4 CPU times in seconds for transient fault simulation of example circuits.

Circuit	Separate	Concurr-ent	Collaps-ing	Saving	No. faults collapsed
2-stage BJT	571	490	305	46%	15
Diff BJT	185	168	107	42%	8
CMOS	2655	2000	1041	61%	9

Table 5. Fault simulation results for transient analysis of BJT differential amplifier

no.	ε	collapse time	no.	ε	collapse time
1	2.404		11	1.311	
2	-		12	0.389	1.1340e-07
3	-		13	0.035	5.5000e-10
4	2.200		14	9.430	
5	-		15	0.538	1.1535e-08
6	-		16	0.656	4.1250e-08
7	9.430		17	0.199	1.0510e-07
8	0.936	1.0510e-07	18	0.018	5.5000e-10
9	2.078		19	1.707	
10	0.075	1.2480e-08			

Concurrent fault simulation is faster than separate fault simulations when fault collapsing is used. For transient analysis, the accuracy of fault collapsing is compared using the difference between the fault-free circuit and faulty circuits on the whole range of analysis

$$\varepsilon^f = \frac{1}{T}\sum_{t=1}^{T}\left\|\mathbf{x}_t^0 - \mathbf{x}_t^f\right\|.$$

Table 5 lists the results for the differential amplifier simulated for 100 μs. The second column shows the greatest instantaneous difference between the faulty and fault-free circuits over the whole simulation period. The third column gives the time at which fault collapsing occurs. It can be seen that in the worst case, the collapsed faults differ from the fault-free circuit behaviour by less than a volt.

5. Summary and further work

A robust, fast concurrent analogue fault simulation algorithm has been proposed and verified in this paper. For DC analysis, a 56% reduction in CPU time can be achieved, while 61% of CPU time can be saved for the case of transient analysis. It should be noted that this achievement depends on circuit types on which different circuits have different degree of fault diversity. In the sense of fault collapse, we should emphases on what we can do rather than what the average achievement. The algorithm has been realised in C on a SUN UltraSPARC under Solaris. The implementation of this algorithm is simple and the whole algorithm has been embedded in a circuit simulator. The most significant difference of this algorithm to traditional analogue fault simulation is that fault collapsing has been realised as an automatic mechanism embedded into the circuit simulator.

The example circuits used have been small and have not contained hierarchy. It is expected that larger, hierarchical circuits will exhibit more significant savings.

Further work will focus on how to embed the clustering technique to realise comprehensive fault collapsing and how to automatically drop non-converging faulty circuits from the fault list during a concurrent fault simulation.

Acknowledgments

This work has been supported by EPSRC grant GR/L35829.

References

1. M.Zwolinski, “Relaxation Methods for Analogue Fault Simulation”, 20th Int. Conf. Microelectronics (MIEL’95), Niš, Serbia, pp. 467-471, 1995.
2. G. B. Bruce, Pattern Recognition, Ideas in Pratice, Plenum Press, 1978.
3. J. Hou and A. Chatterjee, “CONCERT: a concurrent fault simulator for analog circuits”, 4th IEEE International Mixed-Signal Testing Workshop, June 8th-11th, 1998, Atlantic Hotel, The Hague, The Netherlands,1998.
4. 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, no. 2, pp. 141-156, 1998.

5. C. J. Shi and M. W. Tian, “Efficient DC fault simulation of nonlinear analog circuits”, Design, Automation and Test in Europe Conference (DATE ’98), Paris, France, Feb. 23-26, 1998.
6. M. Zwolinski, A. D. Brown and C. D. Chalk, "Concurrent analogue fault simulation", pp. 42-47, in Proc. 3rd IEEE International Mixed-Signal Testing Workshop, 1997.
7. J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, New York, 1983.