Efficient DC Fault Simulation of Nonlinear Analog Circuits

Michael W. Tian

Dept. of Electrical and Computer Engineering
University of Iowa, Iowa City, IA 52242, USA
Tel: +1-319-335-5961, Fax: +1-319-335-6028
e-mail: wtian@eng.uiowa.edu

C.-J. Richard Shi

Dept. of Electrical and Computer Engineering
University of Iowa, Iowa City, IA 52242, USA
Tel: +1-319-384-0530, Fax: +1-319-335-6028
e-mail cjshi@eng.uiowa.edu

Abstract— This paper describes a method to improve the efficiency of nonlinear DC fault simulation. The method uses the Newton-Raphson algorithm to simulate each faulty circuit. The key idea is to *order* the given list of faults in such a way that the solution of previous faulty circuit can serve as a *good* initial point for the simulation of the next faulty circuit. To build a good ordering, one step Newton-Raphson iteration is performed for all the faulty circuits once, and the results are used to quantify how faulty circuits and the good circuit are close in their behaviors. With one-step Newton-Raphson iteration implemented by Householder's formula, the proposed method has virtually no overhead. Experimental results on a set of 36 MCNC benchmark circuits show an average speedup of 4.4 and as high as 15 over traditional stand-alone fault simulation.

I. Introduction

Testing analog portion of mixed-signal circuits and systems is becoming an important issue that affects both the time-to-market and final product cost. Among several methods of analog testing, DC testing is generally cheaper, since it does not require expensive test equipment and has a shorter testing time. In the scenario where DC testing is performed before AC testing and transient testing, DC tests that detect the majority of faults can reduce the overall testing time and cost. Recently, some research effort has been devoted to exploit low-cost DC test generation [6] and DC built-in self-test [1].

DC fault simulation is to simulate the DC behavior of an analog circuit for a given list of faults. It is an important tool for fault-coverage analysis, test generation, and built-in self-test. A considerable amount of effort has been devoted to efficient fault simulation for linear analog circuits [7, 10, 14, 15]. However, DC fault simulation of nonlinear analog circuits—a more difficult and more practically important problem—remains largely unexplored. In general, DC simulation of nonlinear analog circuits amounts to solving a set of nonlinear algebraic equations usually by Newton-Raphson iterative algorithm. This itself is computationally expensive. Further, faults may deteriorate the convergence property. We note some recent progresses in reducing the number of faults to be simulated by inductive fault analysis [12], minimizing the simulation complexity by behavioral modeling [9], and shortening the equation setup time by using the cache mechanism [17].

This paper presents an efficient method for DC fault simulation of nonlinear analog circuits. We propose to *order* the faults in such a way that the circuit response of the previous fault can serve as a "good" initial point for the simulation of next fault. Fault ordering is performed by a greedy heuristic, and is based on predicting the "closeness" of two faults using one step iteration of the Newton-Raphson algorithm.

This paper is structured as follows: Section II presents the basic ideas of the fault simulation continuation. Section III describes how the ideas are realized via one-step relaxation and greedy fault ordering. Section IV describes implementation details. Experimental results are reported in Section V.

II. Simulation Continuation—Why Fault Ordering?

In general, DC simulation amounts to solving a system of nonlinear equations written as:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \tag{1}$$

where $\mathbf{x}$ is the vector of circuit unknowns (node voltages and branch currents), $\mathbf{f}$ represents the system of nonlinear functions.

The widely used *Newton-Raphson* algorithm for solving Eq. (1) is an iterative process. It consists of the following steps:

- Guess an initial point $\mathbf{x}^{(0)}$,
- Solve $\mathbf{J}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = -\mathbf{f}(\mathbf{x}^{(k)})$ for $k = 0, 1, 2, \ldots$, until the convergence criteria is met.

Here $\mathbf{J}(\mathbf{x}^{(k)})$ is the *Jacobian* matrix of $\mathbf{f}(\mathbf{x}^{(k)})$.

How close the initial point $\mathbf{x}^{(0)}$ to the final solution is important for whether the algorithm converges, and how fast it converges [2]. However, predicting a good initial point in general is a difficult problem. In SPICE, the default initial point is set to $\mathbf{0}$.

Homotopy simulation continuation is an important technique to improve the convergence property of difficult-to-converge circuits [4, 11, 16]. The basic idea is to construct a slightly different "pseudo" circuit whose response can serve as a *good* initial point for the simulation of the original circuit, while the convergence of the "pseudo" circuit simulation is easy to achieve. Two simple schemes known as *Gmin stepping* and *source stepping* were implemented in SPICE [4, 8], while more complicated schemes involve the design of circuit traces (a list of "pseudo" circuits), known as *homotopy* [11, 16].

We propose to exploit this idea for DC fault simulation. Given a list of faults, fault simulation is performed in an order, and the simulation result of the previous fault serves as the initial point of next fault simulation. The faults are ordered in such a way that the total number of iterations is minimized. In case of the difficult-to-converge fault circuits, more "pseudo" faults can be embedded into the simulation sequence to help convergence.

This idea is especially attractive for parametric faults and some structural faults that do not cause the catastrophic failure of the circuit. For catastrophic faults with dramatically different responses, $\mathbf{0}$ or the good circuit response $\mathbf{x}_{good}$ can be used as the initial point for fault simulation. This consideration leads to the simulation trace as illustrated in Fig. 1, where solid circles correspond to fault responses, empty circles correspond to some "pseudo" fault responses added to help the convergence, and two shadowed triangle points denote the good circuit response and $\mathbf{0}$ respectively.

III. Fault Ordering via One-Step Relaxation

In this section, we consider how to construct appropriate trace of fault simulation, or simply *fault ordering*. A computationally efficient yet effective heuristic is proposed. It consists of three components. First, we propose to perform one-step Newton-Raphson relaxation for each fault using the good-circuit response $\mathbf{x}_{good}$ as the initial point, and to use the results as estimations of the fault-circuit responses. This can be implemented very efficiently using Householder's formula. Second, we give a simple and effective formula to "quantify" the closeness of two responses. Finally, we describe a greedy heuristic to order the faults for simulation continuation.

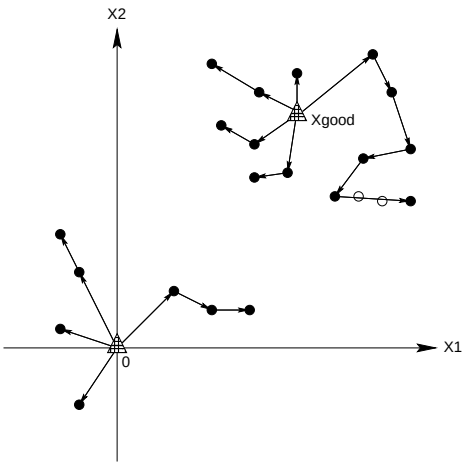


Fig. 1. Illustration of fault simulation continuation.

A. One-Step Relaxation via Householder's Formula

Let $\mathbf{x}_{good}$ denote the good-circuit response, and $\mathbf{x}_{fault}$ the fault-circuit response. *One-step Newton-Raphson relaxation* is to perform one step Newton-Raphson iteration on the faulty circuit using the good-circuit response as the initial point:

$$\mathbf{J}_{fault}(\mathbf{x}_{good})(\mathbf{x}^{(1)}_{fault} - \mathbf{x}_{good}) = -\mathbf{f}_{fault}(\mathbf{x}_{good}) \qquad (2)$$

Here $\mathbf{J}_{fault}(\mathbf{x}_{good})$ and $\mathbf{f}_{fault}(\mathbf{x}_{good})$ are computed from the initial point $\mathbf{x}_{good}$, and $\mathbf{x}^{(1)}_{fault}$ represents the vector to be solved. We propose to use $\mathbf{x}^{(1)}_{fault}$ as the estimation of the fault-circuit responses for fault ordering.

In case that a fault is associated with a linear element (e.g., linear resistor) in a circuit, one-step Newton-Raphson relaxation has the following simple circuit interpretation: take the linearized circuit at the operating point of the good circuit as the "good circuit" model and consider fault simulation of the given fault. This is a linear fault simulation problem. The solution $\mathbf{x}^{(1)}_{fault}$ of one-step relaxation is the fault simulation result of the linearized circuit.

In fact, the solution $\mathbf{x}^{(1)}_{f}$ can be computed very efficiently without solving Eq. (2). This is to exploit the fact that the matrix of a faulty circuit differs only slightly from that of the good circuit. Hence Householder's matrix updating formula can be used [3].

Consider a resistor r_{ij} connected between node i and j with the conductance value g_{ij}, and a fault is associated with the resistor. Let $\mathbf{Y}_g$ be the MNA matrix of

the good circuit, and $\mathbf{w}_f$ be the right-hand-side vector of a faulty circuit. Let $\mathbf{I}$ be the identity matrix, and vector $\mathbf{e}$ be a column vector defined by

$$e_k = 0, \text{except } e_i = 1 \text{ and } e_j = -1 \quad k = 1, 2, \ldots, n. \quad (3)$$

Then from Householder's inverse matrix formula [10], we have:

- Parametric faults. Let a parametric fault cause the change of the resistor's conductance by Δg_{ij}, then

$$\mathbf{x}_f^{(1)} = (\mathbf{I} - \frac{\mathbf{Y}_g^{-1}\mathbf{e}\mathbf{e}^T}{(\Delta g_{ij})^{-1} + \mathbf{e}^T\mathbf{Y}_g^{-1}\mathbf{e}})\mathbf{Y}_g^{-1}\mathbf{w}_f \quad (4)$$

- Open Fault. We can model a resistor open fault by setting Δg_{ij} to $-g_{ij}$, then

$$\mathbf{x}_f^{(1)} = (\mathbf{I} - \frac{\mathbf{Y}_g^{-1}\mathbf{e}\mathbf{e}^T}{-g_{ij}^{-1} + \mathbf{e}^T\mathbf{Y}_g^{-1}\mathbf{e}})\mathbf{Y}_g^{-1}\mathbf{w}_f \quad (5)$$

- Short fault. We can model a resistor short fault by setting Δg_{ij} to ∞, then

$$\mathbf{x}_f^{(1)} = (\mathbf{I} - \frac{\mathbf{Y}_g^{-1}\mathbf{e}\mathbf{e}^T}{\mathbf{e}^T\mathbf{Y}_g^{-1}\mathbf{e}})\mathbf{Y}_g^{-1}\mathbf{w}_f \quad (6)$$

It is important to note that the simplified equations Eqs. (4)–(6) not only reduce the CPU cost of device loading and LU decomposition, but also avoid possible numerical difficulties caused by ∞ and **0** resistors.

B. Closeness Measurement

Given two non-zero circuit responses $\mathbf{x}_i$ and $\mathbf{x}_j$, we define the *normalized absolute distance* as

$$t_{ij} = \frac{1}{n}\sum_{k=1}^{n} |\frac{(\mathbf{x}_i)_k - (\mathbf{x}_j)_k}{(\mathbf{x}_i)_k}|, \quad (7)$$

where n is the dimensions of vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and $(\mathbf{x}_i)_k$ and $(\mathbf{x}_j)_k$ denote the k-th element of the vector $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. We denote the distance between $\mathbf{x}_{good}$—the response of the good circuit—and $\mathbf{x}_j$—the response of j-th faulty circuit—by t_j.

Having tested several other measurements, we have found out that the normalized absolute distance is a simple yet effective measurement of the "closeness" of two faults. Intuitively, if two faults are close in this distance, they requires less iterations if they are simulated with continuation from one to another. Note that the distance between any response and **0** is always 1.

DC_FAULT_SIMULATION_CONTINUATION

```
1   x_good ← result of good circuit simulation;
2   for the ith fault in the fault list (i = 1, ..., m)
3       x_faulti ← result of one step relaxation;
4       t_i ← distance between x_good and x_faulti;
5   if the kth fault has the minimum distance, then
6       exchange the positions of first & kth faults;
7   if t_1 > 1, then
8       x_initial ← 0;
9   else
10      x_initial ← x_good;
11  for the ith fault in the fault list (i = 1, ..., m)
12    x_faulti ← simulation result of the ith fault
            using x_initial as initial point;
13    t_ij ← distance between x_faulti and x_faultj
            j = i+1, ..., m;
14    Let t_ik be the minimum distance;
15    if t_ik > 1 and t_ik > t_i+1, then
16      exchange the positions of i+1th & kth faults;
17      x_initial ← x_faulti;
18    elseif t_ik > t_i+1 and t_i+1 < 1, then
19      x_initial ← x_good;
20    elseif t_ik > 1 and t_i+1 > 1, then
21      x_initial ← 0;
```

Fig. 2. DC fault simulation algorithm.

C. Adaptive Fault Ordering Heuristic

Figure 2 describes the complete fault-simulation continuation algorithm. It is based on an *adaptive* and *greedy* fault ordering strategy. First, one-step Newton-Raphson relaxation is performed for all the faults, the results are denoted as $\mathbf{x}_{faulti}^{(1)}$, $i = 1, 2, \ldots, m$. The normalized absolute distances between fault-circuit responses and the good-circuit response are computed and denoted as t_i, $i = 1, 2, \ldots, m$. The fault that has the minimum distance will be selected as the first fault to simulate. If its distance is less than 1, then $\mathbf{x}_{good}$ is used as the initial point; otherwise **0** is used. This process is repeated for all the faults in the list. Each time, the distances from the result of previous simulated fault to that of all the un-simulated faults (approximated by the one-step relaxation method) are calculated. The fault to be simulated next and its initial point are determined based on the computed distances.

IV. Implementation and Experimental Setting

The proposed method has been implemented into SPICE3F5. A set of benchmark circuits from 1990 MCNC Circuit Simulation and Modeling Workshop [5]

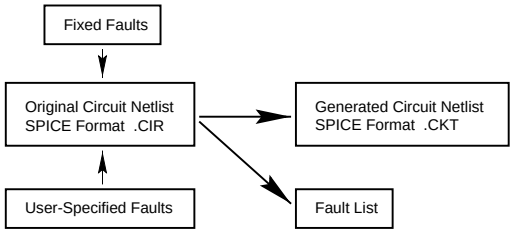


Fig. 3. Diagram of DC-fault-list generator.

TABLE I
SUMMARY OF FAULTS HANDLED BY THE FAULT-LIST GENERATOR.

Device	Fixed Faults	User-Specified Faults
Resistor	short, open	parameter deviation
Diode	short, open	model parameter deviation
BJT	open at C, B, E	model parameter deviation
	short between C-B, C-E and B-E	pipe between E-C
MOS	open at D, G, S	model parameter deviation
	short between D-G, D-S and G-S	pinhole between G-S, G-D and G-channel
Interconnect	—	short, open

are used to test our method. We have also implemented an automatic fault-list generator to insert faults into the benchmark circuits.

Figure 3 describes the structure of our fault-list generator. It supports two methods of fault injection: *fixed* and *user-specified*. Fixed faults are device *hard* (*i.e.*, open or short) faults. For examples, a resistor can have 1 short fault and 1 open fault; a transistor has a total of 6 short/open faults. For an MOS transistor, there are 3 shorts (between gate-drain, gate-source, drain-source), and 3 opens (at gate, drain, source).

User-specified faults include *parametric* faults, and global interconnect short/open faults. Table I summaries the faults handled by the fault-list generator.

In our experiments, the following set of user-specified faults are injected into all of the 36 test circuits:

- Resistors: $\pm 10\%$ to $\pm 90\%$ parameter deviations with 10% incremental.
- BJT transistors: Emitter-collector pipe is modeled by 500Ω to $5K\Omega$ resistors with 500Ω incremental.
- MOS transistors: ± 10 to $\pm 90\%$ Width/Length ratio deviations with 10% incremental. Pinholes between G-S, G-D are modeled by 500Ω to $5K\Omega$ resistors with 500Ω incremental.

In the current implementation, the fault-list generator does not inject faults into sub-circuits described in the SPICE netlist. Therefore, the total number of faults is not proportional to the number of devices for the benchmark netlists containing sub-circuits.

We note that in order to simulate some faults, especially shorts, extra circuit nodes have to be added into the original netlist. Therefore, the fault-list generator creates not only the fault list, but also the expanded circuit netlist in the SPICE format. In practice, it is recommended that the *inductive fault analysis* (IFA) be used to generate the realistic set of faults [12]. In the absence of layout and process data, inductive fault analysis is not used in our experiments.

V. Experimental Results

The statistics of benchmark circuits are summarized in the first 5 columns of Table II. There are 36 transistor-level circuits. The first group (*astabl* to *vreg*) has 9 BJT circuits, the second group (*ab_ac* to *toronto*) has 16 MOS circuits using the SPICE level-2 MOS model, and the third group 11 MOS circuits using the SPICE level-3 model.

In Table II, Column 6 lists the number of iterations required to simulate each circuit without faults (good circuit simulation). Columns 7 and 8 describe, respectively, the total number of iterations and the CPU time used for simulating all the faults using $\mathbf{0}$ as the initial point (stand-alone fault simulation). Columns 9 gives the total number of iterations used for simulating all the faults using $\mathbf{x}_{good}$ (the result of the good circuit simulation) as the initial point. Column 10 describes the ratio of Column 9 with respect to Column 6, which is the speed up of using $\mathbf{x}_{good}$ as the initial point over stand-alone fault simulation. Columns 11 and 12 describe the total number of iterations required by the proposed simulation-continuation method and its speedup over stand-alone fault simulation. The CPU time is collected on an UltraSparc-I workstation.

We note that the speedup depends heavily on each individual circuit and its injected faults. The proposed method achieved an average speedup of 4.4, and as high as 15, over stand-alone fault simulation. If the good circuit response is used as the initial point, an average speedup of 2.7 was achieved. We have observed that the proposed method offers a significant improvement

- when the number of iterations required per circuit simulation is substantial,
- when a circuit has many parametric faults to simulate.

In our experiments, most faults are catastrophic.

The static latch circuit in Table II is used to illustrate the results of one-step relaxation and its comparison with accurate fault simulation. Figure 5 plots the output voltage and supply current computed by one-step relaxation and accurate fault simulation for a list

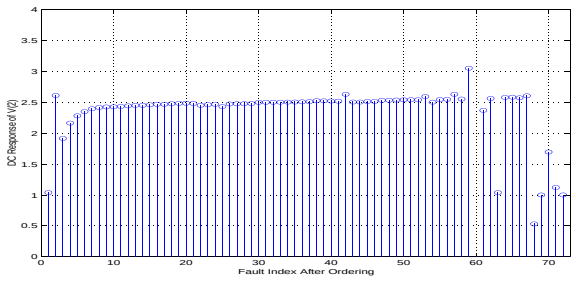


Fig. 4. One-step relaxation results after ordering.

of 72 faults. It can be seen that the results of one-step relaxation approximate well that of accurate fault simulation.

Figure 4 plots the result of one-step relaxation where the list of faults are ordered by the heuristic described in the paper. It can be observed that the ordered results show a better continuity: the results of two successive simulations are closer to each other.

For this circuit, the total numbers of iterations required by stand-alone fault simulation, fault simulation using $\mathbf{x}_{good}$ as the initial point, and the proposed fault simulation continuation (with ordering) method are 1083, 667 and 573, respectively. If the simulation continuation technique is applied directly to the given fault list without performing fault ordering, the number of iterations is 1415.

VI. Conclusions

A simulation-continuation method has been presented for DC fault simulation of nonlinear analog circuits. The method uses the result of one step Newton-Raphson iteration (called one-step relaxation) to predicate the faulty response for each fault, and then uses a greedy heuristic to build the fault order for simulation continuation. Experimental results have demonstrated the effectiveness of the proposed method.

Acknowledgments

This research is sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA) under grant number F33615-96-1-5601 from the U.S. Air Force, Wright Laboratory, Manufacturing Technology Directorate.

References

[1] E. A. Amerasekera and D. S. Campbell, *Failure Mechanisms in Semiconductor Devices,* John Wiley & Sons Press, 1987.

[2] A. Chatterjee, B. C. Kim and N. Nagi, "DC built-in self-test for linear analog circuits," *IEEE Design & Test of Computers,* pp. 26-33, Summer 1996.

[3] M. Hasler and J. Neirynck, *Nonlinear Circuits,* Norwood, MA: Artech House, 1986.

[4] A. S. Householder, "A survey of some closed methods for inverting matrices," *SIAM J. of Applied Mathematics,* pp. 155–169, 1957.

[5] K. S. Kundert, *The Designer's Guide to SPICE & SPECTRE,* Kluwer Academic Publishers, 1995.

[6] ftp.cbl.ncsu.edu:/pub/Benchmark_dirs/CircuitSim90/, *MCNC Circuit Simulation & Modeling Workshop,* 1990.

[7] L. Milor and A. L. Sangiovanni-Vincentelli, "Minimizing production test time to detect faults in analog circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits,* pp. 796–813, vol. 13, 1994.

[8] L. W. Nagel, *SPICE2: A Computer Program to Simulate Semiconductor Circuits,* ERL Memorandum M520, Ph.D. Dissertation, Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley, May 1975.

[9] N. Nagi, A. Chatterjee and J. A. Abraham, "Fault simulation of linear analog circuits," *Analog Integrated Circuits and Signal Processing,* pp. 245–260, vol. 4, 1993.

[10] M. J. Ohletz, "Realistic faults mapping scheme for the fault simulation of integrated analogue CMOS circuits," pp. 776–785 in *Proc. International Test Conference,* 1996.

[11] A. Pahwa and R. A. Rohrer, "Band-faults: Efficient approximations to fault bands for the simulation before fault diagnosis of linear circuits," *IEEE Trans. Circuits and Systems,* pp. 81–88, vol. 29, 1982.

[12] J. S. Roychowdhury and R. C. Melville, "Homotopy techniques for obtaining a DC solution of large-scale MOS circuits," pp. 286–291 in *Proc. IEEE/ACM Design Automation Conference,* 1996.

[13] M. Sachdev and B. Atzema, "Industrial relevance of analog IFA: A fact or a fiction," pp. 61–70 in *Proc. International Test Conference,* 1995.

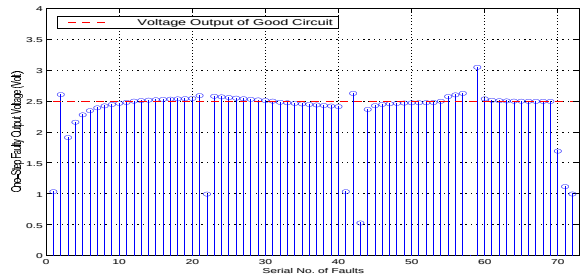
[14] K. Singhal and J. Vlach, "Solution of modified systems and applications," pp. 620–623 in *Proc. IEEE International Symposium on Circuits and Systems,* 1981.

[15] G. C. Temes, "Efficient methods of fault simulation," pp. 191–194 in *Proc. 20th Midwest Symposium on Circuits and Systems,* 1977.

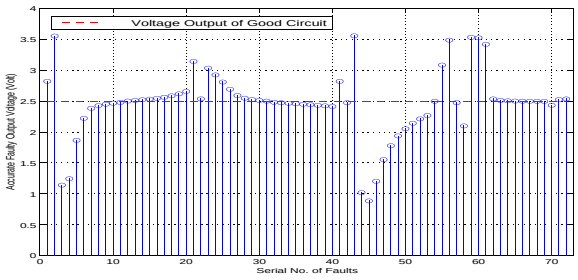
[16] M. W. Tian and C.-J. Shi, "Rapid frequency-domain analog fault simulation under parameter tolerances," pp. 275–280 in *Proc. 34th ACM/IEEE Design Automation Conference,* June 1997.

[17] D. M. Wolf and S. R. Sanders, "Multiparameter homotopy methods for finding DC operating points of nonlinear circuits," *IEEE Trans. Circuits and Systems Part I,* pp. 824–838, vol. 43, 1996.

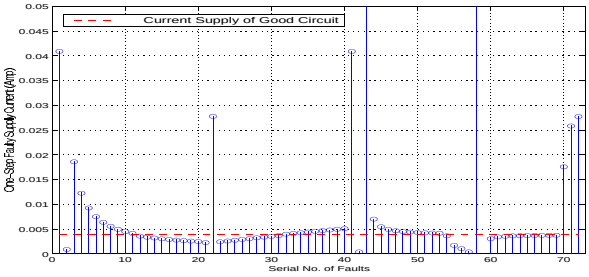
[18] W. Zwolinski, A. D. Brown and C. D. Chalk, "Concurrent analogue fault simulation," pp. 42–47 in *Proc. 3rd IEEE International Mixed-Signal Testing Workshop,* 1997.



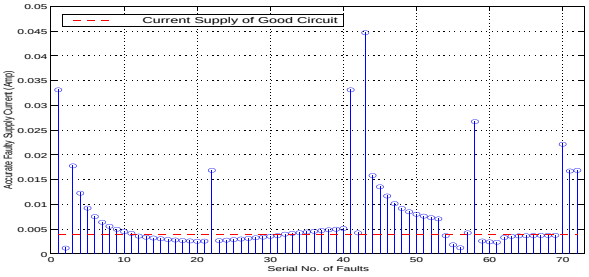
(a) Output voltage by one-step simulation.



(b) Output voltage by accurate simulation.



(c) Supply current by one-step simulation.



(d) Supply current by accurate simulation.

Fig. 5. Output voltage and supply current of static latch circuit.

TABLE II
RESULTS OF CIRCUITSIM90 BENCHMARK CIRCUIT FAULT SIMULATION.

Circuit	No. of Devices			No. of	IterNo. of	Init. Point 0		Init. Point X_{good}		Fault Ordering	
	BJT	MOS	RES	Faults	Good Sim.	IterNo.	CPU sec.	IterNo.	SpeedUp	IterNo.	SpeedUp
astabl	2	0	4	112	9	1112	1.02	372	2.99	340	3.27
bias	13	0	5	308	14	10747	30.14	8838	1.22	8074	1.33
bjtff	41	0	26	1176	25	34518	409.88	7882	4.38	6729	5.13
latch	14	0	10	72	12	1083	3.04	667	1.62	573	1.89
loc	96	0	276	180	45	8010	326.38	612	13.09	535	14.97
nagle	23	0	11	588	28	15712	77.44	5519	2.85	4874	3.22
rca	11	0	12	416	6	3438	16.52	2509	1.37	2185	1.57
schmitecl	4	0	8	224	31	13944	12.02	1407	9.91	1359	10.26
vreg	20	0	10	520	251	87568	201.60	23823	3.68	9308	9.41
ab_ac	0	31	1	1384	142	67122	563.16	12808	5.24	10468	6.41
ab_integ	0	31	3	1424	16	35207	406.40	10611	3.32	10566	3.33
ab_opamp	0	31	24	1444	11	23925	349.10	8620	2.78	8575	2.79
e1480	0	28	130	60	25	1626	12.74	324	5.02	253	6.43
gm6	0	5	0	220	7	1539	4.56	590	2.61	549	2.80
hussamp	0	16	1	724	21	59001	155.3	54862	1.08	20867	2.83
mosrect	0	4	2	216	12	2257	5.88	506	4.46	371	6.08
mux8	0	64	0	2818	11	34473	1531.56	11488	3.00	11117	3.10
nand	0	25	0	1100	6	6796	146.86	2210	3.08	1980	3.43
pump	0	1	0	44	3	132	0.40	44	3.00	44	3.00
reg0	0	13	30	600	3	1800	12.74	951	1.89	907	1.98
ring	0	34	0	1496	6	43823	467.40	57849	0.76	11699	3.75
schmitfast	0	6	0	264	8	2221	7.66	790	2.81	665	3.34
schmitslow	0	8	0	352	9	3137	13.70	1099	2.85	839	3.74
slowlatch	0	14	1	636	11	10095	51.34	4368	2.31	3833	2.63
toronto	0	58	0	2552	30	43356	1227.32	14679	2.95	12059	3.60
arom	0	116	2	5144	11	57520	6130.72	9822	5.86	9439	6.09
b330	0	330	0	14520	14	342553	33340.57	196385	1.74	115113	2.98
counter	0	220	0	9680	19	212690	27392.62	77222	2.75	57688	3.69
gm1	0	46	7	276	16	4416	31.42	4278	1.03	828	5.33
gm17	0	56	3	764	173	92810	526.56	25191	3.68	24727	3.75
gm19	0	162	1	460	23	20892	470.18	25163	0.83	13234	1.58
gm2	0	7	0	44	8	350	0.68	304	1.15	117	2.99
jge	0	348	0	15312	31	320292	27262.29	122552	2.61	73948	4.33
mike2	0	12	0	528	9	5033	30.46	5669	0.89	2297	2.19
rich3	0	106	2	4704	18	84392	5288.18	12014	7.02	9677	8.72
todd3	0	13	1	592	20	17878	60.48	4030	4.44	3842	4.65
Average	6	50	16	1971	30	46430	2193.0	19890	2.7	12213	4.4